Trajectory saliency detection using consistency-oriented latent codes from a recurrent auto-encoder

Léo Maczyta, Patrick Bouthemy, Olivier Le Meur

Trajectory saliency detection using consistency-oriented latent codes from a recurrent auto-encoder

Léo Maczyta, Patrick Bouthemy*, and Olivier Le Meur †

December 17, 2020

Abstract

In this paper, we are concerned with the detection of progressive dynamic saliency from video sequences. More precisely, we are interested in saliency related to motion and likely to appear progressively over time. It can be relevant to trigger alarms, to dedicate additional processing or to detect specific events. Trajectories represent the best way to support progressive dynamic saliency detection. Accordingly, we will talk about trajectory saliency. A trajectory will be qualified as salient if it deviates from normal trajectories that share a common motion pattern related to a given context. First, we need a compact while discriminative representation of trajectories. We adopt a (nearly) unsupervised learning-based approach. The latent code estimated by a recurrent auto-encoder provides the desired representation. In addition, we enforce consistency for normal (similar) trajectories through the auto-encoder loss function. The distance of the trajectory code to a prototype code accounting for normality is the means to detect salient trajectories. We validate our trajectory saliency detection method on synthetic and real trajectory datasets, and highlight the contributions of its different components. We show that our method outperforms existing methods on several scenarios drawn from the publicly available dataset of pedestrian trajectories acquired in a railway station [ARF14].

1 Introduction

Saliency estimation in images is a common task. It may operate as a pre-attention mechanism or be a goal in itself. Its output usually consists of saliency maps providing the probability of each pixel to be salient. Actually, saliency estimation is close to visual attention, object detection, or even image segmentation.

*L. Maczyta and P. Bouthemy are with Inria, Centre Rennes - Bretagne Atlantique, Campus de Beaulieu, Rennes, 35042, France.

†O. Le Meur is with Univ Rennes, CNRS, IRISA Rennes, Campus de Beaulieu, Rennes, 35042, France.

There is not really a single formal definition of image saliency. It is something that stands out from its surroundings in the image and attracts attention. In static images, this generally refers to one (or a few) prominent object(s) in the foreground as adopted in most challenges and benchmark datasets. It becomes a prominent moving object in the foreground when it comes to videos, and one talks about dynamic saliency.

In contrast, we will only take into account *motion* to define dynamic saliency, that is, local motion that deviates from the main motion of its surrounding. It may seem more specific, since it does not exhibit any notion of object. However, it can also be considered more general. Indeed, it involves all the cases where the only differentiating factor is movement, whatever the appearance is. A typical example is an individual walking against a crowd. It can occur in many diverse applications, such as monitoring of urban and road traffic [RB19, BFM19], safety of public areas prone to dense crowd [PRBB17], study of cell dynamics in bio-imaging [RDJ+17], or identification of adverse weather conditions in meteorological image sequences [PBMR00]. Motion saliency detection can be useful to trigger alarms, to focus additional processing or to allow for the detection of a specific event.

Here, we are going one step further, since we are interested in detecting *progressive* dynamic saliency. In other words, we are talking about saliency that may not be visible instantaneously, but rather progressively appears over time. Trajectories are the most natural features to support progressive dynamic saliency detection. Accordingly, we will talk about *trajectory saliency*. Indeed, trajectory-based anomaly detection emerges as a growing need in many applications, possibly under different names [RB19, MMP+15, FXZW18].

Two main issues then arise, namely, which representation for the trajectories and which method for trajectory saliency detection. We aim to design a method as general and efficient as possible. To do this, learning-based approached appear nowadays superior to handcrafted representations. If enough data are available, we can learn the trajectory representation with neural networks. Besides, we will design a decision algorithm to solve the second issue.

Our main contributions can be summarised as follows: 1) Definition of an original trajectory-based framework for progressive dynamic saliency detection in videos; 2) Learning of latent codes for trajectory representation based on a recurrent autoencoder and enforcing consistency for similar trajectories; 3) Thorough and comparative evaluation on a dataset of real pedestrian trajectories.

The rest of the paper is organised as follows. Section 2 is concerned with related work. Section 3 presents our latent trajectory representation based on a LSTM auto-encoder. Section 4 deals with the detection of trajectory saliency. In Section 5, we present the two trajectories datasets, a synthetic one and a real one. In addition, we design the three evaluation settings issued from the real one. In Section 6, we report experimental results on synthetic and real trajectory datasets, including an objective comparative evaluation and additional investigations on the main stages of the method. Finally, Section 7 contains concluding comments.

2 Related work

Saliency estimation in videos consists in identifying moving objects that depart from their context. We can classify the existing approaches in two categories. The first category assumes that salient elements are characterised by distinctive appearance and/or motion [KSS+16, LS16, MRLG11, WSS15]. Recent methods leverage deep learning techniques, either by estimating video saliency end-to-end [WSS18], or by first extracting intermediate deep features [LS18]. A second category of video saliency methods aims to predict which areas of the image attract the gaze of human observers. The objective is then to leverage eye-tracking data in order to predict efficiently eye-fixation maps [CBPH16, LZLL14, QGH18].

Although significant progresses have been done during the last decade, there are still many issues to overcome. Benchmarks used for video saliency evaluation typically consider only one or a few moving objects in the foreground of the viewed scene [OMB14, PPTM+16]. For such configuration, dynamic saliency can be directly estimated from a couple of successive images and possibly from optical flow. However, for more complex settings, this approach may become less adequate. In particular, dynamic saliency may appear only progressively over time. For a denser configuration of independently moving objects, and longer periods of observation, a change of paradigm can lead to better insights. This can be achieved by focusing on trajectories instead of raw visual content.

Trajectories can be obtained through several means. They can be directly provided with sensors such as GPS [ETNI16], which have become ubiquitous and can generate large amount of data. For natural videos, a tracker is usually applied to a unique camera as in [RB18], but more complex settings can be considered. A network of cameras with a processing pipeline including tracking and re-identification can provide trajectories of people over a large area [ARF14]. In specific contexts such as bio-imaging, trackers taking into account the expected properties of image content can be developed [CBO13].

When dealing with trajectories, extracting a compact representation is useful for many tasks. A trajectory auto-encoder network can be an efficient and general way to obtain such a representation without making any prior assumption [CHZC18, CRLG+18, SDZ+16, YZZ+17]. In [YZZ+17], this representation is obtained with a recurrent auto-encoder and is exploited for trajectory clustering, leading to the DRL method. In [SDZ+16], the representation is leveraged for crowd scene understanding. In [CRLG+18], the authors use a latent representation in a reinforcement-learning framework. The authors of [CHZC18] take into account spatial constraint of the environment such as walls for the estimation of the trajectory representation, thanks to the optimisation of a spatially-aware objective function.

With trajectory data, estimating dynamic saliency can be achieved by searching for anomalous patterns. In [RB18], the authors consider road user trajectories. They design the DAE method that reconstructs trajectories with a convolutional auto-encoder. Then, they assume that poorly reconstructed trajectories are necessarily anomalous or salient, since the latter are likely to be

rare or even missing in the training data. In an extended work presenting the ALREC method [RB19], the same authors adopt a Generative Adversarial Network (GAN) to classify trajectories as being normal or salient. They train the discriminator to distinguish normal and abnormal reconstruction errors. While the threshold between normality and saliency is automatically set with the GAN, the reconstruction error remains the core principle of the method.

The trajectory representation may not be limited to moving objects, as it can be relevant for time series data in general. In [FXZW18], the authors consider building energy data over time. They follow similar principles as for other types of trajectories, and base their anomaly detection method on the reconstruction error provided by an auto-encoder.

This type of paradigm suffers from several limitations. First, it assumes that the learned models will generalise badly to unseen data. In addition, saliency is handled in an absolute way. An element is stated as salient or not in itself. However, the same element may be salient or not depending on its context. This central issue should be explicitly addressed and we will introduce an original framework to properly deal with it.

Recently, significant progress has been made in reducing the supervision burden when training. An unsupervised stage can provide a strong pre-trained network for fine-tuning. In [WSM+19], tracking is performed with an unsupervised framework. For the trajectory prediction task [AHP19], the GAN framework is a mean to train a model without requiring manual annotations. Taking inspiration from these works, we will similarly seek to reduce as much as possible the annotation cost to ensure a broad applicability.

3 Latent trajectory representation

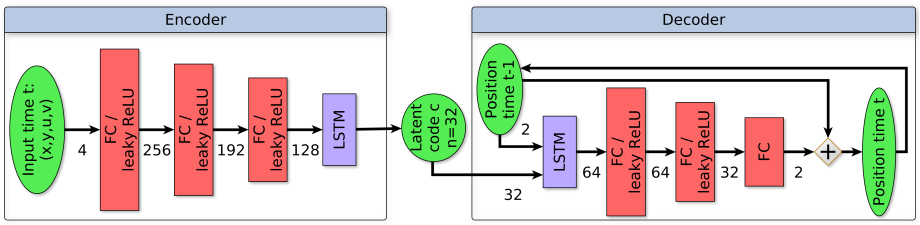


Figure 1: Recurrent auto-encoder network computing the latent representation of trajectories. Numbers indicate the input dimension for each layer.

Our objective is to detect salient trajectories within a set of trajectories, the vast majority of which are normal trajectories. In the following, a scenario $\mathcal{S}$ will consist of a set of normal trajectories for a given context and possibly a few salient trajectories with respect to the same context. Indeed, a trajectory is not salient in itself, but with respect to a given context. Salient trajectories are supposed to depart from the motion pattern shared by the normal trajectories.

We need a relevant representation of the trajectories. It has to be compact to form the basis of an efficient classification method, while descriptive enough to distinguish normal and salient trajectories. It is preferable to transform trajectories of any length into a constant-length representation, in order to enable adaptability to any scenario. It should be applicable to both 2D and 3D trajectories. In that vein, hand-crafted representations based on sophisticated mathematical developments were investigated in the past, as in [HBL08]. The latter provided representations invariant to a large class of geometrical transformations, including translation, rotation, scale, and able to achieve fine classification.

However, it is now more tractable to learn the targeted representation with the advent of neural networks, providing enough data are available for training [SDZ+16, YZZ+17]. An appropriate means is then the use of auto-encoders. The internal (latent) code in the auto-encoder will provide us with the desired representation. Besides, a trajectory exhibits a sequential nature inherent to the time dimension, making a recurrent network a suitable choice. Below, we will first describe the representation we have designed for trajectories based on these requirements. Then, in Section 4 we will explain how we leverage it to detect salient trajectories.

3.1 LSTM-based network for trajectory representation

Our approach is to compute the trajectory representation with a recurrent auto-encoder network. More specifically, we adopt a Long Short-Term Memory (LSTM) network. The objective of an auto-encoder is to reconstruct its input, by relying on an intermediate compact representation, acting as the code of the input [RLL18]. An auto-encoder has the advantage of being unsupervised, since no manual annotation is required for training.

This architecture is inspired from existing networks, such as the generator of [AHP19]. We similarly build a stack of fully connected layers to process the data at each time instant, but we dedicate the temporal processing to the recurrent LSTM units. This choice allows us to easily handle trajectories of variable length, in contrast to 1D temporal convolutions.

The architecture of the auto-encoder supplying the trajectory representation is summarised in Figure 1. The input of the network at time t is composed of the concatenation of the 2D position (x_t, y_t) and velocity (u_t, v_t). For the processing of the whole trajectory $\mathcal{T}_i$, it gives a total of 4τ features if τ is the length of the trajectory, i.e., the number of points it contains. In practice, we assimilate the velocity to the displacement. Although the displacement can be deduced from two successive positions, it still helps the network to properly estimate the trajectory representation at a negligible cost.

The 2D-space can be the image plane and the velocity vector given by the optical flow. If the 3D movements are planar, the 2D-space could be the ground plane of the 3D scene and trajectories (positions and velocities) are referred to this plane. An extension to full 3D trajectories would be straightforward.

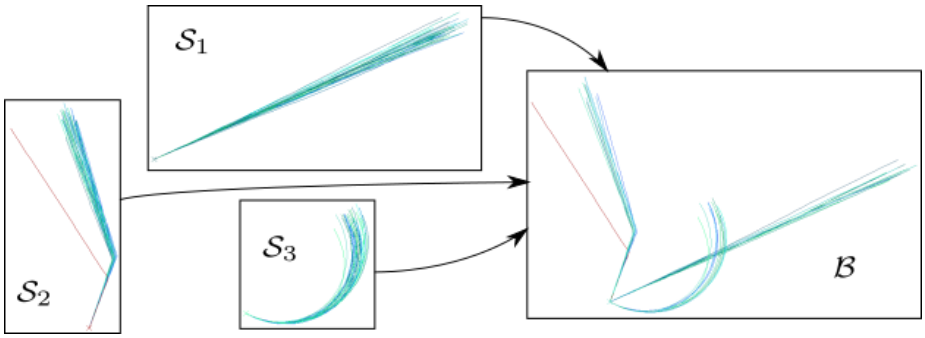


Figure 2: Trajectories of three scenarios $\mathcal{S}_1$, $\mathcal{S}_2$, $\mathcal{S}_3$ (from the STMS dataset described in Section 5.1) are displayed on the left. A batch $\mathcal{B}$ composed of trajectories sampled from the three scenarios is represented on the right.

The exact number of layers and their dimensions have been set during preliminary experiments. In particular, we set the dimension n of the code c to 32, as this value allows us to store enough information to represent the motion patterns, while being shorter that the typical trajectory representation of length 4τ. In the experiments, we consider trajectories involving a number of positions between 20 to 200.

To process one time step, the encoder activates a succession of three fully connected layers with output of dimension 256, 192 and 128 respectively, with the leaky ReLU activation function. Then, a LSTM layer with an output of dimension $n = 32$ is applied to get the representation of the whole trajectory.

The decoder reconstructs the trajectory from code $c \in \mathcal{R}^n$, obtained once the trajectory has been fully processed by the encoder. A LSTM-based decoder takes as input the code c concatenated with the position at time $t-1$, and produces a vector of 64 components. The decoder is only expected to expand the information compressed in the code. The predicted position for the current time instant is obtained through two fully connected layers, followed by the leaky ReLU non linearity, whose respective output dimensions are 64 and 32, and one final fully-connected layer of output dimension 2 representing the displacement. The position is given by the sum of the displacement and of the previous position. Indeed, preliminary experiments showed that predicting the displacement is more robust than predicting the position directly. Let us mention that, for the reconstruction of a trajectory, we assume that its length is available (in practice, we compute it from the list of all the positions). The number of parameters for the whole network is 127,970.

To train the auto-encoder, we adopt the reconstruction loss $\mathcal{L}_r$, defined as follows:

$$\mathcal{L}_r = \sum_{\mathcal{T}_i \in \mathcal{B}} \sum_{t=t_{init}}^{t_{final}} (x_t^i - \hat{x}_t^i)^2 + (y_t^i - \hat{y}_t^i)^2, \tag{1}$$

with t_{init} and t_{final} the initial and final time instants of the trajectory $\mathcal{T}_i$, (x_t^i, y_t^i) its position at time t, $(\hat{x}_t^i, \hat{y}_t^i)$ the predicted position and $\mathcal{B}$ a batch of trajectories. A batch can be composed of trajectories taken from several scenarios as illustrated in Figure 2. It is a standard practice to train auto-encoders with the reconstruction error as loss function [YZZ+17, RLL18]. Since the positions entirely define the trajectory, the loss $\mathcal{L}_r$ involves only positions.

Once trained, the auto-encoder is ready to provide a code representing the whole trajectory at test time. However, there is so far no guarantee that two similar trajectories will be represented with close codes. This is an important issue since we will exploit the codes to predict saliency. It motivates us to introduce a second loss term for code estimation.

3.2 Consistency constraint

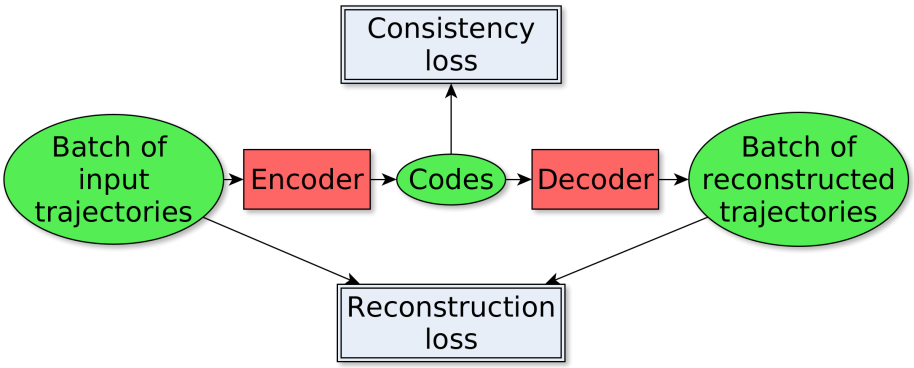


Figure 3: Training scheme of the recurrent auto-encoder with the two associated losses.

Our objective is to represent similar trajectories with close codes and dissimilar trajectories with distant codes to make saliency prediction easier. We somehow take inspiration from the paradigm of deep metric learning [SKP15]. We want to ensure the best possible consistency in the representation of normal trajectories, while drastically minimising, or even discarding, manual annotation. To this end, we complement our auto-encoder network with a consistency constraint, acting as a second loss term. Figure 3 summarises our overall training scheme.

Let us recall that we address the problem of detecting salient trajectories that are supposed to be rare within a group of normal trajectories. The normal trajectories of a given scenario are supposed to all follow a consistent motion pattern. We then define the following consistency loss term $\mathcal{L}_c$ applies to each scenario $\mathcal{S}_k$ present in the batch $\mathcal{B}$ and writes:

$$\mathcal{L}_c = \sum_{\mathcal{S}_k \triangleright \mathcal{B}} \sum_{\mathcal{T}_i \in \mathcal{S}_k \cap \mathcal{B}} ||c_i - \widetilde{c}_k||_2, \tag{2}$$

with c_i the code of the trajectory $\mathcal{T}_i$ estimated by the network, and $\widetilde{c}_k$ the median code for the trajectories of the batch issued from the scenario $\mathcal{S}_k$. The $\triangleright$ operator, designates here scenarios that contribute to the batch.

Applying the consistency loss to all the trajectories of the scenario means obviously that salient trajectories might be involved as well if there are any. On one hand, this setting may seem contradictory to our objective. However, let us point out again that salient trajectories are rare by definition. Therefore, they are unlikely to affect the value of the median code. In addition, the use of the L2 norm instead of the square in the consistency loss function (2) further limits the impact of the consistency constraint on the salient trajectories.

The full loss will add the quadratic loss term $\mathcal{L}_r$ defined in eq. (1) to the consistency constraint of eq. (2). One of the roles of $\mathcal{L}_r$ is to mitigate the effect of $\mathcal{L}_c$ on salient trajectories. Indeed, the combination of the two terms will express the trade-off between the reconstruction accuracy and the code distinctness for the salient trajectories. The large imbalance between salient and normal elements is not an issue, unlike for supervised classification. It even becomes an advantage for the correct training of our network by limiting the possible impact of salient trajectories on the consistency assumption. Besides, the addition of the consistency term still preserves an unsupervised training.

The final expression of the loss taking into account reconstruction and consistency terms is given by:

$$\mathcal{L} = \mathcal{L}_r + \beta \mathcal{L}_c, \tag{3}$$

where β is a weighting parameter to balance the impact of the two loss terms.

We have defined a method to estimate a latent code to represent trajectories. It is specifically designed to obtain close codes for similar trajectories and distant codes for potentially salient trajectories.

4 Trajectory saliency detection

Once trained, the recurrent auto-encoder provides us with codes of input trajectories. Now, we have to define an algorithm to detect salient trajectories against normal ones. Let us recall that normal trajectories share a common motion pattern with respect to a given context. More specifically, we consider a scenario $\mathcal{S}$ of the test dataset, containing normal and possibly salient trajectories, $\mathcal{S} = \{\mathcal{T}_i\}$. The trajectories are represented by their respective latent codes $\{c_i, c_i \in \mathcal{R}^n\}$. The scenario $\mathcal{S}$ will be further defined when reporting each experiment.

4.1 Distance between codes

First, for comparison purpose, we need to characterise the normal class by a generic code. It will be used to classify trajectories into normal and salient ones. Due to the possible presence of salient trajectories in $\mathcal{S}$, the generic code will be given by the median of codes over $\mathcal{S}$, denoted by $\widetilde{c}$. Since we deal with a n-component code, we compute the component-wise median code. Each

component of $\widetilde{c}$ is the median value of the corresponding components of the codes c_i over $\mathcal{S}$.

The classification of a trajectory $\mathcal{T}_i$, as normal or salient, will leverage the distance d_i between its code c_i and the median code $\widetilde{c}$:

$$d_i = ||c_i - \widetilde{c}||_2. \tag{4}$$

The distances related to normal trajectories are expected to be smaller than the ones related to salient trajectories. To make the comparison invariant to the distance range, the empirical average of the distances d_i, noted $\bar{d}$, and the standard deviation of the d_i, noted σ, are computed over scenario $\mathcal{S}$. We normalise the distance d_i to get the following descriptor q_i for each trajectory:

$$q_i = \frac{|d_i - \bar{d}|}{\sigma}. \tag{5}$$

With this definition, q_i belong to $\mathbb{R}^+$, with values ideally close to zero for normal trajectories. The status of each trajectory will be inferred from the descriptor q_i, as described below.

4.2 Inference of trajectory saliency from normalised distance

Descriptors q_i account for the possible deviation from the normal motion of the scenarios for each trajectory $\mathcal{T}_i$. Accordingly, a natural way to predict saliency is to assume that trajectories with a descriptor q_i greater than a given threshold λ are salient. We have investigated two ways of setting this threshold.

4.2.1 *p*-value method

The p-value scheme aims to statistically control the number of false detections and enables to automatically set λ. We assume that the q_i descriptors for normal trajectories follow a known probability distribution. We then need to estimate its parameters to fit the empirical distribution of the q_i descriptors of a representative set of normal trajectories. In order to identify candidate distributions, we looked at several histograms of q_i descriptors and observed that their distribution is skewed. Accordingly, the tested probability distributions were the Weibull distribution [Wei39], the Dagum distribution [Dag77] in the standardised form and the Dagum distribution in the general form. We selected the latter as explained in Appendix A.

Once the three parameters of the general Dagum distribution are estimated, we can fix the p-value and then get the threshold value λ. In an ideal case, the estimated distribution should correspond also to the test data. At this point, let us emphasise that we have no guarantee that this is the case. Indeed, the q_i descriptors depend on the presence of outliers, that is, salient trajectories, through the normalisation with $\bar{d}$ and σ the computed mean and standard deviation. In particular, the standard deviation σ may be noticeably influenced

by salient trajectories. Indeed, it tends to make the q_i smaller and it turns out that only fewer q_i are finally above the threshold λ given by the p-value. As a consequence, very few false detections occur but at the cost of misdetections of salient trajectories. We would have to decrease the p-value. This is confirmed by experiments reported in Appendix A. Then the p-value should be adapted to the considered scenario, which amounts to directly set the threshold λ. The p-value scheme is not effective in our case.

A way to overcome this problem would be to take another definition of the q_i descriptors that would not depend on the presence of saliency. A first idea would be to compute $\bar{d}$ and σ once and for all on normal trajectories of the validation set for each scenario. This way, the dependence on the presence of saliency at test time disappears. Another idea would be to make a robust estimation of $\bar{d}$ and σ on the scenario at test time. However, experiments conducted on the datasets described in Section 5, showed that results obtained for both alternatives were not as good as the ones obtained with the initial q_i.

4.2.2 Hard thresholding

In practice, we have adopted the following approach. To set λ for a given version of the trained network and a given dataset, we probed a range of values for λ. More specifically, we test candidate values over the interval [0,5] sampled every 0.05. Then, we select the one providing the best F-measure over the validation set associated to the given experiment. The choice $\lambda = 5$ as upper bound of the interval corresponds to 5 times the standard deviation with the definition of the q_i descriptors. It is enough to embrace very salient elements.

5 Datasets

In this section, we present the datasets used to train and evaluate our method. We have built a synthetic dataset of trajectories called STMS (Synthetic Trajectories for Motion Saliency estimation). We have also used the dataset made available by [ARF14], comprising pedestrian trajectories acquired in a railway station with a set of cameras over a long time period. We will denote it as RST dataset (RST standing for Railway Station Trajectories).

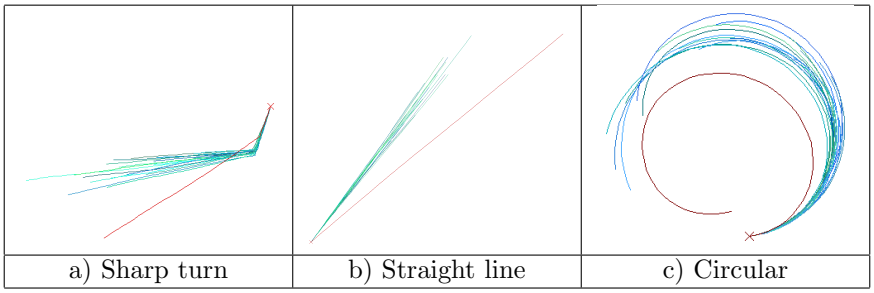
5.1 STMS dataset

Our synthetic trajectory dataset for motion saliency (STMS), contains three trajectory classes: straight lines, trajectories with sharp turns and circular trajectories. A noise of small magnitude is added to the successive trajectory positions to increase the variability of the trajectories.

Each scenario is composed of trajectories of the same kind (e.g., straight line) and with similar parameters (velocity, initial motion direction, etc.). The number of positions is comprised between 20 and 60 for each trajectory. The velocities vary within $[5, 20]$ depending on the scenarios. The initial direction

is randomly taken in $[0, 2\pi]$. For circular trajectories, the angular velocity is constant and lies in $[-0.10, 0.10]$ radians per time step. For trajectories with sharp turns, the rotation angle is chosen in $[-\frac{\pi}{2}, -\frac{\pi}{6}] \cup [\frac{\pi}{6}, \frac{\pi}{2}]$. The initial position of the trajectories is set to the origin (0,0). With the addition of the turn time instant for trajectories with sharp turns, these form the set of parameters required to define normality. For a given scenario, normal trajectories should be similar. As a consequence, only small variations are allowed for the trajectory parameters in a given scenario. Yet, to avoid too uniform trajectories, a random noise is added to each trajectory. The random addition $(x_n(t), y_n(t))$ at time t depends on $(x_n(t-1), y_n(t-1))$ to ensure that the trajectory remains smooth.

To come up with a difficult enough task, salient trajectories are of the same class as normal trajectories. The difference will lie in the parametrisation of the salient trajectories. The parameters are chosen so that the salient trajectory should be not too different from normal trajectories, but still distinct enough to prevent any visual ambiguity about its salient nature. An illustration is given in Figure 4.



a) Sharp turn | b) Straight line | c) Circular

Figure 4: Examples of synthetic trajectories for the three classes, starting from the same origin. Salient trajectories are drawn in red, normal trajectories in green to blue.

The trajectories for training are generated on the fly. We generate scenarios composed of 10 normal trajectories. An additional salient trajectory may be added with a probability of 0.5. Accordingly, we build batches made of 6 scenarios each, for a total of 60 to 66 trajectories.

The validation set and the test set are composed of 500 scenarios each. For these two sets, each scenario is composed of 20 normal trajectories. An additional salient trajectory is included with a probability of 0.5. We end up with a mean rate of salient trajectories of 2.5%. These two different configurations for the training and the test were chosen to challenge our method. A higher ratio of saliency is expected to make the training more difficult. On the other hand, rare salient trajectories are more difficult to find by chance during the test.

5.2 RST dataset

We now describe the RST dataset of real trajectories. We start with a general description of the dataset, before presenting the training procedure we applied to it.

5.2.1 Description of the dataset

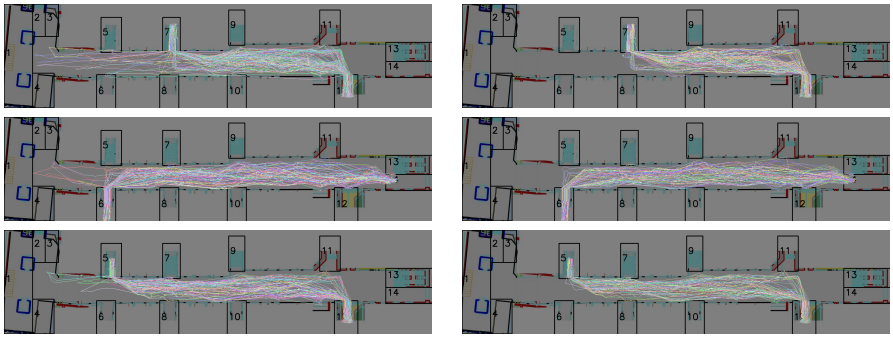


Figure 5: Illustration of the RST dataset (corridor PIW, numbers correspond to gates) and of the pre-processing step used to get homogeneous motion patterns. Display of trajectories before cleaning on the left, after cleaning on the right.

The second dataset is constituted of publicly available real pedestrian trajectories, computed with a tracking system installed in a train station [ARF14]. The tracking system takes videos recorded by a network of cameras. The videos were processed to extract trajectories, which are projected on the 2D ground plane of the train station. More specifically, the trajectories at our disposal were extracted from two underground corridors denoted in the following as PIW and PIE, with gates on their sides and their extremities (one corridor is displayed in Figure 5). The acquisition was performed during 13 days in February 2013, and resulted in a total of 115,245 trajectories. Training data are sampled from the 95,458 trajectories acquired during the 11 first days. We use trajectories sampled from the 19,787 trajectories acquired during the last two days for validation and test.

Compared to the STMS dataset, the trajectories of the RST dataset are longer, and the mean displacement between two points is larger. Very long trajectories are challenging for recurrent networks such as the one we employ. To overcome it, we subsampled RST trajectories by keeping one every five points (consequently multiplying the observed displacement magnitude by a factor five). We also divided the coordinates by 100 to get displacement magnitudes closer to the synthetic case. The full trajectories are recovered by re-identification from one camera to the next. This means that a given trajectory may be interrupted for some time, before being continued at a different point. To avoid these irregularities, we pre-processed the trajectories by splitting them

when unexpectedly large displacements are observed.

For a better stability of the training, we found helpful to make all the trajectories start at the same origin (x_0, y_0). It can be viewed as a translation invariance requirement with respect to the location of the trajectory in the 2D-space.

5.2.2 Training procedure

We pre-trained our network with the synthetic trajectories. For our training procedure, we then leverage the available data to get a relevant consistency constraint. Indeed, the RST dataset involves many different motion patterns. We then build scenarios composed of normal trajectories sharing the same entrance and exit. It may happen that between a given entrance and exit, people take different paths. They can go straight from the entrance to the exit, or they can make a detour to a given location in the train station as displayed in Figure 5. We computed the median length of trajectories associated to a given entrance/exit pair. Only trajectories with a length deviating from less than 10% of the median length were kept. We first built elementary scenarios of 8 trajectories by following the above-mentioned procedure. To get training batches of about 60 trajectories as in the synthetic dataset, we grouped 8 elementary scenarios from several entrance/exit pairs to construct a batch. It improves the diversity of data at each training iteration.

We consider three levels of consistency, respectively of $\beta = 10^5$, $\beta = 10^3$, and $\beta = 0$ (that is, no consistency). The network variants trained with this procedure and with these three consistency levels will be denoted respectively $V\beta_5$, $V\beta_3$ and $V\beta_0$ in the sequel. For this training procedure, 43,079 trajectories from the corridor PIW are included in the training set. Trajectories from the second corridor PIE were not included. In addition, we discarded additional trajectories, such as the ones starting in the middle of the corridor, or the ones following rare paths.

5.3 Evaluation settings for the RST dataset

The RS dataset was built for crowd analysis. There is no pre- defined saliency ground truth. Consequently, we have defined our own saliency experiments from the available data. Let us stress that our goal is to evaluate our method on real trajectory data. We have designed three evaluation settings from the RST dataset:

- RSTE-A is a large-scale general purpose setting,
- RSTE-B was required to be able to compare with other existing methods,
- RSTE-C is a setting focusing on a specific case.

5.3.1 Evaluation setting RSTE-A

The normal trajectories are those starting from the same given entrance and ending at the same given exit. To ensure that normal trajectories follow a similar motion pattern, we resort to the same cleaning step as the one used for the training procedure. By construction, salient trajectories have either a different entrance or a different exit, or both.

We built a test set comprising 2275 normal trajectories. They are divided into 45 scenarios, corresponding to 45 different entrance/exit pairs. In each scenario, salient trajectories are trajectories corresponding to a different entrance/exit pair. We have considered three cases related to three degrees of saliency. The highest the saliency degree, the easiest the trajectory saliency detection.

- In the easiest case, salient trajectories are randomly drawn from entrance/exit pairs different than the one of the normal trajectories. Furthermore, only entrance/exit pairs leading to distinct motion pattern are allowed. For instance, if normal trajectories go from gates 7 to 8, salient trajectories cannot go from gates 9 to 10 since the trajectories would be parallel and of same motion pattern once translated to the same origin (see Figure 5).

- In the medium case, salient trajectories share either the entrance or the exit with normal trajectories. The other gate of salient trajectories is chosen two positions after or before the other gate of normal trajectories on the same side of the corridor. For instance, if a normal trajectory goes from gates 12 to 9, a salient trajectory may go from gates 12 to 5, gate 5 being two positions after gate 9.

- The most difficult case is built similarly as the medium case. The difference is that the gate of a salient trajectory that differs from the gate of normal trajectories is only one position apart, that is, the next gate on the same side of the corridor. For instance, for a normal trajectory going from gates 12 to 9, a salient trajectory may go from gates 12 to 7.

In the following, these cases will be designated with their degree of saliency respectively qualified as high, middle and low.

In addition to the saliency degree, we will consider different ratios of salient versus normal trajectories. Ratios of 5%, 10% and 15% will be tested.

5.3.2 Evaluation setting RSTE-B

We built a second evaluation setting RSTE-B to be able to compare with three existing methods, DAE [RB18], ALREC [RB19] and DRL [YZZ+17]. Indeed, two of them need to be trained on normal data exclusively, and a training set is required for every scenario. The setting RSTE-B involves only two entrance/exit pairs, instead of 45 for the RSTE-A setting. The normal trajectories are checked by hand to make sure they all follow a consistent motion pattern. This requirement explains that only two types of entrance/exit pairs are included. Salient

trajectories are associated to different entrance/exit pairs taken at random as in the easiest case of RSTE-A.

200 normal trajectories are selected coming from gate 12 and going to gate 7. We name them as the G12-7 subset. 200 other trajectories are selected coming from gate 12 and going to gate 8. They are designated as the G12-8 subset. Samples are depicted in Figure 6. For each subset, half of the trajectories is used for training and the other half is used for test. The training subset is denoted as TrG. The training set TrG is only used for DRL and ALREC [RB19, RB18] in the comparative experiments.

Then, a group of 100 trajectories that are salient relatively to both the G12-7 and G12-8 subsets, is constructed. For each G12-7 and G12-8 test subsets, we have built 20 scenarios, each with 100 normal and 5 salient trajectories, leading to a saliency ratio of 5%.

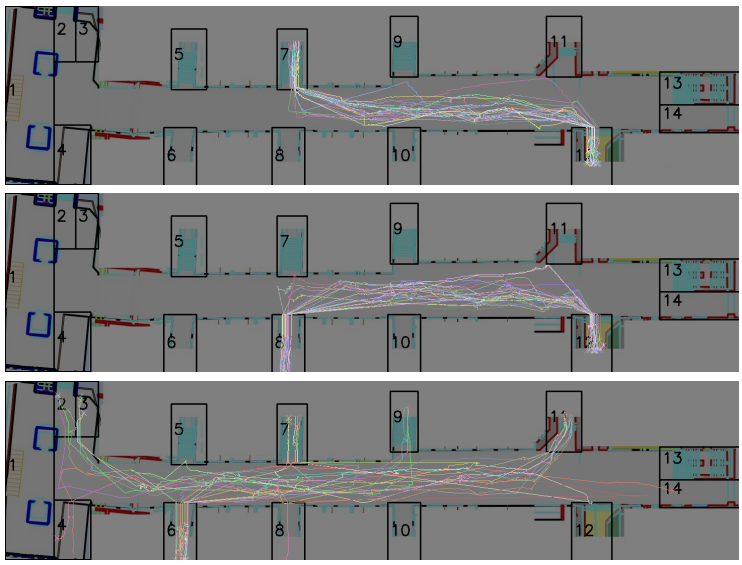


Figure 6: From top to bottom: 25 normal trajectories from the G12-7 subset, 25 normal trajectories from the G12-8 subset, and 25 salient trajectories with different entrance/exit pairs corresponding to the evaluation setting RSTE-B.

5.3.3 Evaluation setting RSTE-C

Finally, the evaluation setting RSTE-C focuses on a specific situation: the presence of different motion patterns for a given entrance/exit pair. More specifically, the trajectories that go straight from the entrance to the exit will correspond to the normal trajectories, and all the other ones will correspond to the salient trajectories (see Figure 7 for an illustration). To be sure that all trajectories are properly assigned to the normal or salient class in the ground truth, the labelling is performed by hand for this setting. RSTE-C will include

a total of 155 trajectories going from gate 12 to gate 7, and among them, 137 are normal and 18 are salient.

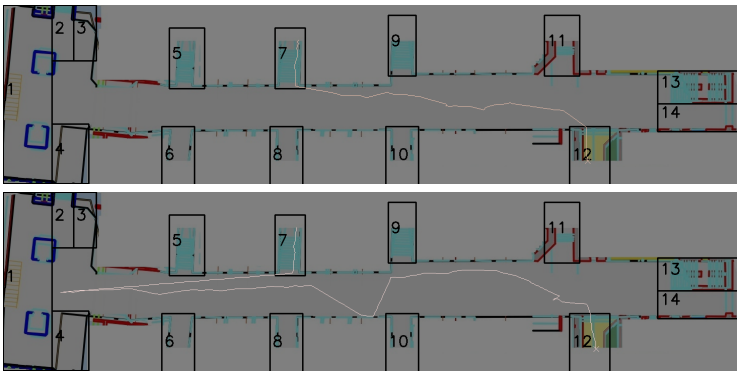


Figure 7: A normal trajectory (top) and a salient trajectory (bottom) for the RSTE-C evaluation setting.

6 Experimental results

In this section, we report the experiments we carried out on the synthetic STMS dataset and the train station RST dataset.

6.1 Implementation details

The network was implemented with the PyTorch framework [P+19]. For training, we used the Adam algorithm [KB14]. We set the learning rate by training the auto-encoder network on the STMS dataset. A learning rate of 10^{-4} provided a good compromise between speed and convergence and was then retained.

Parameter β of the loss function (3) defines the balance between the reconstruction and consistency constraints. It was set as follows. On one hand, a too low value of β would make the consistency constraint negligible. On the other hand, a too high value of β would prevent the auto-encoder from correctly reconstructing trajectories. In practice, we set $\beta = 10^5$ for the experiments on the STMS dataset. For the RST dataset, we considered values of $\beta = 10^5$, $\beta = 10^3$ and $\beta = 0$, as mentioned in Section 5.2.

Regarding the computation time, the forward pass takes 0.1 second and the backward pass 0.5 second, for a batch of 10 trajectories comprising 40 positions, on a machine with 4 CPU cores of 2.3 GHz. We let each network variant train three weeks.

6.2 Results on the synthetic dataset

We first evaluate our trajectory saliency method on the STMS synthetic dataset presented in Section 5.1. Our first goal is to assess the quality of the trajectory reconstruction by the auto-encoder. The quality score r will be defined as the average reconstruction error $\bar{e}$, divided by the average displacement $\bar{v}_n$ over the whole dataset:

$$r = \frac{\bar{e}}{\bar{v}_n}, \tag{6}$$

with

$$\bar{e} = \frac{1}{N} \sum_{\mathcal{T}_i} \frac{1}{t_f - t_i + 1} \sum_{t=t_{init}}^{t_{final}} \sqrt{(x_t^i - \hat{x}_t^i)^2 + (y_t^i - \hat{y}_t^i)^2}. \tag{7}$$

N is the number of trajectories in the dataset. r is defined this way to be dimensionless.

Our network is trained on the STMS dataset with the procedure described in Section 6.1. For this dataset, we consider only one variant with β set to 10^5. We got a score $r = 0.62$ for this dataset. We plot samples of reconstructed trajectories in Figure 8.

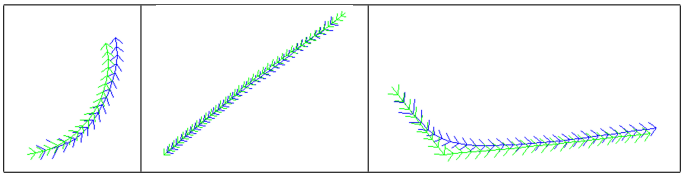


Figure 8: Three examples of synthetic trajectories (in green) and their reconstructed counterpart superimposed (in blue) for the three classes of trajectories. They nicely match, demonstrating an accurate reconstruction.

The second goal is to assess the trajectory saliency detection performance. We compute precision, recall and F-measure on the class of salient trajectories only, because they are the ones we are interested in. The rationale of this choice is to avoid a strong bias on performance due to overwhelming normal trajectories. The F-measure is the harmonic mean of precision and recall:

$$\text{F-measure} = 2 \cdot \frac{\text{precision} \cdot \text{recall}}{\text{precision} + \text{recall}} \tag{8}$$

Let us recall that the STMS dataset was constructed with 2.5% of salient trajectories. As shown in Table 1, we reached a F-measure equal to 0.89 for this dataset. It proves that we were able to correctly find even subtle trajectory saliency.

To highlight the contribution of the consistency constraint, we conducted an ablation study of our method. More specifically, the consistency constraint is removed, letting the auto-encoder reconstruction constraint drive the latent

code estimation. The reconstruction score is $r = 0.58$, and as expected it is slightly better than the one $r = 0.62$ obtained with the additional consistency constraint. However, the F-measure collapses to 0.26 as shown in Table 1. This clearly demonstrates the importance of the consistency constraint. Our intuition is that without the consistency constraint, the network employs all the available degrees of freedom to encode the trajectory pattern including any small variation. With the consistency constraint, the network is more focused on correctly representing the overall motion pattern of each trajectory, which allows us to better distinguish normal and salient trajectories.

	Precision	Recall	F-measure
Our full method	0.91	0.87	0.89
Without consistency	0.22	0.31	0.26

Table 1: Performance of our method evaluated on the STMS dataset, with and without the consistency constraint. We computed precision, recall and F-measure w.r.t. the salient trajectory class.

6.3 Comparative results on the RST dataset

We will now present results on the three settings RSTE-A, RSTE-B and RSTE-C involving real pedestrian trajectories of the train station RST dataset.

6.3.1 Results on the evaluation setting RSTE-A

Method variant	Saliency degree	Precision	Recall	F-measure
$V\beta_5$	Low	**0.65**	**0.73**	**0.69**
$V\beta_3$	Low	0.49	0.56	0.52
$V\beta_0$	Low	0.53	0.59	0.56
$V\beta_5$	Medium	**0.88**	**0.99**	**0.93**
$V\beta_3$	Medium	0.73	0.89	0.80
$V\beta_0$	Medium	0.82	0.89	0.85
$V\beta_5$	High	**1.0**	**1.0**	**1.0**
$V\beta_3$	High	0.97	0.99	0.98
$V\beta_0$	High	0.95	**1.0**	0.97

Table 2: Evaluation setting RSTE-A of our method variants for a saliency ratio of 5%. Threshold λ was set to 2 for all the experiments. For each saliency degree, the best performance is in bold. The highest the saliency degree, the easiest the case.

Results on the RSTE-A setting are collected in Table 2, for the three variants of our method, $V\beta_5$, $V\beta_3$ and $V\beta_0$, for different degrees of saliency. Let us recall that $V\beta_5$, $V\beta_3$ and $V\beta_0$ are the variants for which the parameter on the

consistency constraint is respectively 10^5, 10^3 and 0. For all these experiments, we set the threshold λ to a given value ($\lambda = 2$) once and for all. First of all, we observe as expected that the more salient trajectories depart from normal ones, the better the saliency estimation results are. Then, among $V\beta_5$, $V\beta_3$ and $V\beta_0$, the best method for a given degree of saliency is $V\beta_5$ by a clear margin, which confirms the interest of the consistency constraint.

We now report a second experiment, which consists in varying the ratio of saliency in the test dataset (defined as the ratio between the number of salient and normal trajectories). Results are collected in Table 3 for the best variant $V\beta_5$. We observe that the overall performance evaluated with the F-measure does not vary much when the ratio of saliency increases, which demonstrates that our method is applicable for different saliency regimes.

Method variant	Saliency degree	Saliency ratio	Precision	Recall	F-measure
$V\beta_5$	Low	5%	0.65	0.73	0.69
$V\beta_5$	Low	10%	0.85	0.61	0.71
$V\beta_5$	Low	15%	0.91	0.51	0.66
$V\beta_5$	Medium	5%	0.88	0.99	0.93
$V\beta_5$	Medium	10%	0.96	0.96	0.96
$V\beta_5$	Medium	15%	1.0	0.89	0.94
$V\beta_5$	High	5%	1.0	1.0	1.0
$V\beta_5$	High	10%	1.0	0.99	1.0
$V\beta_5$	High	15%	1.0	0.91	0.96

Table 3: Evaluation setting RSTE-A, with different saliency ratios for the best performing variant of our method. Threshold λ was set to 2 for all the experiments.

6.3.2 Results on the evaluation setting RSTE-B

We use the saliency evaluation setup presented in Section 5.3.2 to compare our method with several existing methods: DAE [RB18], ALREC [RB19] and DRL [YZZ+17]. For a fair comparison, the decision threshold involved in each method is set as best as possible.

Our method variants are trained as described in Section 5.2. The decision threshold λ is set as for the evaluation setting RSTE-A, over a validation set with low saliency degree and saliency ratio of 5%.

DAE and ALREC methods estimate saliency by training an auto-encoder on normal data only, and by considering that trajectories poorly reconstructed are salient. Then, for a fair comparison, we trained each of the two networks on the training subset of G12-7 and on the training subset of G12-8 respectively according to the experiment conducted. The decision threshold is set as proposed by the authors for DAE and ALREC. We then estimated trajectory saliency first on the G12-7 subset and then on the G12-8 test subset. The train-

ing subsets include 100 trajectories per scenario, which is more than the 16 to 31 trajectories per scenario that the authors considered in their own work. Due to the higher number of trajectories, we did not include data augmentation.

The DRL method represents trajectories with a constant-length code. This representation is obtained directly by training a network on the test trajectories. DRL was designed to produce input to clustering algorithms. To apply DRL to the trajectory saliency detection task, we considered two alternatives. The first option consists in using the k-means clustering algorithm with two classes. We then state that the class with fewer elements is the salient class. However, there is a risk that the large imbalance between the normal and salient classes makes this first option unstable. It motivates the following alternative. The second option exploits a similar decision procedure than ours but applied to the code produced by DRL. We took 101 values of λ regularly sampled in $[0, 5]$, and we selected the one that provided the best performance for DRL on the test set.

Results on the two test subsets are given in Table 4. We can observe that our method yields the best performance as expressed by the F-measure for both the G12-8 and G12-7 datasets. It even provides the best precision and recall scores in all cases.

RST subset	G12-7			G12-8		
Method	P	R	FM	P	R	FM
DAE [RB18]	0.83	0.32	0.46	0.81	0.32	0.46
ALREC [RB19]	0.72	0.43	0.54	0.71	0.45	0.55
DRL-KMean [YZZ+17]	0.05	0.54	0.10	0.08	0.53	0.14
DRL-C [YZZ+17]	0.72	0.24	0.36	0.54	0.52	0.53
Ours $V\beta_5$	**1.0**	0.98	**0.99**	**1.0**	0.89	0.94
Ours $V\beta_3$	**1.0**	0.98	**0.99**	**1.0**	0.98	**0.99**
Ours $V\beta_0$	**1.0**	**0.99**	**0.99**	**1.0**	**0.99**	**0.99**

Table 4: Comparative results for the RSTE-B evaluation setting, on the G12-8 and G12-7 test subsets. P, R and FM denote respectively precision, recall and F-measure computed w.r.t. the salient trajectory class. Best results are in bold.

In addition to this objective experimental comparison, we now discuss inherent differences between our method and methods such as DAE and ALREC. DAE and ALREC methods build a model to represent the normal data, and then, expect that this model will fail for anomalous elements. The failure level of the model is interpreted as the trajectory saliency indicator. A major limitation of this paradigm is its low generalisation capability. Indeed, for a new configuration, the normal and abnormal trajectories are likely to change, requiring to train again the network. For a deployment to many different scenarios, this can quickly become unpractical. These approaches are by design unable to handle relative saliency. Relative saliency refers to situations where an element is salient only because it is different from its local context. If the context changes the same element may be non salient.

In contrast to this family of methods, our approach is able to handle relative saliency, as illustrated by the experiments presented in Sections 6.2 and 6.3. Indeed, for these experiments, a trajectory appears as salient in a given context, while in a different one the same trajectory may be non salient. Furthermore, we do not rely on the assumption that a salient trajectory is not present in the training set. In fact, if salient trajectories are included in the training set, we expect they will be properly reconstructed and encoded, thus facilitating the decision. It enables us to train the network only once for each dataset, whatever the scenario considered.

6.3.3 Results on the evaluation setting RSTE-C

We report results on the RSTE-C described in Section 5.3.3. The decision threshold λ is set on a validation set with low saliency degree and a saliency ratio of 15% since more salient trajectories are expected. Results are collected in Table 5 for our method variants $V\beta_5$, $V\beta_3$ and $V\beta_0$.

The RSTE-C experiment is in fact a difficult one with very particular characteristics. The salient trajectories are visually different, but not that much. More specifically, they are structurally of the same vein since they share the same entrance and exit with normal trajectories. The only difference lies in a more wandering path. We observe that the $V\beta_5$ variant completely fails, which was not really expected. The intuitive explanation is that an unbalanced compromise between reconstruction and consistency constraints forces the representation of salient trajectories to conform to a more direct path between entrance and exit, which is precisely the pattern of normal trajectories. A too strong weighting of the consistency term of the loss function has thus adverse effects. In contrast, the $V\beta_3$ variant corresponding to a lighter weight on the consistency term, supplies good results, and in particular the best precision. The $V\beta_0$ variant is effective too. In short, this experiment shows that, beyond the overall benefit of the consistency constraint as demonstrated on the STMS dataset and on the two other general evaluation settings RSTE-A and RSTE-B, attention should be paid to its weighting in the loss function for specific situations.

Method variant	Precision	Recall	F-measure
$V\beta_5$	0.11	0.06	0.07
$V\beta_3$	**0.86**	0.67	0.75
$V\beta_0$	0.82	**0.78**	**0.80**

Table 5: Evaluation setting RSTE-C. Precision, recall and F-measure are computed w.r.t. the salient trajectory class. The best performance is in bold.

6.4 Additional investigations and visualisations

We now present additional investigations about the training, latent code estimation and decision stages. They will enable to better understand the behaviour of our method, and to provide more details on a few specific issues.

6.4.1 Initial state of the LSTMs

The network designed for trajectory representation includes LSTM layers in the encoder and the decoder. In a recurrent network, the value predicted for the next time step is estimated with the help of a hidden state that keeps memory of past information. However, there is no past information for the first time step. In practice, there are several ways to initialise the hidden LSTM state (see for instance [ZTG12]). The simplest solution consists in setting the initial state to a constant value, by default to 0. Slightly more sophisticated possibilities consist in setting it randomly or in learning it. If the initial value is randomly chosen, it is a way to introduce data augmentation, since the network faces more diverse configurations for the same training dataset. It may consequently help to improve performance. Learning the initial state can be done by including the initial state as a variable in the backpropagation algorithm. In preliminary experiments, we did not find any significant difference between these different initialisations in the network performance. We then decided to adopt a random initialisation, by drawing the initial state from a normal distribution.

With a random initialisation of the hidden state, the resulting parametrisation of the network is of course not strictly deterministic. Still, the network delivers consistent predictions when the same trajectory is given as input several times. This is illustrated in Figure 9 displaying several reconstructions of the same trajectory for several random initialisations. Apart from a hardly visible variation near the starting position of the trajectory (bottom-right), the reconstructions are almost identical.

6.4.2 Evolution of the model during training

Our method builds constrained latent codes to represent trajectories, and the codes are exploited to detect trajectory saliency. Our decision framework takes benefit from the location of codes in the latent space. It is somewhat inspired by deep metric learning.

Figure 10 displays the evolution of the trajectory reconstruction error and the F-measure related to trajectory saliency detection on the STMS validation set along the training iterations. Each iteration corresponds to a batch of 60 to 66 trajectories, depending on the random inclusion of salient trajectories. We observe that, after an initial phase with large improvements, the gain in performance becomes slower and slower. There is a noticeable improvement around iteration 200000, even if the final gain remains limited. On the other hand, regarding saliency detection, performance reaches a plateau in a few iterations, and even deteriorates a little for a while. Only after more than 100000 iter-

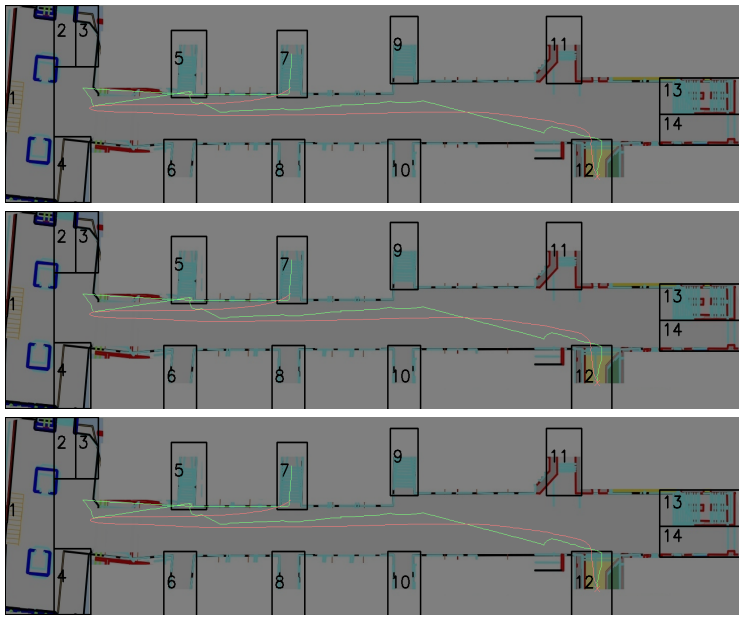


Figure 9: One trajectory reconstructed several times by a network with different initialisations of the hidden state of the LSTM. The reconstructed trajectory is drawn in red and the ground truth trajectory is drawn in green. The initial state of the LSTM is random, but the impact on the reconstructed trajectory is minimal.

ations, performance improves again relatively quickly, before reaching another plateau.

A similar behaviour is observed for deep metric learning [HBL17]. Indeed, codes are expected to evolve in the latent space during learning, in order to correctly represent different elements into separate clusters. However, while the clusters are not clearly distinguishable, the saliency detection algorithm cannot work well. As a consequence, it is beneficial to let the training go on, even if saliency detection performance apparently stagnates.

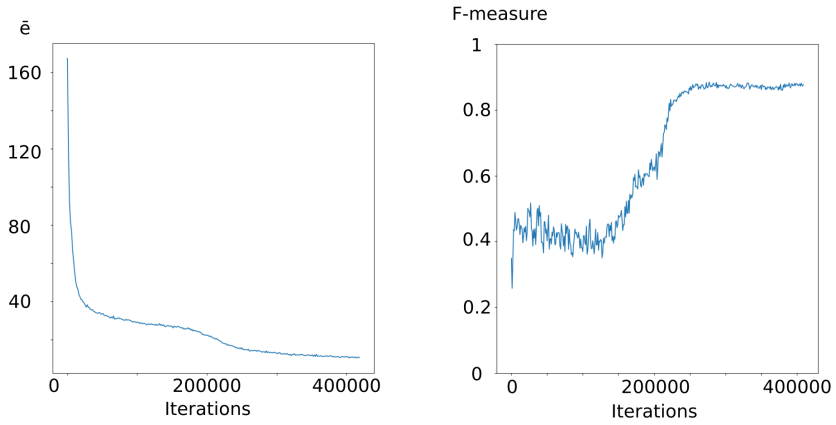


Figure 10: Evolution of the trajectory reconstruction error $\bar{e}$ (left) and of the F-measure related to trajectory saliency detection (right) on the validation set of the STMS dataset during training. Each iteration corresponds to a batch of 60 to 66 trajectories (depending on the random inclusion of salient trajectories).

6.4.3 Code distribution

The consistency constraint was designed to make codes representing similar elements closer. To better understand the effects of this constraint, we computed the empirical distribution of the learned codes with and without the consistency constraint on the STMS validation dataset. The components of each code are plotted as histograms in Table 6. The components corresponding to a training without consistency are denoted f_i, and those with consistency are denoted g_i. Their values lie in $[-1, 1]$.

Without the consistency constraint, all the code components vary largely. Two main patterns stand out. Either the code values are spread in [-1,1], or the code values are restricted to positive or negative ones. When the consistency constraint is applied, the variability is far lower. In almost all cases, the difference between the smallest and largest values is smaller than 0.5, and for several components, the predicted value is practically constant.

The consistency constraint has then a clear impact on the codes. Without it, the network tends to take profit of all the possible values to reconstruct the

trajectories with high precision. When enforcing the constraint, we end up with far smaller variations and only the main significant information is stored. Local variations inherent to each trajectory are more likely to be discarded.

6.4.4 Influence of choice of λ

To assess the influence of the threshold value on the method performance, the trajectory saliency detection was conducted on the STMS validation set with a regular sampling of λ (every 0.05) in the interval $[0, 5]$. The corresponding precision, recall and F-measures are plotted in Figure 11. It shows that in this case there is a plateau around $\lambda = 3$, which means that the performance is not sensitive to the decision threshold value. This observation is consistent with the results reported in Tables 2 and 3 on the real dataset RTS, where the same threshold value was used for all the experiments. The threshold value is of course dependent on the dataset and the nature of the trajectories.

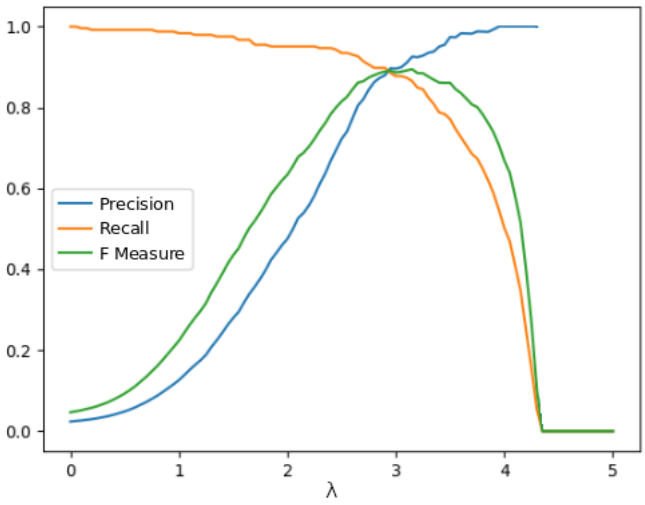


Figure 11: Precision, recall and F-measure computed w.r.t. the salient trajectory class. These metrics quantify the trajectory saliency detection performance for different values of λ on the STMS validation set. λ values are sampled every 0.05 within [0,5].

7 Conclusion

We have defined a novel framework for trajectory saliency detection based on a recurrent auto-encoder supplying a compact latent representation of trajectories. The auto-encoder training includes a consistency constraint in the loss

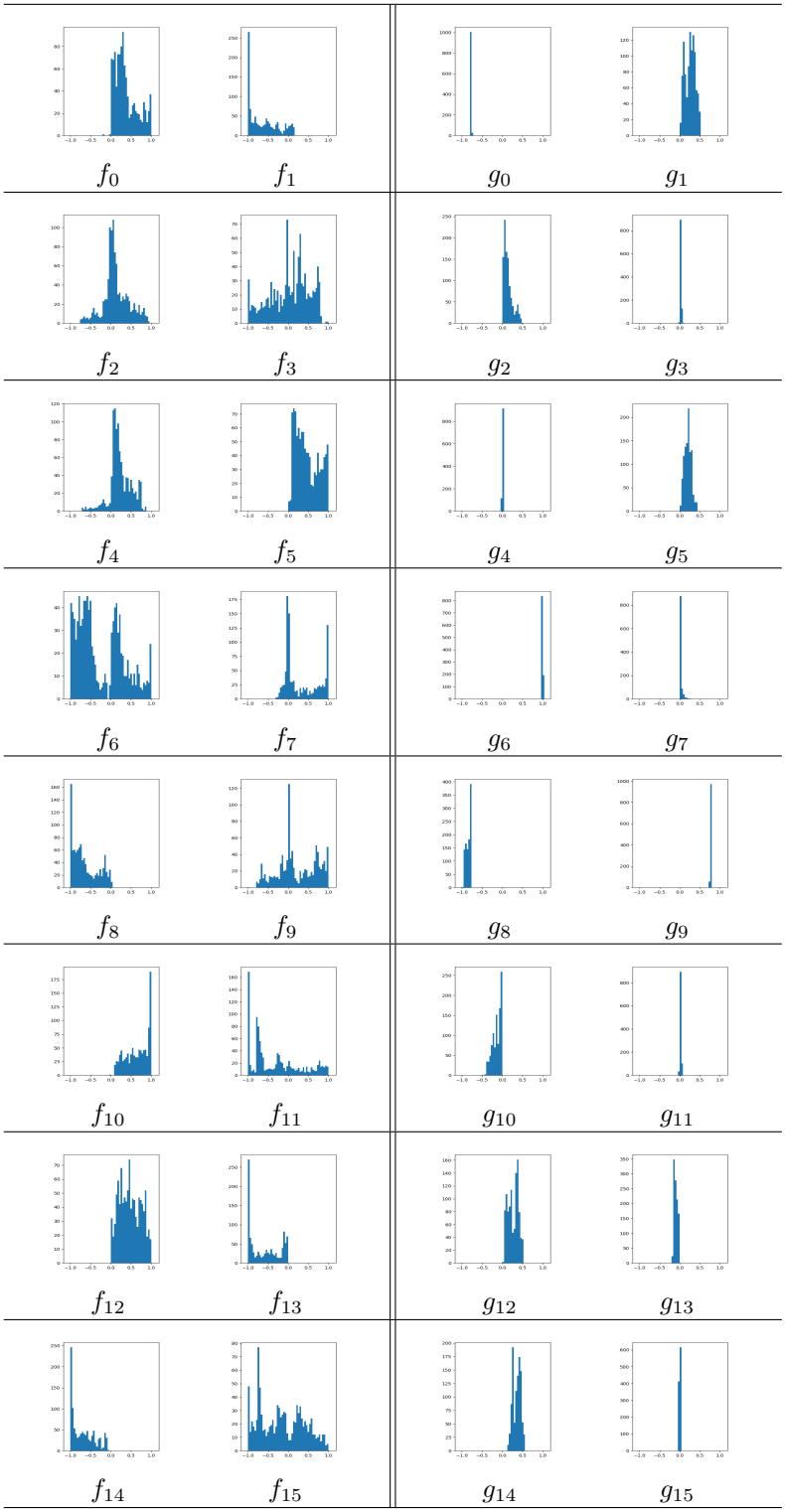


Table 6: Histograms of the first components of the trajectory codes, after training on STMS dataset. The remaining components exhibit a similar behaviour. The f_i components on the left correspond to a training without the consistency constraint, and the g_i on the right correspond to a training with the consistency constraint. The dispersion of the components is far more limited with the consistency constraint as expected.

function. The proposed method remains almost unsupervised, while being able to find saliency even when the difference with normality is subtle. We have explicitly formulated the trajectory saliency state and its associated decision algorithm. A trajectory is not salient in itself and so is not labelled only because a similar occurrence has never been seen in the training data. Trajectory saliency is defined with respect to a given context that specifies the normal trajectories. Such a paradigm allows for an easy generalisation to new configurations. We have experimentally validated our trajectory saliency detection method and its main components on synthetic and real trajectory datasets. We have demonstrated on several scenarios drawn from a publicly available dataset of pedestrian trajectories that our method outperforms existing methods.

A *p*-value method to set the saliency threshold

We report experiments on the p-value scheme to set the λ threshold for the trajectory saliency detection. We tested three probability distributions, the Weibull distribution [Wei39], the Dagum distribution [Dag77] in the standardised form and the Dagum distribution in the general form, to fit the empirical distribution of the q_i descriptors defined in equation 5.

Empirical and estimated probability distributions are plotted in Figure 12 for the STMS validation dataset. From this set, 10000 q_i corresponding to normal trajectories are computed. Distributions are fitted with the maximum likelihood method.

Visually, the best fit is obtained with the general form of the Dagum distribution. In addition, to get a quantitative measure of the fitting error, we use the criterion $\mathcal{F}$ defined as follows:

$$\mathcal{F} = \sum_b |\mathcal{G}(b) - \mathcal{H}(b)| \tag{9}$$

with b a bin, $\mathcal{G}(b)$ the area under the curve of the fitted probability law for the bin b, and $\mathcal{H}(b)$ the observed proportion of elements falling into the bin b. In practice, we use 101 bins regularly sampled between 0 and 5. The fitting errors are 0.05, 0.10 and 0.08 respectively for the Dagum distribution with the general form, the Dagum distribution with the standardised form and the Weibull distribution, confirming the visual evaluation. Consequently, we adopted the Dagum distribution with the general form.

Once the three parameters of the general Dagum distribution are estimated, we can fix the p-value and then get the threshold value λ. In an ideal case, the estimated distribution should correspond also to the test data. At this point, let us emphasise that we have no guarantee that this is the case. Indeed, the q_i descriptors depend on the presence of outliers, that is, salient trajectories, through the normalisation with $\bar{d}$ and σ, the computed mean and standard deviation. In particular, the standard deviation σ may be noticeably influenced by salient trajectories. Indeed, it tends to make the q_i smaller. As a consequence, fewer q_i will be above the threshold λ.

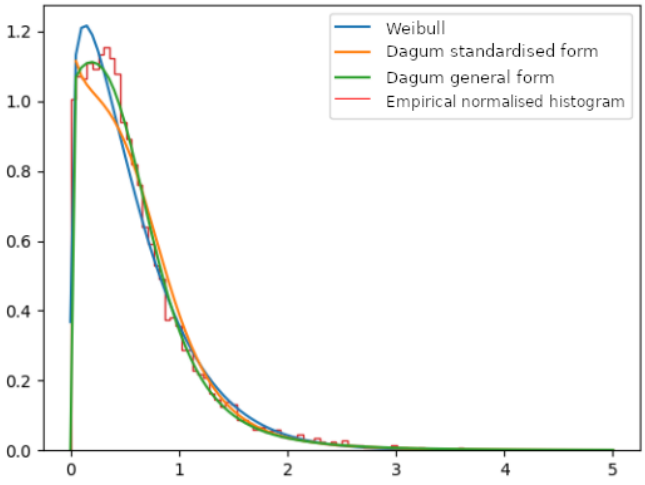


Figure 12: Plots of the empirical distribution of the q_i descriptors and the fitted distributions. They are computed from the 10000 normal trajectories of the STMS validation dataset. Each scenario is processed separately to compute the q_i descriptors.

The p-value approach converts the problem of setting the value of λ into the choice of the tolerated proportion of false positive in the prediction. In contrast, λ is not directly interpretable. The issue is that at test time, the p-value does not correspond actually to the expected proportion of false positives. Instead, it only provides an upper bound for this quantity. If the upper bound is too large, the p-value scheme will be no longer effective.

To test the p-value scheme, we take the evaluation setting RSTE-A with the lowest saliency degree. To fit the distribution, we need a subset of only normal trajectories. The TrG set meets this criterion. We used the $V\beta_5$ variant. Results are given in Table 7 for saliency ratios from 0.05 to 0.15. We give the chosen p-value and the False Positive Rate (FPR) that corresponds to the ratio between the number of false positives and the number of normal trajectories.

Results confirm that the empirical probability distribution of the q_i descriptors changes in presence of outliers. Indeed, the p-value and the FPR, which should normally have similar values (the FPR can be viewed as the empirical p-value), differ largely in practice.

Saliency ratio	p-value	λ	FPR	P	R	FM
5%	0.025	2.4	0.010	0.73	0.64	0.68
5%	0.05	1.9	0.019	0.61	0.73	0.67
10%	0.05	1.9	0.012	0.83	0.62	0.71
10%	0.10	1.5	0.035	0.67	0.76	0.71
15%	0.075	1.7	0.016	0.85	0.63	0.72
15%	0.15	1.3	0.050	0.67	0.73	0.70

Table 7: Evaluation setting RSTE-A, and λ set with the p-value scheme. P, R and FM denote respectively precision, recall and F-measure w.r.t. the salient trajectory class. FPR denotes the False Positive Rate.

Acknowledgement

This work was supported in part by the DGA and the Région Bretagne through co-funding of Léo Maczyta's PhD thesis.

References

[AHP19] Javad Amirian, Jean-Bernard Hayet, and Julien Pettré, *Social ways: Learning multi-modal distributions of pedestrian trajectories with gans*, Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), 2019, pp. 0–0.

[ARF14] A. Alahi, V. Ramanathan, and L. Fei-Fei, *Socially-aware large-scale crowd forecasting*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2014, pp. 2211–2218.

[BFM19] C. Barata, M. A. T. Figueiredo, and J. S. Marques, *Multiple motion fields for multiple types of agents*, IEEE International Conference on Image Processing (ICIP), 2019, pp. 1287–1291.

[CBO13] N. Chenouard, I. Bloch, and J. Olivo-Marin, *Multiple hypothesis tracking for cluttered biological image sequences*, IEEE Transactions on Pattern Analysis and Machine Intelligence **35** (2013), no. 11, 2736–3750.

[CBPH16] Souad Chaabouni, Jenny Benois-Pineau, and Ofer Hadar, *Prediction of visual saliency in video with deep CNNs*, SPIE Optical Engineering+ Applications, Applications of Digital Image Processing XXXIX, vol. 9971, August 2016.

[CHZC18] Ka Ho Chow, Anish Hiranandani, Yifeng Zhang, and S.-H. Gary Chan, *Representation learning of pedestrian trajectories using actor-critic sequence-to-sequence autoencoder*, CoRR **abs/1811.08069** (2018).

[CRLG+18] John Co-Reyes, YuXuan Liu, Abhishek Gupta, Benjamin Eysenbach, Pieter Abbeel, and Sergey Levine, *Self-consistent trajectory autoencoder: Hierarchical reinforcement learning with trajectory embeddings*, Proceedings of the 35th International Conference on Machine Learning, vol. 80, PMLR, 10–15 Jul 2018, pp. 1009–1018.

[Dag77] Camilo Dagum, *New model of personal income-distribution-specification and estimation*, Economie appliquée **30** (1977), no. 3, 413–437.

[ETNI16] Yuki Endo, Hiroyuki Toda, Kyosuke Nishida, and Jotaro Ikedo, *Classifying spatial trajectories using representation learning*, International Journal of Data Science and Analytics **2** (2016).

[FXZW18] Cheng Fan, Fu Xiao, Yang Zhao, and Jiayuan Wang, *Analytical investigation of autoencoder-based methods for unsupervised anomaly detection in building energy data*, Applied Energy **211** (2018), 1123 – 1135.

[HBL08] A. Hervieu, P. Bouthemy, and J. Le Cadre, *A statistical video content recognition method using invariant features on object trajectories*, IEEE Transactions on Circuits and Systems for Video Technology **18** (2008), no. 11, 1533–1543.

[HBL17] Alexander Hermans, Lucas Beyer, and Bastian Leibe, *In defense of the triplet loss for person re-identification*, CoRR **abs/1703.07737** (2017).

[KB14] Diederik Kingma and Jimmy Ba, *Adam: A method for stochastic optimization*, International Conference on Learning Representations (ICLR), 12 2014.

[KSS+16] A. H. Karimi, M. J. Shafiee, C. Scharfenberger, I. BenDaya, S. Haider, N. Talukdar, D. A. Clausi, and A. Wong, *Spatio-temporal saliency detection using abstracted fully-connected graphical models*, IEEE International Conference on Image Processing (ICIP), Sept 2016, pp. 694–698.

[LS16] Trung-Nghia Le and Akihiro Sugimoto, *Contrast based hierarchical spatial-temporal saliency for video*, Image and Video Technology, 2016, pp. 734–748.

[LS18] Trung-Nghia Le and Akihiro Sugimoto, *Video salient object detection using spatiotemporal deep features*, IEEE Transactions on Image Processing **27** (2018), no. 10, 5002–5015.

[LZLL14] Z. Liu, X. Zhang, S. Luo, and O. Le Meur, *Superpixel-based spatiotemporal saliency detection*, IEEE Transactions on Circuits and Systems for Video Technology **24** (2014), no. 9, 1522–1540.

[MMP+15] H. Mousavi, S. Mohammadi, A. Perina, R. Chellali, and V. Murino, *Analyzing tracklets for the detection of abnormal crowd behavior*, IEEE Winter Conference on Applications of Computer Vision, 2015, pp. 148–155.

[MRLG11] M. Mancas, N. Riche, J. Leroy, and B. Gosselin, *Abnormal motion selection in crowds using bottom-up saliency*, 18th IEEE International Conference on Image Processing (ICIP), 2011, pp. 229–232.

[OMB14] P. Ochs, J. Malik, and T. Brox, *Segmentation of moving objects by long term video analysis*, IEEE Transactions on Pattern Analysis and Machine Intelligence **36** (2014), no. 6, 1187–1200.

[P+19] Adam Paszke et al., *Pytorch: An imperative style, high-performance deep learning library*, Advances in Neural Information Processing Systems 32 (NIPS), Curran Associates, Inc., 2019, pp. 8024–8035.

[PBMR00] Christophe Papin, Patrick Bouthemy, Etienne Mémin, and Guy Rochard, *Tracking and characterization of highly deformable cloud structures*, Computer Vision — ECCV 2000 (Berlin, Heidelberg) (David Vernon, ed.), 2000, pp. 428–442.

[PPTM+16] F. Perazzi, J. Pont-Tuset, B. McWilliams, L. V. Gool, M. Gross, and A. Sorkine-Hornung, *A benchmark dataset and evaluation methodology for video object segmentation*, IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2016, pp. 724–732.

[PRBB17] Juan-Manuel Pérez-Rúa, Antoine Basset, and Patrick Bouthemy, *Detection and localization of anomalous motion in video sequences from local histograms of labeled affine flows*, Frontiers in ICT, Computer Image Analysis (2017).

[QGH18] Wenliang Qiu, Xinbo Gao, and Bing Han, *Eye fixation assisted video saliency detection via total variation-based pairwise interaction*, IEEE Transactions on Image Processing **PP** (2018), 1–1.

[RB18] P. R. Roy and G. Bilodeau, *Road user abnormal trajectory detection using a deep autoencoder*, Advances in Visual Computing, Springer International Publishing, 2018, pp. 748–757.

[RB19] ———, *Adversarially learned abnormal trajectory classifier*, 16th Conference on Computer and Robot Vision (CRV), May 2019, pp. 65–72.

[RDJ+17] P. Roudot, L. Ding, K. Jaqaman, C. Kervrann, and G. Danuser, *Piecewise-stationary motion modeling and iterative smoothing to track heterogeneous particle motions in dense environments*, IEEE Transactions on Image Processing **26** (2017), no. 11, 5395–5410.

[RLL18] Manassés Ribeiro, André Eugênio Lazzaretti, and Heitor Silvério Lopes, *A study of deep convolutional auto-encoders for anomaly detection in videos*, Pattern Recognition Letters **105** (2018), 13 – 22, Machine Learning and Applications in Artificial Intelligence.

[SDZ+16] Hang Su, Yinpeng Dong, Jun Zhu, Haibin Ling, and Bo Zhang, *Crowd scene understanding with coherent recurrent neural networks.*, IJCAI, vol. 1, 2016, p. 2.

[SKP15] Florian Schroff, Dmitry Kalenichenko, and James Philbin, *Facenet: A unified embedding for face recognition and clustering*, The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), June 2015.

[Wei39] W. Weibull, *A statistical theory of the strength of materials*, Proc. Roy. Swedish Inst. Eng. Res., 1939, pp. 1–45.

[WSM+19] N. Wang, Y. Song, C. Ma, W. Zhou, W. Liu, and H. Li, *Unsupervised deep tracking*, IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), June 2019, pp. 1308–1317.

[WSS15] W. Wang, J. Shen, and L. Shao, *Consistent video saliency using local gradient flow optimization and global refinement*, IEEE Transactions on Image Processing **24** (2015), no. 11, 4185–4196.

[WSS18] W. Wang, J. Shen, and L. Shao, *Video salient object detection via fully convolutional networks*, IEEE Transactions on Image Processing **27** (2018), no. 1, 38–49.

[YZZ+17] D. Yao, C. Zhang, Z. Zhu, J. Huang, and J. Bi, *Trajectory clustering via deep representation learning*, 2017 International Joint Conference on Neural Networks (IJCNN), May 2017, pp. 3880–3887.

[ZTG12] Hans-Georg Zimmermann, Christoph Tietz, and Ralph Grothmann, *Forecasting with recurrent neural networks: 12 tricks*, pp. 687–707, 2012.