On Accurate Computation of Trajectory Similarity via Single Image Super-Resolution

Hanlin Cao[1], Haina Tang[1*], Yulei Wu[2], Fei Wang,[3] Yongjun Xu[3]
[1]*Department of Artificial Intelligence*, *University of Chinese Academy of Science*, *Beijing, China*
[2]*Department of Computer Science*, *University of Exeter*, *Exeter, United Kingdom*
[3]*Institute of Computing Technology*, *Chinese Academy of Sciences*, *Beijing, China*
caohanlin18@mails.ucas.ac.cn, hntang@ucas.ac.cn, Y.L.Wu@exeter.ac.uk, {wangfei,xyj}@ict.ac.cn

2021 International Joint Conference on Neural Networks (IJCNN) | 978-1-6654-3900-8/21/$31.00 ©2021 IEEE | DOI: 10.1109/IJCNN52387.2021.9533802

***Abstract*—Measuring the similarity between trajectories is fundamental to many location-aware applications. However, the trajectory data collected in the real world suffer from the low-quality problem caused by non-uniform sampling rates and noises, which significantly affects the accuracy of similarity measurement. Traditional pairwise point-matching methods are susceptible to non-uniform sampling rates inherently, since they assume the consistent sampling rate. Although the recurrent neural network (RNN) based methods have addressed this problem by complementing trajectory data, they have the drawback of predicting positions conditioned on the historical data generated by the model itself, which could lead to accumulated bias during the inference stage. In this paper, we propose a novel generative model to address the important issue of low-quality trajectory data based on single image super-resolution. The trajectory similarity is thus computed using trajectory images instead of the trajectory sequential data. By utilizing the images to represent trajectories, we effectively overcome the issues encountered by existing methods. Extensive experimental results using real-world datasets demonstrate that our method outperforms existing methods in terms of accuracy.**

I. Introduction

Trajectory similarity computation has always been recognized as a fundamental task. An accurate similarity measurement is of significant importance to a wide range of trajectory based applications, such as moving together discovery [1], trajectory-user linking [2], trajectory clustering [3] and trajectory anomaly detection [4]. Unfortunately, due to the longstanding issue of the low-quality trajectory data, inaccurate computation of trajectory similarities hinders the performance and effectiveness of trajectory based applications. The problem of low-quality trajectory data mainly manifests in two aspects:

1. **Non-uniform sampling rates.** The sampling rates often vary across different trajectories due to the different device settings and communication conditions. Even for the same trajectory, the sampling rates may vary in different segments. This could bring about the mixed dense and sparse segments in trajectory data. Besides, the trajectories collected from the social networks like geo-tagged tweets and geo-tagged photo albums are inherently non-uniform [5].
2. **Noise.** The raw trajectory data always come with noises in reality, which indicates that the collected data have a certain degree of deviation from the real trajectory. Such noises could result from signal blockage, atmospheric conditions, and receiver design features/quality [6].

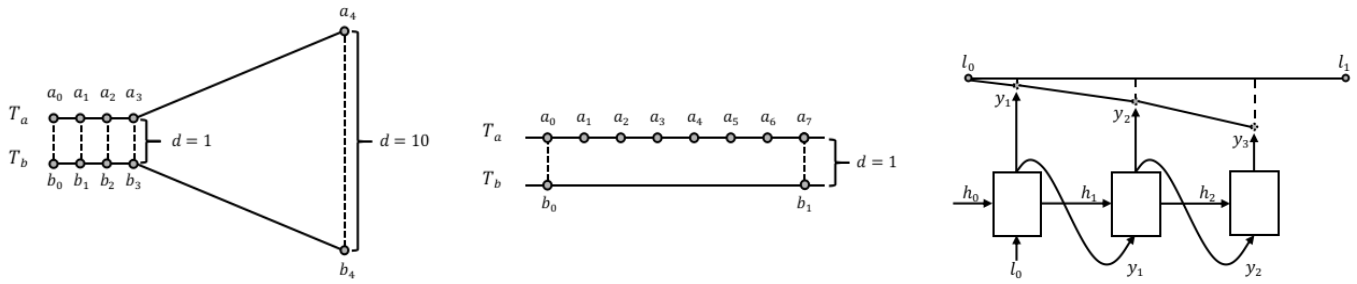
Existing trajectory similarity measurements can be categorized into *pairwise point-matching based* and *RNN based* methods. Pairwise point-matching based methods, including Dynamic Time Warping (DTW) [7], Longest Common Subsequences (LCSS) [8] and Edit Distance on Real sequence (EDR) [9], mainly use some kind of aggregation of the distances between trajectory sampling points to measure the similarity. Such kind of methods may encounter performance degradation when using low-quality trajectory data. Let us take EDR as an example to illustrate the problem of non-uniform sampling rates. EDR assigns a distance of 0 to points pair within the distance threshold ϵ, otherwise it assigns an edit distance of 1. Given $\epsilon = 1$, in Fig. 1a, EDR only yields a distance of 1 to (a_4, b_4) and the remaining points pairs are matched with the distance of 0. However, T_a and T_b are unlike in most parts. Therefore, the distance of 1 does not reflect the actual dissimilarity between the two trajectories. Similarly, as illustrated in Fig. 1b, only (a_0, b_0) and (a_7, b_1) can be matched, while the remaining unmatched points yield a distance of 6 by EDR. Although such two trajectories are similar, they are assumed dissimilar using EDR due to the different sampling rates.

Recently, RNN based methods were proposed to cope with the above problems by managing to predict the missing positions through maximizing the likelihood of each target position. However, despite their opportunity, RNN based methods still suffer from the so-called *exposure bias* in the inference stage [10]: the model predicts the next trajectory point conditioned on the previously predicted ones that may be never seen in the training data. As a result, the prediction error would be accumulated along with the generated trajectory sequence. Besides, the data point in a trajectory depends not only on the preceding (historical) points but also the succeeding (future) points within a segment of the trajectory, while RNN based methods only give inferences based on historical points. Fig. 1c illustrates an example, where RNN predicts y_3 based on the information contained in y_2 and h_2, while it cannot make use of the information from the near future, like l_1.

Moreover, both pairwise point-matching based and RNN based methods can only treat trajectory as sequential data, thus they heavily rely on the sequential order to measure the similarity. Specifically, only trajectories with similar sequential

* Corresponding author.

978-0-7381-3366-9/21/$31.00 ©2021 IEEE



(a) (b) (c)

Fig. 1: Illustration of the challenges in computing trajectory similarity. (a) Originally dissimilar trajectories are assumed similar by EDR; (b) Originally similar trajectories are assumed dissimilar by EDR; (c) The prediction process of RNN cannot take advantage of the information from the near future.

orders can be assumed similar. However, in many scenarios, the similarity primarily refers to the closeness of the physical positions between trajectories, while the sequential or the time information is not as important. For example, in trajectory-user linking, we need to link trajectories to users who generate them. In this case, a person's daily commuting trajectories, including going to work from home and going back home after work, should be regarded similar because they have the similar spatial shape. Existing methods fall short in directly distinguishing such similar trajectories with inverse orders.

In this paper, we propose a novel generative method for the trajectory similarity computation called **TrjSR** (Trajectory via single image Super Resolution), to tackle the important issue of low-quality trajectory data caused by non-uniform sampling rates and noises. The single image super-resolution (SISR) refers to the task of estimating a high-resolution (HR) image from a single low-resolution (LR) counterpart. By super-resolving, the image details can be reconstructed and thus the quality of the image is enhanced. In practice, the LR images are usually downsampled from the HR images with some blurring due to added noises, which is similar to the situation that the low-quality trajectories resulting from the non-uniform sampling rates (downsample) and noises (blurring). This observation enlightens us to introduce SISR to improve the quality of trajectory data. Specifically, we develop an SISR model to efficiently generate high-quality trajectory images from their low-quality counterparts, and a similarity-aware perceptual loss function is designed to improve the accuracy of similarity computation.

The main contributions of this paper are three-fold:

- We propose a generative method based on SISR to accurately compute trajectory similarity. To the best of our knowledge, this is the first work of introducing SISR techniques in trajectory similarity computation, making it more accurate than existing methods.
- We design a similarity-aware perceptual loss function to better assess the requirement for trajectory similarity computation. By doing so, not only the perceptually better super-resolution (SR) results can be obtained, but also the deep representation of trajectories can be used for measuring similarity.
- With extensive experimental results on two real-world datasets, we demonstrate that TrjSR achieves more accurate results against the non-uniform sampling rates and noises than the state-of-the-art trajectory similarity measurements.

II. Related Work

In this section, we briefly review the works related to trajectory similarity computation and SISR.

A. Trajectory similarity computation

Traditional similarity metrics are mainly based on pairwise point-matching. Dynamic Time Warping (DTW) [7] was proposed to measure trajectories with different lengths by recursively searching all possible points combination among trajectories with the minimal distance. Edit distance with Real Penalty (ERP) [11] is a trajectory similarity computation method based on edit distance, which utilized L_1-norm as the distance measure to seek the minimum number of edit operations required to change one trajectory to another. Longest common subsequence (LCSS) [8] matches two sequences by allowing them to stretch, without rearranging the sequence of the elements but allowing some elements to be unmatched. Edit Distance on Real sequence (EDR) [9] further against trajectory noises by assigning penalties to the gaps between the two matched sub-trajectories. Edit Distance with Projections (EDwP) [12] was proposed to mainly cope with the challenge of non-uniform sampling rates. Besides, researchers also adopt the metrics in mathematics to measure trajectory similarity, like Fréchet distance [13] and Hausdorff distance [14].

Deep learning models have recently drawn significant attention, and the ability to learn over large data endows them with more potential and vitality. [5], [15], [16] all used an RNN based encoder-decoder model to learn a vector representation of trajectory and compute trajectory similarity between the representation vectors. Besides, there was a branch of works focusing on accelerating trajectory similarity computation for different measures. For example, [17], [18] designed intricate models to reduce the time complexity. However, those methods

are all approximate algorithms and they exchange accuracy for efficiency. Our work differs from the above RNN based methods in that we utilize convolutional neural network (CNN) to compute trajectory similarity and we especially focus on improving the accuracy against low-quality trajectory data.

B. Single image super-resolution

There are mainly three types of SISR methods: *interpolation based*, *reconstruction based*, and *learning based methods*. Interpolation based methods, such as bilinear interpolation [19], bicubic interpolation [20] and Lanczos resampling [21], are extremely straightforward by estimating the HR pixels using their neighborhoods. However, such methods tend to generate overly smooth images. Reconstruction based SR methods [22] establish an observation model for the image generation process, and the restored HR image should reproduce the observed LR images after applying the inverse problem of the observation model. Nevertheless, the results may lack important high-frequency details when the desired magnification factor is large or the number of available input images is small [23]. Learning based methods aim to establish a mapping relationship between HR examples and their LR counterparts through training. Most of the recent learning based methods fall into the example based methods [24], where the prior knowledge is learned from LR and HR training pairs.

Recently, deep learning based SISR approaches are brought to the attention due to their dramatic superiority to reconstruction based and other shallow learning based methods. Dong et al. [25] first applied CNN to the SR problem and proposed SRCNN, which is the first deep learning based SR method. Since then, various CNN architectures [26]–[29] have been studied. Among which, both SRResNet [30] and EDSR [31] employed a deep residual network with skip-connection and achieved relatively better results. However, recent deep learning models often have massive parameters, which can cause time and memory issues during the training and inference stages. In our work, we focus on trajectory images instead of photo-realistic RGB images. Since trajectory images have much fewer texture details than the realistic RGB images, we develop a lighter SISR model to better cope with the time and memory issues.

III. Problem Formulation

A. Definitions

Definition 1: **Route** is the actual movement of an object in the spatial domain. It is a continuous function from time to space.

Yet, the route cannot be recorded continuously due to the sampling nature of location acquisition techniques.

Definition 2: **Trajectory** is a sequence of spatial-temporal positions sampled from the route and it can be formulated as $T = \{(x_0, y_0, t_0), \dots (x_k, y_k, t_k)\}$, where (x_i, y_i) is the geometric coordinates and t_i denotes the timestamp at which the moving object passes the location.

B. Problem description

Given a set of raw trajectories collected from a specific region, we aim to find trajectories sharing similar routes. In other words, we concern more about finding trajectories with shared spatial proximity. Thus we only consider two-dimensional coordinates as the trajectory data since they contain sufficient spatial patterns to compute similarity. However, raw trajectory data may meet low-quality problems caused by non-uniform sampling rates and noises, and therefore, it can affect the accuracy of the similarity computation. Our method is expected to be capable of alleviating such problems and achieve better results in accuracy.

IV. The Proposed Method

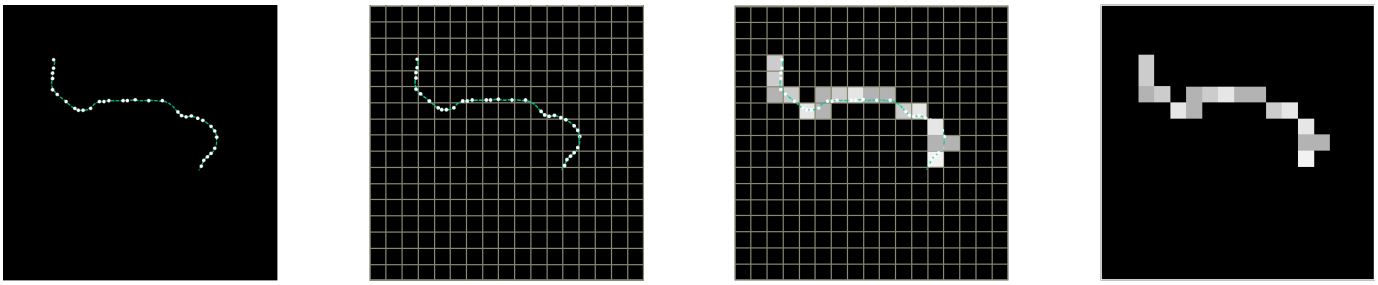
In this section, we first explain our motivations of using generative model. Then we describe the generation process of the trajectory image. Next, we introduce how we create example pairs for SISR. Finally, we present our SISR based method to compute trajectory similarity.

A. Motivations

The main reason that pairwise point-matching based methods cannot handle non-uniform sampling rates is that those methods measure similarity upon trajectories instead of routes. In principle, the route is the better description of objects' movement since they characterize the entire spatial information. However, the route only exists theoretically and not available in practice. Therefore, an intuitive way to tackle non-uniform sampling rates is to infer the underlying route from the raw trajectory data. To this end, we take advantage of the generative model to generate high-quality trajectory images from their low-quality counterparts, where the low-quality images represent the non-uniformly sampled trajectories. Those sparse or missing trajectory segments will be complemented through the generation process.

B. Trajectory image generation

Given the region where trajectories are collected, we first select a square area that contains sufficient amount of trajectories and then scale this area into a fixed-size image. For example, we can crop a square area belonging to the main city, where most trajectories happen. Then for a trajectory collected in this area, we map its coordinates to the pixels of the image, as depicted in Fig. 2a. However, only with those discrete pixels may result in a sparse trajectory image. To alleviate the sparsity, we partition the image into grids of equal sizes, as depicted in Fig. 2b, and instead we map trajectory coordinates to the grids. Then, in Fig. 2c, we assign value to pixels in each grid according to the number of trajectory points fallen on the gird: the more points imply the longer duration that the object stays in this grid (a small area in reality), and the higher value is assigned to pixels in this grid. Thus, different pixel values can be used to capture the temporal property of the trajectory. Finally, we transform the raw trajectory data into a single-channel grayscale image depicted in Fig. 2d.



(a) Map coordinates onto region images.

(b) Partition the images into grids.

(c) Assign value to pixels in grid.

(d) The grayscale trajectory images.

Fig. 2: The process of generating trajectory image.

The motivation of utilizing the trajectory images to represent raw trajectories is based on the following three observations:

1) Trajectory sequence does not explicitly reflect the spatial pattern. For example, spatially close positions may not be sequentially adjacent to each other in the trajectory data.
2) Existing methods compute similarity directly on trajectory sequence, thus they fail to identify trajectories sampled from similar routes but with different sequential orders.
3) Applying RNN to trajectory sequences suffers from the problem of *exposure bias* [10], and the prediction process cannot take references from the information of the near future.

Those observations motivate us to use image representation to tackle these problems. First, the image is naturally spatial-aware, both spatially close and temporally close positions can be depicted as neighboring pixels in the image. Second, trajectories will be represented as similar images if they are sampled from the similar routes. Thus the impact of sequential order is excluded from the similarity computation. Third, by using image representation, we can take advantage of CNN to address the problems caused by RNN. For each convolution operation, the convolutional kernel only actually takes effect around those image patches with non-zero pixels, which avoid predicting non-zero pixel values that are far from the current position. In addition, the receptive field covered by convolutional kernels contains both former and latter parts of the current trajectory segment, which means the prediction is not only based on the preceding data but also takes the succeeding data into consideration.

C. Creating example pairs

Recall that our method attempts to complement the sparse or missing trajectory segments by generating a high-quality trajectory image from its low-quality counterpart. To this end, it is essential to create reliable HR and LR image pairs for training. In theory, the trajectory image generated from the underlying route R should be worthy of the HR image since it illustrates the most detailed movement of the moving object. Yet, underlying route is not available in practice, therefore we adopt the relatively high sampling rate trajectory T_{high} and the relatively low sampling rate trajectory T_{low} to create HR and LR image pairs, where HR image represents the high-quality trajectory and LR image represents the low-quality trajectory

(a) LR trajectory image

(b) HR trajectory image

Fig. 3: An example of an LR and HR image pair.

Specifically, given a raw trajectory sequence T^k_{raw}, where k denotes the number of the sampling points this trajectory includes, we assume $T^k_{high} = T^k_{raw}$ as the relatively high sampling rate trajectory. Then we downsample the T^k_{high} by randomly dropping trajectory points to obtain the relatively low sampling rate trajectory $T^{k'}_{low}$, where $k' \leq k$. Both T^k_{high} and $T^{k'}_{low}$ are sampled from the same underlying route R, with only one difference: T^k_{high} is relatively closer to R since it contains more sampling points.

Furthermore, to create low-quality trajectory affected by noises, we further add Gaussian noises to coordinates belonging to $T^{k'}_{low}$:

$$x = x + 100 \cdot d_x, \quad d_x \sim \mathcal{N}(0,1)$$
$$y = y + 100 \cdot d_y, \quad d_y \sim \mathcal{N}(0,1)$$

Here, we set the radius of 100 (meters) because in trajectory images, each grid has the size around 100 meters.

We first transform $T^{k'}_{low}$ into LR image $I^{LR}_{W \cdot H}$, where W and H are the width and height of the image. To get the HR image, we take $2\times$ as upscaling factor r to transform T^k_{high} into $I^{HR}_{rW \cdot rH}$. For brevity, we omit the subscript of the image size in the remaining contents. As Fig. 3 shows, the LR trajectory image depicts a low-quality trajectory suffering from non-uniform sampling rates and noises.

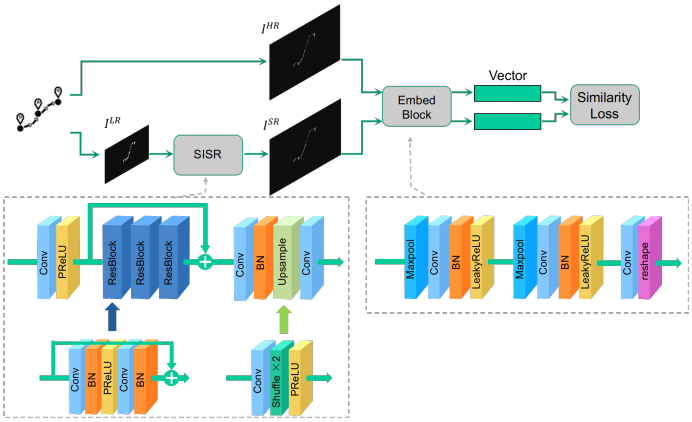


Fig. 4: The architecture of TrjSR.

D. The TrjSR Architecture

The architecture is shown in Fig. 4, which can be mainly divided into three parts: the SISR, the image embedding and the loss function.

1) SISR: Note that most existing SISR models were designed to reconstruct photo-realistic RGB images and they are often with heavy parameters. While trajectory images in this paper are single-channel grayscale images without much realistic texture details, it would be computational expensive to apply these models to our case. To implement SISR, we design a lighter but efficient model based on SRResNet to better cope with the time and memory issues. The details of the our SISR model are depicted in Fig. 4.

Recall that I^{LR} is the low-quality trajectory image, and I^{HR} is the corresponding high-quality counterpart. By maximizing the probability $\mathbb{P}(I^{HR}|I^{LR})$, the output of SISR, i.e. I^{SR}, is expected to be as close to I^{HR} as possible. Thus, we endow the model with the ability to complement the sparse and inaccurate segments from the low-quality trajectory data.

2) Image embedding: SISR improves the quality of I^{LR}, and we can further use I^{SR} to compute similarity. However, I^{SR}, is not adequate to be directly used for trajectory similarity computation, mainly for two reasons. First, I^{SR} contains thousands of pixels (in our case, the pixel amount is 256×256), which is time-consuming to compare images pixel-wise. Second, as Fig. 3 depicts, most pixels of the trajectory images have zero value, and it is redundant with so many zero-value pixels. Therefore, to efficiently compute similarity among trajectories, we further embed the I^{SR} and I^{HR} into low dimensional vector space as illustrated in Fig. 4. After max pooling and strided convolution layers are used to reduce the image resolution, we stretch the final image (32×32) into a one-dimensional vector. As thus, not only a concise representation of trajectory can be obtained, but also the accuracy and efficiency of similarity computation is much improved, which is demonstrated in Section V-E.

3) Loss function: To train the embed block and the SISR model jointly, we design two loss functions to separately evaluate the trajectory similarity and the performance of SISR.

a) Similarity loss: We define the trajectory similarity loss as the Euclidean distance between the embedding vectors:

$$l_{vec} = \frac{1}{|\mathbf{v}|}\|\mathbf{v}^{SR} - \mathbf{v}^{HR}\|_1$$

where $\mathbf{v}^{SR}$ and $\mathbf{v}^{HR}$ denote the embedding vectors of the super-resolved image I^{SR} and the target image I^{HR} respectively, and $|\mathbf{v}|$ denotes the dimension of $\mathbf{v}$.

The reason of using L1 distance instead of L2 distance here is that we expect the divergence at each dimension to be wide, while the L2 distance tends to flatten those divergences. For example, $[0.5, 0.5, 0.5, 0.5]$ and $[1, 1, 0, 0]$ have the same L1 distance value, while the L2 distance gives a smaller loss to the former one. In fact, we prefer the latter one because the big disagreements at the first two dimensions of $[1, 1, 0, 0]$ provide evidence of the greater ability to distinguish vectors. Our experimental result in Section V-F proves that using L1 distance is indeed better than L2 distance in practice.

Using the similarity loss for training our embed block encourages similar trajectory images are as close as possible after being embedded into the vector space.

Algorithm 1 Mini-batch training of TrjSR.

Input: Trajectory image pairs.
Output: The learned SISR model G_{θ_g} and embed block E_{θ_e}.

1: **for** number of training iterations **do**
2: Sample a minibatch of n image pairs: $\langle I^{LR}_{(i)}, I^{HR}_{(i)}\rangle_{i=1}^{n}$
3: Generate the super-resolved images: $I^{SR}_{(i)} = G_{\theta_g}(I^{LR}_{(i)})$
4: Embed the images into vectors:

$$\mathbf{v}^{SR}_{(i)},\ \mathbf{v}^{HR}_{(i)} = E_{\theta_e}(I^{SR}_{(i)}),\ E_{\theta_e}(I^{HR}_{(i)})$$

5: Update E_{θ_e} by descending its stochastic gradient:

$$\nabla_{\theta_e}\frac{1}{n}\sum_{i=1}^{n} l^{(i)}_{vec}$$

6: Update G_{θ_g} by descending its stochastic gradient:

$$\nabla_{\theta_g}\frac{1}{n}\sum_{i=1}^{n} l^{(i)}_{SR}$$

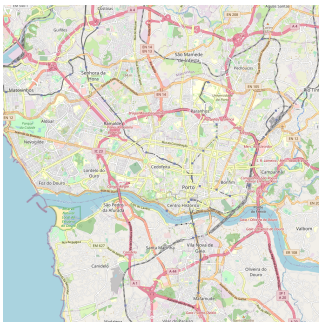
7: **end for**

b) Perceptual loss: The pixel-wise MSE loss is the most widely used optimization target for SISR:

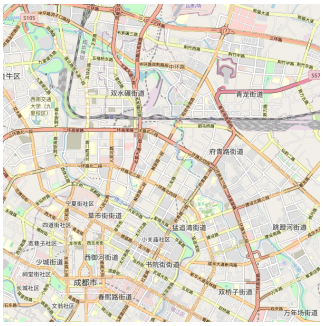
$$l_{MSE} = \frac{1}{r^2WH}\sum_{x=1}^{rW}\sum_{y=1}^{rH}\left(I^{HR}_{x,y} - G_{\theta_G}\left(I^{LR}\right)_{x,y}\right)^2$$

In our case, we not only desire I^{SR} is perceptually similar to I^{HR}, but also expect the I^{SR} can be embedded into a better vector representation for similarity computation. Thus, we design our similarity-aware perceptual loss function as weighted sum of both pixel-wise MSE loss l_{MSE} and similarity loss l_{vec}:

$$l_{SR} = \lambda \cdot l_{MSE} + \mu \cdot l_{vec}$$



(a) Porto



(b) Chengdu

Fig. 5: The selected area. (a) Porto with size of 15×15 km^2; (b) Chengdu with size of 10×10 km^2

.

where λ and μ denote the weights for the two components. The optimization procedure is formally presented in Algorithm 1.

V. Experimental Results and Analysis

We evaluate the performance of TrjSR on two real world taxi datasets. Three sets of experiments are performed to analyze the effectiveness of the proposed TrjSR in Section V-B, with parameter study in Section V-C. Examples of our recovered trajectory images are shown in Section V-D. Besides, we also study the impact of the image embedding and the distance measurement in Section V-E and Section V-F, respectively. Source code and implementation details are available online[1].

A. Experiments Settings

1) Dataset: We conduct our experiments on publicly available taxi datasets.

- The first dataset was collected by 442 taxis running in the city of Porto, in Portugal over 1 year. The sampling rate is around 15 seconds. We select a square area in the main city as shown in Fig. 5a, where 1.5M trajectories are collected in this area. The mean length of the trajectories is 87.
- The second dataset is collected in Chengdu, China and released by DiDi Company[2]. We choose its first 5 days' data and select an area as illustrated in Fig. 5b, which include nearly 1M trajectories. The sampling rate is approximately 3 seconds. The mean length of trajectories is 196.

To avoid those extremely short trajectories, we remove trajectories with length less than 60 location points for both datasets. We use a subset of 200K trajectories to create our training dataset, 10% of which are used for validation.

To create training image pairs as described in Section IV-C, we take [5] for reference. Specifically, for a raw trajectory, two types of data transformation methods are applied: downsample and distortion. First, we downsample the trajectory sequence by randomly dropping location points at four different probabilities $[0, 0.2, 0.4, 0.6]$. Then, we distort the above downsampled instances by randomly adding Gaussian noise to coordinates at four different probabilities $[0, 0.2, 0.4, 0.6]$. After the above operations, $4 \times 4 = 16$ trajectories reflecting varying levels of low-quality are created, and they are further transformed into LR images as Section IV-B described. The original raw trajectory is used to generate the HR image. As such, we obtain 16 example pairs $\langle I_{(i)}^{LR}, I^{HR} \rangle_{i=0}^{16}$ from one raw trajectory data.

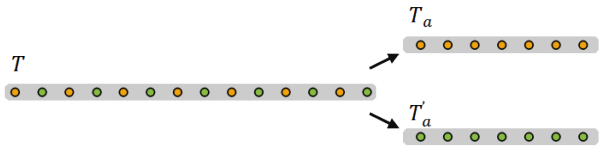


Fig. 6: Illustration of alternately sample trajectory.

2) Baseline methods: We compare the accuracy of TrjSR with LCSS [8], EDR [9], Fréchet distance [13], Hausdorff distance [14], EDwP [12] and t2vec [5]. We do not take DTW and ERP for comparison since both EDR and EDwP have shown superiority over them. t2vec is based on RNN which achieved competitive results in trajectory similarity computation.

3) Training details and parameter settings: Our implementation uses Pytorch on Ubuntu 18.04; training takes roughly 20 hours on a single GeForce RTX 2080Ti GPU. We train our model with Adam optimizer by setting $\beta_1 = 0.9$, $\beta_2 = 0.999$, and $\epsilon = 10^{-8}$, with the learning rate of 10^{-4} and the batch size of 32 for 90,000 iterations, giving approximately 4 epochs over the training data.

For training the SISR model, we use the single-channel grayscale image of size 128×128 as input LR image, with the corresponding HR image of size 256×256 as the target. The weights for similarity-aware perceptual loss function l_{SR} are set as $\lambda = 0.1$, $\mu = 1$. For training the embed block, the input SR/HR images are embedded into the vectors of 1024 dimensions. The matching threshold ϵ for LCSS and EDR is set following the original papers [8], [9].

B. Performance Evaluation

Recall that we initially attempt to infer the underlying route R from the raw trajectory T_{raw}^k via a generative model, and those trajectories sharing the similar underlying route are identified as similar. Since R is not available in practice, we alternatively take T_{raw}^k as the suboptimal choice to represent R because it is the closest trajectory to R in dataset. Thus, those low-quality trajectories transformed (i.e. downsample and distortion) from the same T_{raw}^k can be regarded as similar since they all reflect the same pseudo underlying route T_{raw}^k.

With the above idea in mind, we apply the similar procedure as mentioned in [5] by creating two datasets: D_Q and D_B. D_Q contains 1000 trajectories used as the query, and trajectories in D_B are served as database. The size of D_B is a parameter to be evaluated. For each trajectory $T \in D_Q$, we alternately

[1]https://github.com/C-Harlin/trjsr

[2]https://outreach.didichuxing.com/research/opendata/en

TABLE I: Mean rank of 1000 query trajectories versus different database size.

Porto					
database size	20k	40k	60k	80k	100k
LCSS	3.203	5.581	7.882	10.502	13.037
EDR	3.213	5.36	7.65	9.53	11.278
EDwP	1.678	2.275	2.859	3.409	3.965
Fréchet	2.569	4.205	5.692	7.239	8.794
Hausdorff	1.26	1.5	1.675	1.884	2.129
t2vec	1.292	1.415	1.579	1.753	1.972
TrjSR	**1.179**	**1.227**	**1.305**	**1.372**	**1.466**

Chengdu					
database size	20k	40k	60k	80k	100k
LCSS	11.134	20.842	30.479	40.303	49.887
EDR	1.131	1.254	1.375	1.508	1.637
EDwP	1.334	1.644	1.965	2.289	2.626
Fréchet	10.483	19.869	29.325	38.71	48.053
Hausdorff	1.689	2.36	3.075	3.804	4.523
t2vec	1.308	1.608	1.907	2.213	2.524
TrjSR	**1.097**	**1.195**	**1.288**	**1.383**	**1.478**

TABLE II: Mean rank of 1000 query trajectories versus different downsampling rates r_1.

Porto					
r_1	0.2	0.3	0.4	0.5	0.6
LCSS	42.278	61.526	116.576	169.02	284.751
EDR	5.921	23.257	47.398	254.862	693.718
EDwP	6.782	8.124	11.872	23.319	44.891
Fréchet	17.937	19.617	52.978	108.015	227.972
Hausdorff	8.786	15.202	36.542	143.171	259.849
t2vec	7.782	28.33	29.029	124.266	184.508
TrjSR	**3.156**	**4.839**	**10.518**	**19.828**	**43.636**

Chengdu					
r_1	0.2	0.3	0.4	0.5	0.6
LCSS	81.223	99.035	124.059	159.544	201.715
EDR	2.916	3.051	9.496	65.51	149.343
EDwP	2.795	3.149	3.606	4.56	12.427
Fréchet	70.94	58.478	51.541	65.321	97.646
Hausdorff	28.97	26.593	8.946	40.089	128.983
t2vec	3.271	3.142	5.051	5.579	13.362
TrjSR	**2.258**	**2.736**	**3.454**	**3.627**	**6.181**

take trajectory points to obtain two sub-trajectories T_a and T'_a as shown in Fig. 6. This procedure guarantees that T_a and T'_a are both similar and distinct as well. T_a and T'_a are further used to create two datasets $D_a = \{T_a\}$ and $D'_a = \{T'_a\}$. For each trajectory $T_a \in D_a$, there exists a corresponding similar trajectory T'_a in D'_a. We conduct the same procedure to trajectories in D_B to get D_b and D'_b. Then, for each trajectory $T_a \in D_a$, we retrieve the top-k most similar trajectories in $D'_a \cup D'_b$ (or $D'_a \cup D_b$) and calculate the rank of T'_a. According to our assumption, T'_a should be ranked at the top place, since T_a and T'_a are generated from the same trajectory. Besides, we can also perform data transformation (downsample and distortion) on D_a, D'_a and D'_b to evaluate the performance under different types of low-quality.

*1) **Accuracy on clean data**:* We first evaluate the performance on clean data, in which the methods are expected to find $T'_a \in D'_a \cup D'_b$ as the most similar trajectory for each $T_a \in D_a$. We increase the size of $D'_a \cup D'_b$ by varying the size of D_B. Euclidean distance is adopted to measure the similarity between trajectories.

Table I shows the results of the mean ranks of 1000 query trajectories versus the increased size of $D'_a \cup D'_b$ from 20K to 100K on Porto and Chengdu datasets. We can see that LCSS performs worst as it ignores the differences of the gap between trajectories, which leads to inaccuracy. Fréchet distance has similar results to LCSS. EDR achieves better results than LCSS because it assigns penalties to the unmatched point pairs which improves the accuracy, and it performs much better on Chengdu dataset. EDwP and Hausdorff distance perform relatively more stable and better than other pairwise point-matching methods on both datasets. Although t2vec demonstrates competitive results, our method outperforms all the other methods, which sheds light on the feasibility of using CNN to deal with sequential trajectory data.

*2) **Accuracy against non-uniform sampling rates**:* Next, we conduct data transformations on trajectories to evaluate the ability against non-uniform sampling rates. Specifically, we randomly downsample trajectory points at the rates varying between 0.2 and 0.6 to simulate the condition of having non-uniform sampling rates. The size of $D'_a \cup D'_b$ is fixed to 100K.

Table II presents the mean ranks of 1000 query trajectories versus different downsampling rates r_1 on both datasets. As the downsampling rate increases, the performance of all methods gets worse. Among all the baseline methods, EDwP performs best owing to the technique of linear interpolation it utilizes. t2vec performs relatively well when r_1 is low, but its mean ranks degrade quickly as r_1 increases, which implies that t2vec is not competent enough against non-uniform sampling rates. Note that all the methods get better results on Chengdu dataset. This is because trajectory data collected in Chengdu have higher sampling rate of appropriately 3 seconds per point, which alleviates the problem of sparsity and thus benefits the similarity computation.

*3) **Accuracy against noises**:* Then, we study the effect of noises on different methods. We randomly distort the coordinates of trajectory in D_a and $D'_a \cup D'_b$ to simulate the situation where noises exist. The distorting rate r_2 varies from 0.2 to 0.6 and $|D'_a \cup D'_b| = 100,000$.

From Table III, we can observe that the performance of all methods, except for EDR and LCSS, does not degrade much as r_2 increases, which means they are all robust to noises to some extent. Similar to the case of downsample, the mean rank of EDR starts to degrade dramatically when r_2 increases. EDwP and t2vec demonstrate similar results on both datasets. TrjSR outperforms all the other methods with a large margin on Porto dataset and has competitive results on Chengdu dataset, indicating that our method is the most robust against noises when computing trajectory similarity.

C. Parameter study

The number of convolutional kernel and residual block determines the quality of the SR image, and further affects the accuracy of similarity computation. To evaluate their influence,

TABLE III: Mean rank of 1000 query trajectories versus different distort rates r_2.

Porto					
r_2	0.2	0.3	0.4	0.5	0.6
LCSS	16.978	22.526	26.998	34.639	51.524
EDR	25.516	34.413	49.573	110.356	174.791
EDwP	3.585	3.668	3.612	3.431	3.545
Fréchet	8.855	8.616	8.612	7.939	8.652
Hausdorff	3.068	2.889	3.087	3.032	3.325
t2vec	3.365	3.809	5.074	5.683	10.518
TrjSR	**1.602**	**1.841**	**1.883**	**2.18**	**2.169**

Chengdu					
r_2	0.2	0.3	0.4	0.5	0.6
LCSS	104.834	139.937	184.449	249.332	301.863
EDR	**1.667**	6.906	34.54	70.086	122.792
EDwP	3.372	3.652	**3.492**	3.714	3.768
Fréchet	50.259	50.244	43.289	51.634	46.278
Hausdorff	7.576	8.642	8.025	8.6	8.281
t2vec	3.321	3.44	3.745	4.153	5.049
TrjSR	2.415	**3.006**	3.545	**3.627**	**3.208**

TABLE IV: The impact of the number of convolutional kernel and residual block on the accuracy using Porto dataset.

#convolutional kernel	8	16	32	64
$r_1 = 0.6$	72.814	**49.325**	230.858	83.523
$r_2 = 0.6$	1.998	**1.621**	3.773	2.302

#residual block	1	2	3	4	5
$r_1 = 0.6$	153.071	81.958	**63.054**	82.932	118.029
$r_2 = 0.6$	1.986	2.447	**1.544**	2.401	1.724

we compare the mean ranks of 1000 queries when $r_1 = 0.6$ and $r_2 = 0.6$ on Porto dataset.

As shown in Table IV, the combination of 3 residual blocks with 16 kernels in each convolutional layer of SISR model gives the best results.

D. Result of SISR

Recall that we aim to generate high-quality trajectory images from low-quality ones through SISR. Here, we present the qualitative results of our trajectory SISR as shown in Fig. 7. The proposed model successfully complements the incomplete segments with generated points and also reconstructs the details from noises.

E. Effect of image embedding

As we discussed in Section IV-D2, a trajectory image is not adequate to be directly used for similarity computation. Therefore, we embed trajectory images into a vector space. In this section, we evaluate the effectiveness of image embedding for trajectory similarity computation.

To measure the similarity of trajectory images directly for comparison, we independently train our SISR model from scratch on Porto dataset, with the same parameter settings as we use in the process of training TrjSR. As for the similarity measurements, we utilize MSE and structural similarity index measure (SSIM), which are commonly used similarity loss functions in the field of SISR. Similar to the implementation in Section V-B1, we retrieve the rank of $T'_a \in D'_a \cup D'_b$ for each

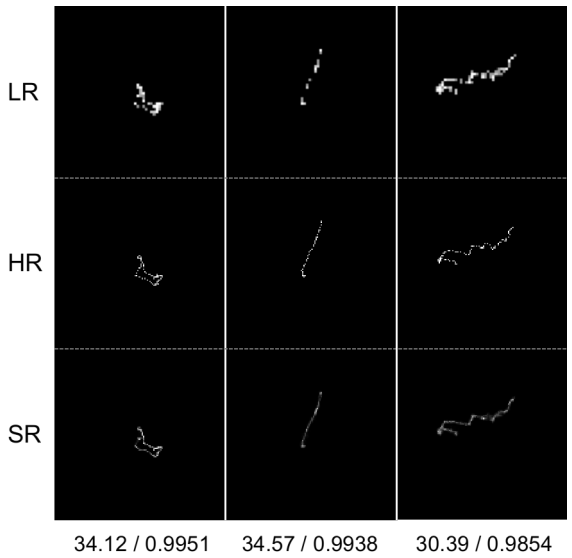


Fig. 7: Qualitative results (PSNR(dB) / SSIM) of trajectory image $\times 2$ super-resolution.

TABLE V: Mean rank for different similarity measurements versus different database sizes on Porto dataset.

database size	1k	2k	3k	4k
MSE	1.015	1.032	1.044	1.063
SSIM	1.082	1.187	1.257	1.341
Embedding	**1.01**	**1.017**	**1.028**	**1.041**

$T_a \in D_a$. The size of $D'_a \cup D'_b$ is reduced to the range between 2K and 4K, because the time consumption for calculating SSIM on large image datasets is unaffordable.

The results in Table V prove that simply minimizing MSE or SSIM on trajectory images does not bring about the most accurate results in similarity computation. Instead, with the trajectory image embedding, we can improve the accuracy of similarity computation as well as obtain the concise trajectory representation, which benefit our solution in time consumption when calculating the similarity.

F. Effect of L1 and L2 distance

To prove that L1 distance is better than L2 distance for measuring the similarity loss in our case as we analyzed in Section IV-D3a, we conduct the following experiment on Porto dataset to compare their effects on the accuracy of trajectory similarity computation.

We first pre-train our SISR model independently for 3 epochs to make it generate good-enough super-resolved images. Then, two embed blocks with the only difference of L1 distance and L2 distance in similarity loss function l_{vec} are used to jointly train with the SISR model for another 1 epoch respectively. Similar procedure for evaluating accuracy is applied as we did in Section V-B2 and Section V-B3.

As shown in Table VI, under the same training condition, the accuracy of using L2 distance declines rapidly as downsampling rate increases. As for distortion, using L1 distance brings about more robustness against distorting rate. From the above, we

TABLE VI: The impact of different distance measures on Porto dataset.

Distance	downsampling rate r_1			distorting rate r_2		
Measure	0.2	0.4	0.6	0.2	0.4	0.6
L2	**1.892**	94.759	142.065	1.246	1.702	12.674
L1	3.137	**12.806**	**85.234**	**0.739**	**1.322**	**1.157**

can conclude that L1 distance is more adequate to be used in similarity computation in our case. Besides, it is more computational expensive to compute L2 loss compared to L1 loss.

VI. Conclusion

In this paper, we proposed a novel generative model, TrjSR, to deal with the low-quality trajectory data caused by non-uniform sampling rates and noises in trajectory similarity computation. The SISR techniques are adopted to generate high-quality trajectory images, and we developed a similarity-aware perceptual loss function to encourage our model to generate high-quality trajectory images in favor of computing similarity. In this sense, the proposed TrjSR bridged the gap between the sequential trajectory data and the SISR. Extensive experimental results showed that the proposed method outperforms existing methods in terms of accuracy. Moreover, we conducted comparative study to further validate the effectiveness of our method. In addition, the embedding vector of trajectory images can be used not only for similarity computation but also as the trajectory representation for other applications, which we leave for our future investigation.

Acknowledgment

This work is supported by the National Natural Science Foundation of China (NSFC) (Grant No. 52071312), and the Open Program of Zhejiang Lab (Grant No. 2019KE0AB03).

References

[1] L.-A. Tang, Y. Zheng, J. Yuan, J. Han, A. Leung, C.-C. Hung, and W.-C. Peng, "On discovery of traveling companions from streaming trajectories," in *2012 IEEE 28th International Conference on Data Engineering*. IEEE, 2012, pp. 186–197.

[2] F. Zhou, Q. Gao, G. Trajcevski, K. Zhang, T. Zhong, and F. Zhang, "Trajectory-user linking via variational autoencoder," *IJCAI International Joint Conference on Artificial Intelligence*, vol. 2018-July, pp. 3212–3218, 2018.

[3] B. Han, L. Liu, and E. Omiecinski, "A systematic approach to clustering whole trajectories of mobile objects in road networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 29, no. 5, pp. 936–949, 2017.

[4] Z. Fu, W. Hu, and T. Tan, "Similarity based vehicle trajectory clustering and anomaly detection," in *IEEE International Conference on Image Processing 2005*, vol. 2. Ieee, 2005, pp. II–602.

[5] X. Li, K. Zhao, G. Cong, C. S. Jensen, and W. Wei, "Deep representation learning for trajectory similarity computation," in *2018 IEEE 34th International Conference on Data Engineering (ICDE)*, 2018, pp. 617–628.

[6] National Coordination Office. (2020, April) How accurate is gps? [Online]. Available: https://www.gps.gov/systems/gps/performance/accuracy/#how-accurate

[7] B. K. Yi, H. V. Jagadish, and C. Faloutsos, "Efficient retrieval of similar time sequences under time warping," in *Proceedings - International Conference on Data Engineering*, 1998.

[8] M. Vlachos, G. Kollios, and D. Gunopulos, "Discovering similar multidimensional trajectories," *Proceedings - International Conference on Data Engineering*, 2002.

[9] L. Chen, M. T. Özsu, and V. Oria, "Robust and fast similarity search for moving object trajectories," in *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, 2005, pp. 491–502.

[10] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2015.

[11] L. CHEN and R. NG, "On The Marriage of Lp-norms and Edit Distance," in *Proceedings 2004 VLDB Conference*, 2004.

[12] S. Ranu, P. Deepak, A. D. Telang, P. Deshpande, and S. Raghavan, "Indexing and matching trajectories under inconsistent sampling rates," in *Proceedings - International Conference on Data Engineering*, 2015.

[13] T. Eiter and H. Mannila, "Computing discrete Fréchet distance," *Notes*, 1994.

[14] E. Belogay, C. Cabrelli, U. Molter, and R. Shonkwiler, "Calculating the Hausdorff distance between curves," *Information Processing Letters*, 1997.

[15] Y. Zhang, A. Liu, G. Liu, Z. Li, and Q. Li, "Deep Representation Learning of Activity Trajectory Similarity Computation," *2019 IEEE International Conference on Web Services (ICWS)*, pp. 312–319, 2019.

[16] T. Y. Fu and W. C. Lee, "TremBR: Exploring road networks for trajectory representation learning," *ACM Transactions on Intelligent Systems and Technology*, 2020.

[17] D. Yao, G. Cong, C. Zhang, and J. Bi, "Computing trajectory similarity in linear time: A generic seed-guided neural metric learning approach," in *Proceedings - International Conference on Data Engineering*, 2019.

[18] H. Zhang, X. Zhang, Q. Jiang, B. Zheng, Z. Sun, W. Sun, and C. Wang, "Trajectory similarity learning with auxiliary supervision and optimal matching," in *IJCAI International Joint Conference on Artificial Intelligence*, 2020.

[19] X. Li and M. T. Orchard, "New edge-directed interpolation," *IEEE Transactions on Image Processing*, 2001.

[20] R. G. Keys, "Cubic Convolution Interpolation for Digital Image Processing," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, 1981.

[21] C. E. Duchon, "LANCZOS FILTERING IN ONE AND TWO DIMENSIONS." *Journal of applied meteorology*, 1979.

[22] M. E. Tipping and C. M. Bishop, "Bayesian image super-resolution," in *Advances in Neural Information Processing Systems*, 2003.

[23] S. Baker and T. Kanade, "Limits on super-resolution and how to break them," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2002.

[24] W. T. Freeman, E. C. Pasztor, and O. T. Carmichael, "Learning low-level vision," *International Journal of Computer Vision*, 2000.

[25] C. Dong, C. C. Loy, K. He, and X. Tang, "Image super-resolution using deep convolutional networks," *IEEE transactions on pattern analysis and machine intelligence*, vol. 38, no. 2, pp. 295–307, 2015.

[26] C. Dong, C. C. Loy, and X. Tang, "Accelerating the super-resolution convolutional neural network," in *European conference on computer vision*. Springer, 2016, pp. 391–407.

[27] W. Shi, J. Caballero, F. Huszar, J. Totz, A. P. Aitken, R. Bishop, D. Rueckert, and Z. Wang, "Real-Time Single Image and Video Super-Resolution Using an Efficient Sub-Pixel Convolutional Neural Network," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

[28] J. Kim, J. K. Lee, and K. M. Lee, "Accurate image super-resolution using very deep convolutional networks," in *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 2016.

[29] K. Zhang, W. Zuo, S. Gu, and L. Zhang, "Learning deep CNN denoiser prior for image restoration," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017.

[30] C. Ledig, L. Theis, F. Huszár, J. Caballero, A. Cunningham, A. Acosta, A. Aitken, A. Tejani, J. Totz, Z. Wang, and W. Shi, "Photo-realistic single image super-resolution using a generative adversarial network," in *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, 2017.

[31] B. Lim, S. Son, H. Kim, S. Nah, and K. M. Lee, "Enhanced Deep Residual Networks for Single Image Super-Resolution," in *IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, 2017.