Kernel Density Estimation for Target Trajectory Prediction

Vahab Akbarzadeh[1], Christian Gagné[2], and Marc Parizeau[3]

***Abstract*— This paper proposes the use of a kernel density estimation to measure similarities between trajectories. The similarities are then used to predict the future locations of a target. For a given environment with a history of previous target trajectories, the goal is to establish a probabilistic framework to predict the future trajectory of currently observed targets based on their recent moves. Instead of clustering trajectories into groups, we calculate the similarity between a given test trajectory and the set of all past trajectories in a dataset. Next, we use a weighted mechanism for prediction, that can be used in target tracking and collision avoidance applications. The proposed method is compared with two other commonly used similarity models (PCA and LCSS) over a dataset of simulated trajectories, and two datasets of real observations. Results show that the proposed method significantly outperforms the existing models for those datasets and experimental settings.**

I. Introduction

Trajectory prediction is the process of estimating the future location of a target moving inside an environment. A trajectory is usually modelled as a sequence of observed locations for the target. Therefore, in trajectory prediction, the goal is to predict the future location of a given target, given a history of previous target locations. Trajectory prediction has many applications in robotics. For example, in an environment where humans and robots coexist, the trajectory of the robots can be determined based on human trajectories, as the system tries to avoid possible collisions between robots and humans. Therefore, it is crucial to have an accurate estimate for the future positions of the humans [6].

A simple approach for this problem is to perform an estimation using only an on-going trajectory of the same target. Kalman Filters [4] and Particle Filters [3] are two classical methods commonly used for this purpose. The problem with these approaches is that they are not able to predict far into the future. For example, if one uses particle filtering (or Kalman filtering) to predict 10 time steps ahead, the prediction variance will grow 10 times larger than the one time step variance, which would make the method hardly usable for most applications, as the prediction uncertainty could rapidly become larger than the area under study.

A more informed approach is to exploit the history of other targets that have already traversed the same environment, to predict the trajectory of future targets. To exploit the history of previous targets, some approaches require the calculation of the similarity between two trajectories. Here, the two trajectories are that of a new target and another trajectory from the set of trajectories of previous targets. There are several approaches proposed to solve this problem. For instance, Bashir et al. [1] proposed to determine the dissimilarity between two trajectories using the Euclidean distance between the Principal Component Analysis (PCA) coefficients of each trajectory. Dynamic Time Warping (DTW) [5], and Longest Common Subsequence (LCSS) [9] are two other methods that calculate similarity through subsequence alignment. They have been shown to produce stable results [11].

The mentioned similarity-based prediction methods can be viewed as non-parametric approaches for the problem. The parametric approaches (e.g. hidden Markov models (HMM) [7], or inverse reinforcement learning [13]) have also been applied for the prediction problem.

Compared to parametric methods, non-parametric methods have desirable properties for trajectory prediction, such as their ability to work in an online fashion. In other words, with parametric methods, the addition of new trajectories in the history dataset implies the re-estimation of all system parameters, while this is not the case for non-parametric methods. This is an important property for the trajectory prediction problem, because new trajectories are detected all the time and they should be added to the history dataset. Non-parametric methods have the advantage of being insensitive to the addition of new trajectories, as each prediction is computed on-demand over the history dataset, with little or no other computation required beforehand. However, the computation required is proportional to the history dataset size, such that for a large set of trajectories observed, one should limit computation by restricting the size of the history dataset used in some way (e.g., through a sliding window). Specifically, in the case where the obstacles in the environment are mobile, the combination of online trajectory addition and the sliding variable becomes important, as the most recent trajectories should be used for prediction of new trajectories.

Besides, the parameters of a parametric method can be fine tuned for a specific problem, but usually differ from problem to problem. For example, in the HMM approaches, the choice of the number and form of hidden states (which could be line segments [7], mixture of Gaussians [2], or Voronoi diagrams [8], etc.) is different for different environments.

In this paper, we propose to use kernel density estimation (KDE) as a non-parametric method, to model the similarity between two trajectories, and to use that similarity to predict the future locations of targets. Moreover, instead of clustering trajectories into groups, we calculate the similarity between

Authors are with Laboratoire de vision et systèmes numériques, Département de génie électrique et de génie informatique, Université Laval, Québec (Québec), Canada. E-mails:
[1]Vahab.Akbarzadeh.1@ulaval.ca
[2]Christian.Gagne@gel.ulaval.ca
[3]Marc.Parizeau@gel.ulaval.ca

a given test trajectory and the set of all trajectories in the history dataset. Next, we use a weighted mechanism for prediction. The performance of the proposed similarity measure is compared with the well-known LCSS and PCA methods over a dataset of simulated trajectories and two datasets of real observations.

II. Problem Definition

Trajectory prediction is the process of estimating the future location of a given target using the information of its previous movements and the movements that other targets have made in the same environment. More precisely, let $\mathbf{T}_j^{(t)}$ represent an on-going target's trajectory in an environment Ξ, currently being observed at time t, since its arrival at time t_j. We wish to predict the location of the target after s time steps (i.e., $\widehat{\mathbf{T}}_j^{(t+s)}$) as accurately as possible.

We build a probabilistic model to represent the target's future location. Let $X_j^{(t+s)}$ denote over the sample space of all locations $\mathbf{q} \in \Xi$, a discrete random variable for target $\mathbf{T}_j$ at time $t+s$. Variable $X_j^{(t+s)}$ is defined by a probability mass function $P(X_j^{(t+s)} = \mathbf{q})$ which returns the probability of target $\mathbf{T}_j$ being at location $\mathbf{q}$ at time $t+s$. For brevity, we note this value as $\mathcal{P}_{j\mathbf{q}}^{(t+s)}$.

The goal is to incrementally learn the probabilistic model for each target as it moves within the environment, using the information of N previously observed target trajectories $\mathbf{T}_i$ in the same environment, $\mathbb{T} = \{\mathbf{T}_1, \mathbf{T}_2, \ldots, \mathbf{T}_N\}$.

The performance of the trajectory prediction method is measured using the expected value of distance between the prediction and the actual location of the target. More precisely, using the law of the unconscious statistician, the prediction error $E_j^{(t+s)}$ for location $\mathbf{p}_j^{(t+s)}$ of a test trajectory $\mathbf{T}_j$, at time $t+s$, is defined as:

$$E_j^{(t+s)} = \sum_{\mathbf{q} \in \Xi} \mathcal{P}_{j\mathbf{q}}^{(t+s)} \left\| \mathbf{q} - \mathbf{p}_j^{(t+s)} \right\|_2, \tag{1}$$

where the sum of the Euclidean distances ($||\cdot||_2$) between the actual location of the target and all possible positions $\mathbf{q}$ in the environment is weighted by the predicted probability $\mathcal{P}_{j\mathbf{q}}^{(t+s)}$ that the target will be at position $\mathbf{q}$. The goal of the prediction method is to identify the prediction mechanism which gives the minimum value for the prediction error:

$$\mathcal{P}_{j\mathbf{q}}^{*(t+s)} = \underset{\mathcal{P}_{j\mathbf{q}}^{(t+s)}}{\operatorname{argmin}} \; E_j^{(t+s)}. \tag{2}$$

III. The Proposed Method

We propose to carry out the prediction using the similarity measure between the trajectories. The principal assumption is that the similarity between the previous locations and displacements of two trajectories is directly related to the closeness of their future locations. The main problem is how to define the similarity between two trajectories.

More formally, at a time step t, we wish to measure the similarity $S(\mathbf{T}_j^{(t)}, \mathbf{T}_m)$ between the current state $\mathbf{T}_j^{(t)}$ of an ongoing trajectory and the whole history of another trajectory $\mathbf{T}_m \in \mathbb{T}$. Then, we form a four dimensional space, where the state of a trajectory at one time step is represented by a point in this space, consisting of its current location $\mathbf{p}_j^{(t)} = (x_j^{(t)}, y_j^{(t)})$ and its most recent displacement $\Delta\mathbf{p}_j^{(t)} = (\Delta x_j^{(t)}, \Delta y_j^{(t)})$. As a result we have $\mathbf{T}_j^{(t)} = \{x_j^{(t)}, y_j^{(t)}, \Delta x_j^{(t)}, \Delta y_j^{(t)}\}$.

In the mentioned four dimensional space we estimate the density of trajectory $\mathbf{T}_m$, and we define the similarity between the state of trajectory $\mathbf{T}_j$ at time t, as its value within the density estimated for the trajectory $\mathbf{T}_m$. More formally we have:

$$S(\mathbf{T}_j^{(t)}, \mathbf{T}_m) = \widehat{F}_{\mathbf{T}_m}(\mathbf{T}_j^{(t)}), \tag{3}$$

where $S(.,.)$ is the similarity function and $\widehat{F}_{\mathbf{T}_m}(\mathbf{T}_j^{(t)})$ is the density estimated for trajectory $\mathbf{T}_m$ evaluated at point $\mathbf{T}_j^{(t)}$. For density estimation, we use the non-parametric multivariate kernel method. More precisely we have:

$$\widehat{F}_{\mathbf{T}_m}(\mathbf{z}) = \frac{1}{n_m} \sum_{t=1}^{n_m} \mathbf{K}(\mathbf{T}_m^{(t)}, \mathbf{z}, \mathbf{h}_m), \tag{4}$$

where n_m is the total number of displacements of target $\mathbf{T}_m$ and $\mathbf{h}_m = \{h_1, h_2, h_3, h_4\}$ are the bandwidth parameters for trajectory $\mathbf{T}_m$ along the four mentioned dimensions. These parameters define the length-scale of the function for location and displacement, respectively. Informally speaking, h_1, and h_2 define how far two locations should be in the environment to be considered far (along the x and y dimensions), and h_3, and h_4 define how much the directions of the two displacements should diverge before they are considered different (along the Δx and Δy dimensions).

For the kernel function $\mathbf{K}(.,.,.)$, we use the normal distribution defined as:

$$\mathbf{K}(\mathbf{z}, \mathbf{z}', \mathbf{h}) = \prod_{d=1}^{4} \frac{1}{h_d} K\left(\frac{z_d - z_d'}{h_d}\right), \tag{5}$$

$$K(u) = \frac{1}{\sqrt{2\pi}} \exp\left(-\frac{u^2}{2}\right). \tag{6}$$

A. The problem of missing observations

The mentioned similarity function is defined in four dimensions and therefore difficult to visualize. As a result we present the behaviour of the similarity function in a one dimensional space (Fig. 1). In this figure, there are two consecutive observations at positions 1 and 5, and we observe that the similarity is higher close to those points. In the figure, the dotted lines represent the underlying normal distributions we mapped over each observation.

The assumption behind the similarity function is that the trajectory of the history target $\mathbf{T}_m$ has been observed at all time steps. As we will see in Sec. IV, this is not the case in most real datasets. In reality the time step difference between consecutive observations of the same trajectory might be different. For example, assume that for the data presented in Fig. 1, the time difference between the two observations is four time steps (instead of one). The current model is not compatible with this assumption and therefore should be modified. One possibility is to evenly divide the space

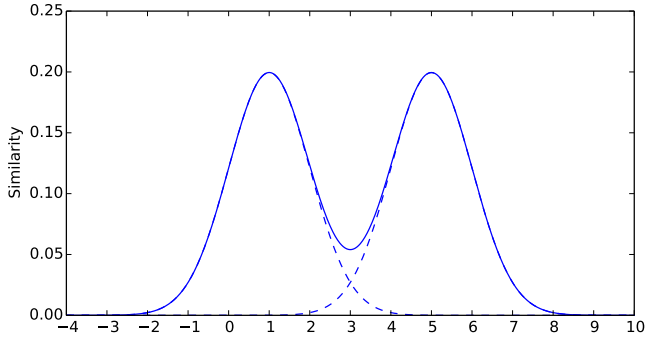

Fig. 1: The behaviour of the similarity function in one dimension.

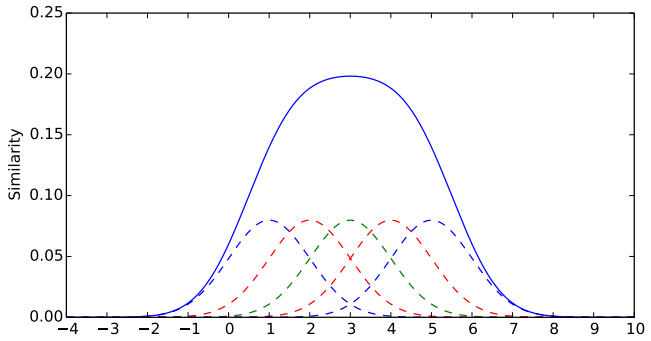

Fig. 2: The behaviour of the similarity function with unknown intermediate observations and fixed bandwidth parameters.

between the two observations and add each interpolated point as an observation. In this model all the newly created observations would have equal bandwidth parameters $\mathbf{h}$. This model is presented in Fig. 2.

The problem with this approach is that it assumes that the intermediate observations, which we did not actually observe, have the same bandwidth parameter as the actual observations we made. For the example given in Fig. 2, the similarity at point 3 is higher than that of point 5, while we had an observation at point 5, but none at point 3.

For that reason, we propose a model in which the bandwidth parameter of the interpolated observations increase linearly as the interpolated point is located further away from an actual observation. Fig. 3 represents our model for the example given before. As can be seen, the bandwidth parameter is the same for the points where we had an observation (points 1 and 5), and the bandwidth parameter increases as we move away from the observations. In the same way, we modify the similarity equation (Eq. 4), so that the bandwidth parameter is a function of the observation, and therefore not fixed for all points. The modified version of the similarity equation (Eq. 4) is as follows:

$$\widehat{F}_{\mathbf{T}_m}(\mathbf{z}) = \frac{1}{n_m} \sum_{t=1}^{n_m} \mathbf{K}(\mathbf{T}_m^{(t)}, \mathbf{z}, \mathbf{h}_m^{(t)}), \tag{7}$$

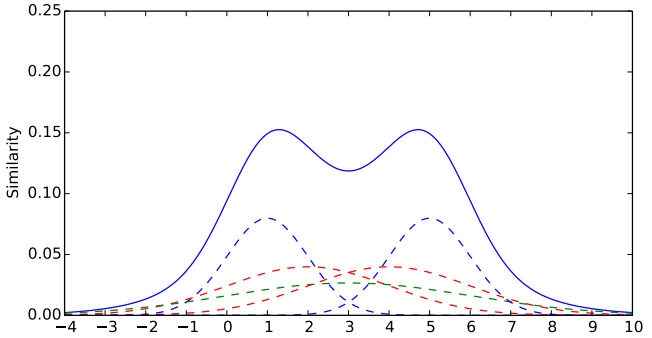

Fig. 3: The behaviour of the similarity function with unknown intermediate observations and variable bandwidth parameters.

where, in the new formula, the bandwidth parameter $\mathbf{h}_m^{(t)}$ is defined for each trajectory m at each time step t.

B. Parameter estimation

The only parameters of the KDE method that should be decided are the bandwidth parameters. As previously mentioned, the bandwidth parameters define the distance at which two points are considered close. Therefore, a means of determining the value of the bandwidth parameters for the interpolated points is needed.

We are proposing to calculate the bandwidth parameters by first interpolating the intermediate points. Following this, the bandwidth parameters are computed by maximizing a leave-one-out cross-validation likelihood:

$$\widehat{F}_{\mathbf{T}_m - i}(\mathbf{T}_m) = \frac{1}{n_m - 1} \sum_{t=1, t\neq i}^{n_m} \mathbf{K}(\mathbf{T}_m^{(t)}, \mathbf{T}_m^{(i)}, \mathbf{h}_m). \tag{8}$$

In other words, in order to evaluate a parameter $\mathbf{h}_m$ for trajectory $\mathbf{T}_m$, only one of the displacements of the trajectory ($\mathbf{T}_m^{(i)}$) is removed from the summation and the likelihood of the similarity function is evaluated at that point. For a given bandwidth parameter, the process is repeated for all displacements in the trajectory. Therefore the final process is:

$$\prod_{i=1}^{n_m} \widehat{F}_{\mathbf{T}_m - i}(\mathbf{T}_m) = \frac{1}{(n_m-1)^{n_m}} \prod_{i=1}^{n_m} \sum_{t=1, t\neq i}^{n_m} \mathbf{K}(\mathbf{T}_m^{(t)}, \mathbf{T}_m^{(i)}, \mathbf{h}_m). \tag{9}$$

In the given formula the multiplicands are usually small and the final product leads to underflow. Therefore it is more convenient to maximize the log of the likelihood:

$$\sum_{i=1}^{n_m} \log\left[\widehat{F}_{\mathbf{T}_m - i}(\mathbf{T}_m)\right] = -n_m \log(n_m - 1) + \sum_{i=1}^{n_m} \log\left[\sum_{t=1, t\neq i}^{n_m} \mathbf{K}(\mathbf{T}_m^{(t)}, \mathbf{T}_m^{(i)}, \mathbf{h}_m)\right]. \tag{10}$$

Taking the derivatives of the given formula with respect to the bandwidth parameters leads to complex equations, which cannot be solved in a closed form. Therefore, we are

proposing to adjust the parameters of the method by a line search conducted within the range $[1, 20]$ in increments of 0.5, independently for each bandwidth parameter. Indeed, according to our formulation of the Gaussian kernel (see Eq. 5), the effect of each bandwidth parameter on the likelihood is not related to the others. The best value for each bandwidth parameter is chosen as the value of the parameter for each trajectory.

Notice that the mentioned formula defines the bandwidth parameter for the observed points. For each interpolated point, we define a level property $L(\mathbf{T}_j^{(t)})$, which measures the distance between the mentioned point and the closest observed point. For the example given in Fig. 2, points 2 and 4 have level 2, and point 3 has level 3. The observed points (1 and 5) both have level 1. Using this formula the bandwidth parameter is defined as:

$$\mathbf{h}_j^{(t)} = L(\mathbf{T}_j^{(t)}) \times \mathbf{h}_j, \tag{11}$$

where $\mathbf{h}_j$ is the base bandwidth parameter calculated using the method proposed in Eq. 10.

C. Prediction using the similarity function

Once the similarities have been calculated, at each time step, the similarity function behaves as a weighting parameter that can be used to predict the future locations of target $\mathbf{T}_j^{(t)}$ as a weighted sum of the trajectory of the previous targets. More precisely we have:

$$w_{ji}^{(t)} = \frac{S(\mathbf{T}_j^{(t)}, \mathbf{T}_i)}{\sum_{\mathbf{T}_m \in \mathbb{T}} S(\mathbf{T}_j^{(t)}, \mathbf{T}_m)}, \tag{12}$$

where $w_{ji}^{(t)}$ is the normalized similarity between the displacement of target $\mathbf{T}_j$ at time t and the whole trajectory $\mathbf{T}_i \in \mathbb{T}$. Using this formulation, the future location of a target is a discrete probability distribution over environment Ξ, where the probability of the target associated with $\mathbf{T}_j$ being at each location $\mathbf{q} \in \Xi$ after s time steps from current time t is defined as:

$$\mathcal{P}_{j\mathbf{q}}^{(t+s)} = \sum_{\mathbf{T}_m \in \mathbb{T}} w_{jm}^{(t)} \mathbf{1}(\mathbf{T}_m^{(\theta^*+s)}, \mathbf{q}), \tag{13}$$

where function $\mathbf{1}(\mathbf{T}_m^{(\theta^*+s)}, \mathbf{q})$ returns one if the target that generated $\mathbf{T}_m$ was at location $\mathbf{q}$ at time $\theta^* + s$, and zero otherwise.

In order to predict the next locations of trajectory $\mathbf{T}_j^{(t)}$ at each time step among all of the displacements $\mathbf{T}_m^{(\theta)}$ of a sample trajectory $\mathbf{T}_m$, the displacement which is most similar with the current displacement $\mathbf{T}_j^{(t)}$ of target $\mathbf{T}_j$ should be found. This most similar displacement occurs at time step:

$$\theta^* = \underset{\theta}{\operatorname{argmax}}\, S(\mathbf{T}_m^{(\theta)}, \mathbf{T}_j^{(t)}). \tag{14}$$

The whole process of trajectory prediction and performance evaluation is shown for a simple example in Fig. 4. As before, $\mathbf{T}_j$ is the test trajectory for which we want to predict future locations, and we have the history of two reference trajectories $\mathbf{T}_1$ and $\mathbf{T}_2$. We will use the reference trajectories for predicting the location of the test trajectory. Target $\mathbf{T}_j$ is currently at the third time step, $t = 3$, and we want to determine its location after four time steps, $s = 4$.

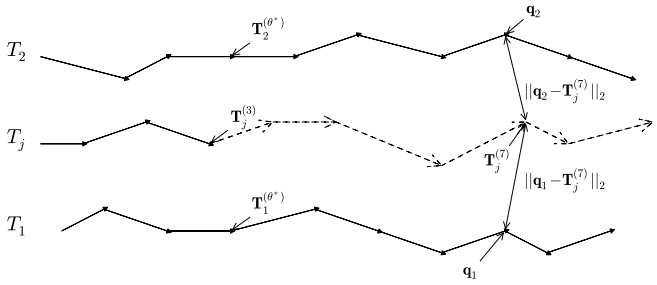

Fig. 4: Target trajectory prediction process.

First, the similarity between the last displacement of the test trajectory ($\mathbf{T}_j^{(3)}$) and the reference trajectories is calculated. Let us assume that $S(\mathbf{T}_j^{(3)}, \mathbf{T}_1) = 0.2$, and that $S(\mathbf{T}_j^{(3)}, \mathbf{T}_2) = 0.3$. Therefore (using Eq. 12) we have $w_{j1}^{(3)} = 0.4$, and $w_{j2}^{(3)} = 0.6$. Next, we use Eq. 14 to determine the displacements in the reference trajectories that maximize similarity with the current displacement of the test trajectory ($\mathbf{T}_1^{(\theta^*)}$, $\mathbf{T}_2^{(\theta^*)}$). A non-zero prediction probability is assigned to the two reference trajectories, which are used to predict the future locations of the test trajectory ($\mathbf{T}_j^{(7)}$).

To evaluate the performance, the distance between our prediction and the actual location of the test trajectory is calculated ($\|\mathbf{q_1} - \mathbf{p}_j^{(7)}\|_2$,$\|\mathbf{q_2} - \mathbf{p}_j^{(7)}\|_2$). Based on this result the performance of our prediction would be $E_j^{(7)} = 0.4 \times \|\mathbf{q_1} - \mathbf{p}_j^{(7)}\|_2 + 0.6 \times \|\mathbf{q_2} - \mathbf{p}_j^{(7)}\|_2$.

IV. Experiments

In the following, we are presenting an experimental comparison of the proposed KDE-based similarity measure with two other well-known similarity measures, LCSS and PCA. Three different experimental settings are tested, one with simulated trajectories that we have designed and two configurations based on datasets of real targets tracked in 2D.

The trajectory prediction is computed in an online fashion. Thus, the history of reference trajectories $\mathbb{T}$ begin by containing the first trajectory $\mathbb{T} = \{\mathbf{T}_1\}$. Then, every time a new trajectory $\mathbf{T}_j$ appears in the environment, its trajectory is predicted using the reference trajectories in $\mathbb{T}$, and once its trajectory is completed, it is added to the reference trajectories $\mathbb{T} = \mathbb{T} \cup \mathbf{T}_j$.

As mentioned previously in Sec. III-A, in most real datasets the time difference between consecutive displacements of trajectories is not constant. This is why the variable bandwidth parameter was defined for the kernel density estimation method. This is also the case for the two real datasets we present here. In our experiments, we used only the trajectories for which the time difference between consecutive displacements is less than 10 time steps. If the

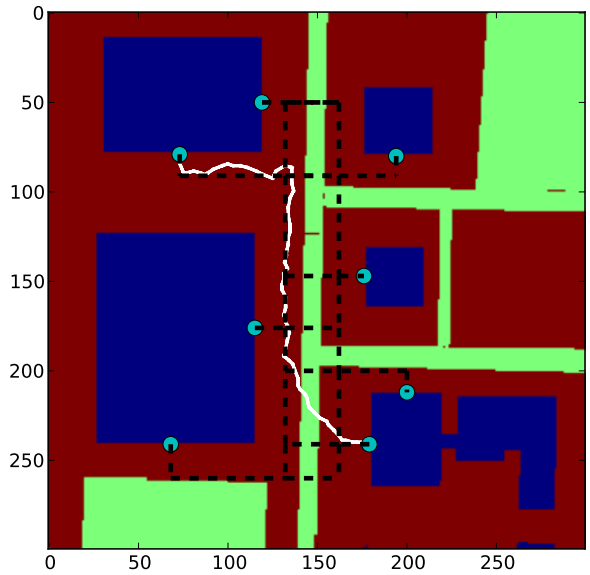

Fig. 5: ULaval campus map and simulation setting. Eight gates are used, each gate is represented by a cyan circle. Targets can enter through any of the eight gates, walk around the campus and exit the environment through another gate. The pedestrian paths inside the campus are shown using the dashed black line. The target follows the path to move from one gate to another. The white line represents the trajectory of a sample target.

time difference is more than 10 time steps, we split the trajectory into two trajectories, and treat each one as an independent trajectory. The reason for this selection is that if the time difference is more than 10 time steps, the two parts of the trajectory can diverge so much that it is not feasible to consider them as a single trajectory.

As the final criteria, we only select the trajectories whose length is at least 35 steps. The reason is that in real datasets, there are some short trajectories which do not add useful information for the prediction of other trajectories, and therefore we did not include them in the dataset.

A. ULaval campus simulated dataset

This simulated dataset was generated by simulating pedestrians walking on part of the Université Laval campus, in Québec City, Canada. The environment is shown in Fig. 5. In this figure, the blue colour represents the buildings, green represents the street and parkings, and red is the ground. The target enters the environment through one gate of a building (cyan circles in the figure), walks inside the environment, and exits through another gate. The trajectory of each target is generated based on its current location and the temporary goal location it is aiming at, which is based on the shortest path between the initial and final gates on the pedestrian path. The pedestrian path in the figure is represented by the dashed black line. In the simulation, the pedestrian path is represented as a graph with intersections being the vertices and the paths themselves being the edges. Each intersection

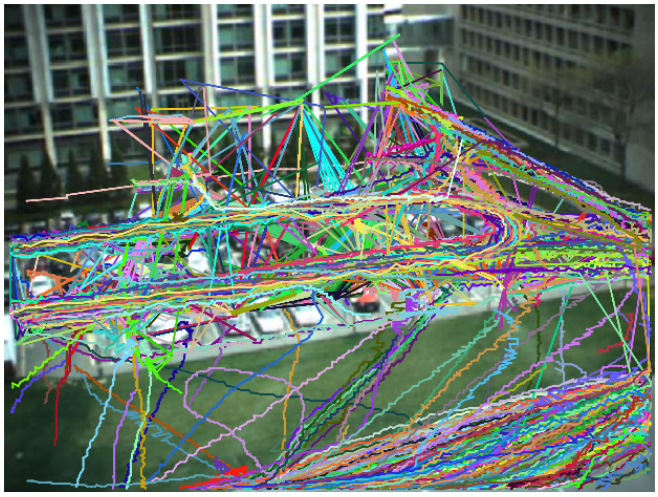

Fig. 6: MIT Trajectory dataset [10]. Trajectories were generated from real targets moving in a parking lot within five days. One thousand of the 36 075 trajectories used for experiments are shown in the image.

(vertex) on the shortest path between the two gates can be a temporary goal for the target. In order to add some randomness to the system, and to make the simulation more realistic, the targets sometimes skip a temporary goal (vertex) in their shortest path and go directly to the next vertex. In the figure, this phenomena is shown at the bottom, where the target diagonally crosses the street.

At each time step, the next location of a target is randomly chosen from a distribution that results from the (normalized) multiplication of a beta and a Gaussian distribution. More precisely, at each time step, a normal distribution is applied on the angle between the target's location and its temporary goal, and a beta distribution is applied on the distance to determine the step size. Parameters are $\alpha = 2$ and $\beta = 2$ for the beta distribution and $\mu = 0$ and $\sigma^2 = 125$ for the normal distribution. We produced a total of 1000 trajectories for our experiments.

B. MIT trajectory dataset

The MIT Trajectory dataset [10] contains information concerning the trajectories of 40 453 targets moving in a parking lot. The information was gathered using a single camera for five days. Using the criteria mentioned in the beginning of the section we selected 36 075 among all of the trajectories in the dataset. The parking lot and 1000 of the chosen trajectories used for our experiments is presented in Fig. 6.

C. Train station dataset

The Train Station dataset [12] contains the trajectories of 47 866 targets moving in the New York Grand Central Station. The data comes from a 30 minute video with a resolution of 480×720 pixels at 24 fps. Similarly to the parking lot dataset, only those trajectories who satisfied the criteria of time difference between consecutive displacements

Fig. 7: Train Station dataset [12]. The trajectories were generated from the targets moving in the train station within 30 minutes. A thousand of the 46 741 trajectories used for experiments are shown in the image.

were extracted and used for the simulations in this paper, resulting in a subset of 46 741 such trajectories. The train station and 1000 of these trajectories are shown in Fig. 7.

D. Longest common subsequence (LCSS)

The proposed similarity measure given in Eq. 7 is compared with the well-known similarity measure called Longest Common Subsequence (LCSS) algorithm [9]. It consists in identifying the longest similar subsequence between two trajectories using dynamic programming. More precisely, for two trajectories that end with states $\mathbf{T}_i^{(x)}$ and $\mathbf{T}_j^{(y)}$, the recursive $LCSS$ function is defined as:

$$LCSS_{\epsilon,\delta}(\mathbf{T}_i^{(x)}, \mathbf{T}_j^{(y)}) = \begin{cases} 0 & \text{if } x < 1 \text{ or } y < 1 \\ 1 + LCSS_{\epsilon,\delta}(\mathbf{T}_i^{(x-1)}, \mathbf{T}_j^{(y-1)})) & \\ \qquad \text{if } \|\mathbf{p}_i^{(x)} - \mathbf{p}_j^{(y)}\|_2 < \epsilon \text{ and } |x-y| < \delta & , \\ \max(LCSS_{\epsilon,\delta}(\mathbf{T}_i^{(x)}, \mathbf{T}_j^{(y-1)}), LCSS_{\epsilon,\delta}(\mathbf{T}_i^{(x-1)}, \mathbf{T}_j^{(y)})) & \\ \qquad \text{otherwise} & \end{cases} \tag{15}$$

where ϵ defines the maximum acceptable distances between corresponding target positions, and δ is the maximum allowable time warp. From this definition, the similarity between two trajectories is defined as:

$$S(\mathbf{T}_i, \mathbf{T}_j) = \frac{LCSS_{\epsilon,\delta}(\mathbf{T}_i^{(n_i)}, \mathbf{T}_j^{(n_j)})}{\min(n_i, n_j)}. \tag{16}$$

For our experiments, we replace the KDE-based measure of Eq. 12 by the LCSS-based one, and allow the other equations to be the same for the two methods. The hyper-parameters δ and ϵ of the LCSS method were optimized by trial-and-error through a grid search. The δ parameter was defined as a function of the length of the two trajectories that the LCSS algorithm was applied to. The same value for the δ parameter was derived for all the datasets, that is 20% of the maximum length of the two trajectories, in line with the suggestion made in [9]. For the ϵ parameter, different values were derived for each dataset, that is $\epsilon_{sim} = 1$ for the ULaval Campus dataset, $\epsilon_{MIT} = 5$ for the MIT Trajectory dataset, and $\epsilon_{station} = 31$ for the Train Station dataset.

E. PCA-based similarity

We also compared our proposal with a second approach to evaluate the similarity between different trajectories, using the Euclidean distance between the Principal Component Analysis (PCA) coefficients of the trajectories [1]. For this purpose, all trajectories are first resampled to the median size of the trajectories in the dataset, in order to resize trajectories to the same length. Then, a PCA is applied to these trajectories (PCA is applied separately to x and y coordinates of each trajectory). The number of coefficients kept from the PCA was selected in order to retain at least 95 % of the variance (denoted by K for each dataset). From this, the following similarity measure is devised:

$$S(\mathbf{T}_i, \mathbf{T}_j) = \frac{1}{\sum_{k=1}^{K} \left[(\gamma_x^k)^2 + (\gamma_y^k)^2)\right] + 1}, \tag{17}$$

where γ_x^k and γ_y^k are the distance between the k-th PCA coefficient of the two trajectories along the x and y axes.

In our experiments, we observed that matching a full trajectory $\mathbf{T}_i$ with part of another trajectory $\mathbf{T}_j$ using the PCA method produces poor results. Therefore, in Eq. 17, we evaluated the similarity between the sub-trajectory of the two trajectories. In other words, assuming that the test target $\mathbf{T}_j$ is at time step t, when computing the similarity between this target and another target $\mathbf{T}_i$, we consider only the first t displacements of the two trajectories for the PCA calculation.

V. Results

Four different configurations were used for the experiments. In each setting, t shows the current time in the test trajectory, and s is the number of time steps in the future. For example, when $t = 5$ and $s = 10$, the target is at the 5-th time step and we want to predict its location at the 15-th time step.

The similarity based prediction method presented here is categorized as a non-parametric method. Therefore, the performance of the mentioned methods depends on the size of the history of reference trajectories used for prediction. We performed some experiments and observed that as the size of the history of reference trajectories exceeds 1000 trajectories, the increases in size does not significantly affect the performance of the prediction method (see Fig. 8 and 9). Therefore, the trajectory history was maintained as a sliding window with the size of 1000. This means that at each time step, in order to predict the trajectory of a new target, only the trajectories of the previous 1000 targets are used. In this manner, the prediction time for each trajectory remains manageable for the large datasets (MIT and Train Station).

The performances of the KDE, LCSS, and PCA prediction methods are reported in Table I. Each method was run

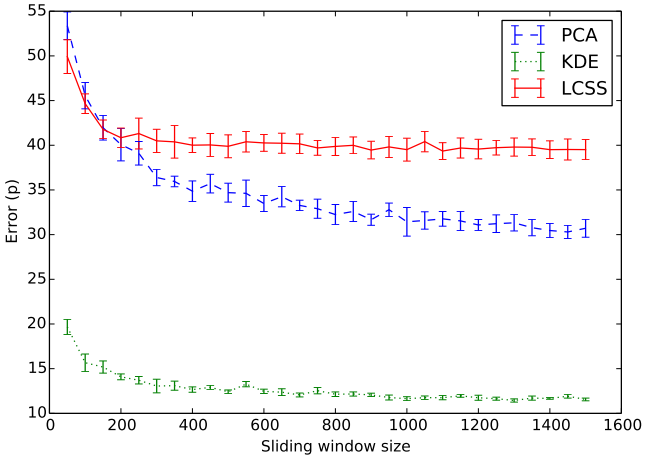

Fig. 8: Effect of the sliding window size on the performance of different prediction methods over the MIT trajectory dataset.

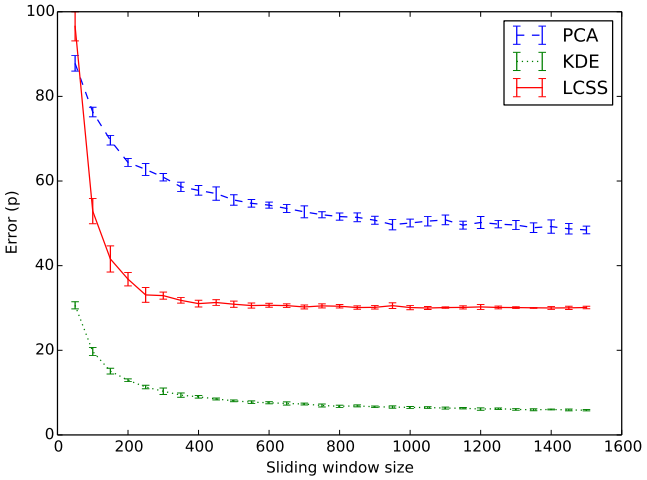

Fig. 9: Effect of the sliding window size on the performance of different prediction methods over the Train station dataset.

30 times, where each run used a different order for the processing of the trajectories, although the same order was used for each method to be fair. Therefore, at each run for KDE, LCSS, and PCA a different set of trajectories were used for the prediction of each trajectory. Reported results are average performances over 30 runs. In this table, the average distance between the prediction made by each method and the actual location of the target is reported (as defined by Eq. 1).

It can be seen that the KDE method outperformed the LCSS and PCA methods considering the error measure on all datasets and with all simulation settings. The reason for the strong performance of the KDE method is twofold. First, the KDE method is the only method which considers both the location and the displacement in the similarity function. The addition of displacement in the similarly function helps the KDE method to differentiate more clearly between targets

TABLE I: Average error, standard deviation of error, and CPU time using KDE-, LCSS- and PCA-based trajectory similarities. Error values are calculated using Eq. 1, smaller values are better. CPU time is the Wall Clock time needed to estimate the location of one target. Tested settings correspond to different current times (t) and prediction time steps (s). Bold values represent statistically significant results according to the Wilcoxon signed-rank test (p-value=0.05) when compared pairwise with both other methods.

		$t=5$ $s=5$	$t=5$ $s=20$	$t=15$ $s=5$	$t=15$ $s=20$
ULaval Campus simulated dataset					
KDE	Avg. err.	**10.04 m**	**44.18 m**	**11.45 m**	**35.52 m**
	Stdev. err.	0.27 m	1.2 m	0.3 m	0.74 m
	CPU time	4.0 s	4.5 s	3.5 s	3.5 s
LCSS	Avg. err.	15.62 m	53.66 m	31.18 m	67.22 m
	Stdev. err.	0.34 m	1.2 m	0.38 m	.92 m
	CPU time	1.5 s	1.5 s	4.0 s	4.0 s
PCA	Avg. err.	15.7 m	55.22 m	21.04 m	51.16 m
	Stdev. err.	0.4 m	0.87 m	0.6 m	1.05 m
	CPU time	1.0 s	1.0 s	1.5 s	1.5 s
MIT Trajectory dataset					
KDE	Avg. err.	**11.91 p**	**26.0 p**	**10.95 p**	**24.10 p**
	Stdev. err.	0.7 p	0.6 p	0.4 p	0.6 p
	CPU time	4.0 s	4.0 s	4.0 s	3.8 s
LCSS	Avg. err.	27.62 p	37.61 p	28.73 p	40.72 p
	Stdev. err.	1.2 p	1.8 p	0.9 p	0.6 p
	CPU time	1.4 s	1.5 s	4.4 s	4.3 s
PCA	Avg. err.	31.58 p	44.41 p	36.12 p	48.62 p
	Stdev. err.	0.9 p	0.6 p	1.8 p	0.5 p
	CPU time	1.0 s	1.0 s	1.5 s	1.5 s
Train Station dataset					
KDE	Avg. err.	**12.99 p**	**17.86 p**	**12.66 p**	**18.06 p**
	Stdev. err.	0.4 p	0.51 p	0.41 p	0.6 p
	CPU time	4.5 s	4.9 s	4.8 s	4.7 s
LCSS	Avg. err.	30.42 p	31.17 p	34.52 p	36.42 p
	Stdev. err.	0.2 p	0.1 p	1.4 p	1.0 p
	CPU time	2.4 s	2.3 s	4.0 s	4.1 s
PCA	Avg. err.	52.01 p	55.98 p	53.0 p	57.71 p
	Stdev. err.	0.6 p	0.6 p	0.4 p	0.8 p
	CPU time	1.0 s	1.0 s	1.5 s	1.5 s

which pass through the same location in the environment but in different directions (or the same direction but with different speeds).

Secondly, KDE is the only probabilistic method among the three. The probabilistic foundations of the method helps it to make smoother decisions while estimating similarity. For example, through its ϵ and δ parameters, the LCSS method makes binary decisions in matching the subparts of two trajectories. Thus if the distance between two parts is less than the mentioned parameter values the two parts are similar and not otherwise. In contrast, the similarity measure defined by the probability density function of the KDE method changes more smoothly from one to zero.

In general, we observe that the PCA method's performance is inferior to that of the other two methods. One reason could be that in PCA, as mentioned before in section IV-E, new trajectories are matched with the beginning sub-part of the trajectories in the history dataset. In consequence, there can be trajectories of different size with different PCA coefficients, even if the two trajectories had some common

sub-parts.

Considering the computational demand, PCA is the least computationally expensive method. The computational cost of the LCSS method depends on the prediction times. For the smaller prediction times ($t = 5$), the computational cost is comparable with the PCA method, but for the larger prediction times ($t = 15$), the computation time is similar to that of the KDE method.

In all settings, increasing the prediction time step from ($s = 5$) to ($s = 20$) has increased the expected error, which is reasonable – predicting further into the future increases the uncertainty. The interesting point is that for each dataset, increasing the prediction time step has roughly the same effect over different methods (i.e., the increase of the expected error is comparable for different methods over one dataset). This suggests that the properties of the dataset (i.e., nature of the trajectories composing the dataset) has a more significant impact over long term prediction accuracy than the prediction methods. For example, the simulated dataset has the largest drop in performance between the short ($s = 5$) and long ($s = 20$) prediction time steps. This could be related to the fact that the targets change direction more frequently in the simulated dataset than in the other two datasets. In this dataset, targets share part of their trajectory with many other targets, unlike the other two datasets, where each target usually keeps the initial direction of its movement.

Another interesting observation is that the increase of the prediction time from ($t = 5$) to ($t = 15$) has decreased the performance of the LCSS and PCA algorithms, but in most cases improved the performance of the KDE algorithm. When increasing the prediction time, each algorithm has more information to process. If done properly, as in the case of KDE, this leads to a decrease in the number of false positives (wrongfully considering two trajectories similar, while they are not). Otherwise, as in LCSS and PCA, this only increases the confusion of the overall algorithm, and reduces the performance of the system.

Another explanation is that KDE has a mechanism to deal with missing data points by taking into account the uncertainty (i.e., variance) over the estimated location of the target, while the two other methods rely only in the interpolated positions. That could be the reason that the performance of different methods is closer on the simulated dataset (in which there is no unobserved trajectory) compared to the two other real datasets where there are unobserved points.

VI. Conclusion

This paper has presented a model to measure similarity between trajectories of targets moving inside an environment using the kernel density estimation. The effectiveness of the proposed model is shown in the trajectory prediction problem. The goal in trajectory prediction is to estimate the future location of a new target moving in an environment using the history of previous trajectories that were observed so far in the same environment.

For experiments conducted on one simulated and two real datasets, the proposed model was compared with two other commonly used methods, namely PCA and LCSS, for measuring the similarity between trajectories. Results showed that although the proposed approach is slightly more expensive in terms of computation time, it outperformed both of the other methods over all tested datasets and simulation settings. The strong performance of the method is related to its probabilistic nature and consideration of the displacement in the similarity function.

Acknowledgements

This work was supported through funds from NSERC (Canada) and access to computational resources of Calcul Québec / Compute Canada. We thank Annette Schwerdtfeger for proofreading this manuscript.

References

[1] Faisal I Bashir, Ashfaq A Khokhar, and Dan Schonfeld. Segmented trajectory based indexing and retrieval of video data. In *Proc. of the Intl Conf. on Image Processing (ICIP 2003)*, volume 2, pages 623–626, 2003.

[2] Maren Bennewitz, Wolfram Burgard, and Sebastian Thrun. Learning motion patterns of persons for mobile service robots. In *Proc. of the IEEE Intl Conf. on Robotics and Automation (ICRA 2002)*, volume 4, pages 3601–3606, 2002.

[3] Neil J Gordon, David J Salmond, and Adrian FM Smith. Novel approach to nonlinear/non-gaussian bayesian state estimation. *IEE Proceedings F (Radar and Signal Processing)*, 140:107–113, 1993.

[4] Rudolph Emil Kalman. A new approach to linear filtering and prediction problems. *Journal of Basic Engineering*, 82(1):35–45, 1960.

[5] Eamonn J Keogh and Michael J Pazzani. Scaling up dynamic time warping for datamining applications. In *Proc. of ACM Intl Conf. on Knowledge Discovery and Data Mining (KDD 2000)*, pages 285–289, 2000.

[6] Markus Kuderer, Henrik Kretzschmar, Christoph Sprunk, and Wolfram Burgard. Feature-based prediction of trajectories for socially compliant navigation. In *Robotics: Science and Systems*, 2012.

[7] Cynthia Sung, Dan Feldman, and Daniela Rus. Trajectory clustering for motion prediction. In *Proc. of the IEEE/RSJ Intl Conf. on Intelligent Robots and Systems (IROS 2012)*, pages 1547–1552, 2012.

[8] Dizan Vasquez, Thierry Fraichard, and Christian Laugier. Growing hidden Markov models: An incremental tool for learning and predicting human and vehicle motion. *International Journal of Robotics Research*, 28(11-12):1486–1506, 2009.

[9] Michail Vlachos, George Kollios, and Dimitrios Gunopulos. Discovering similar multidimensional trajectories. In *Proc. of the Intl Conf. on Data Engineering (ICDE 2002)*, pages 673–684, 2002.

[10] Xiaogang Wang, Eric Grimson, Gee-Wah Ng, and Keng Teck Ma. Trajectory analysis and semantic region modeling using a nonparametric bayesian model. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2008)*, 2008.

[11] Zhang Zhang, Kaiqi Huang, and Tieniu Tan. Comparison of similarity measures for trajectory clustering in outdoor surveillance scenes. In *Proc. of the Intl Conf. on Pattern Recognition (ICPR 2006)*, volume 3, pages 1135–1138, 2006.

[12] Bolei Zhou, Xiaogang Wang, and Xiaoou Tang. Understanding collective crowd behaviors: Learning a mixture model of dynamic pedestrian-agents. In *Proc. of the IEEE Conf. on Computer Vision and Pattern Recognition (CVPR 2012)*, pages 2871–2878, 2012.

[13] Brian D. Ziebart, Nathan Ratliff, Garratt Gallagher, Christoph Mertz, Kevin Peterson, J. Andrew Bagnell, Martial Hebert, Anind K. Dey, and Siddhartha Srinivasa. Planning-based prediction for pedestrians. In *Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IROS'09, pages 3931–3936, 2009.