Shape-based Trajectory Clustering

Telmo Pires
telmopires@tecnico.ulisboa.pt

Instituto Superior Técnico, Lisboa, Portugal

October 2016

Abstract

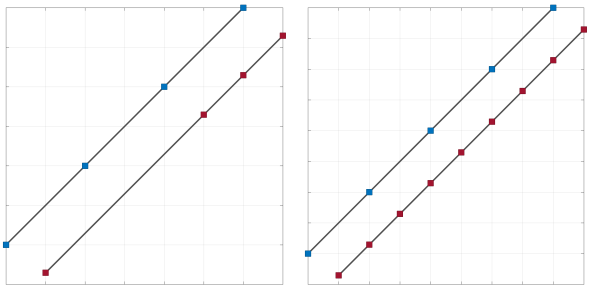
Automatic trajectory classification has countless applications, ranging from the natural sciences, such as zoology and meteorology, to urban planning and sports analysis, and has generated great interest and investigation. The purpose of this work is to propose and test new methods for trajectory clustering, based on shape, rather than spatial position, as is the case with previous methods. The proposed approach starts by uniformly resampling the trajectories using splines, and then characterizes them using the angles of the tangents at the resampled points. Angular data introduces some challenges for analysis, due to its periodic nature, therefore preventing the direct application of common clustering techniques. To overcome this problem, three methods are proposed/adapted: a variant of the k-means algorithm, a mixture model using multivariate Von Mises distributions, which is fitted using the EM algorithm, and sparse nonnegative matrix factorization. Since the number of clusters is seldom known *a priori*, methods for automatic model selection are also introduced. Finally, these techniques are tested on both real and synthetic data, and the viability of this approach is demonstrated.
Keywords: Unsupervised Learning, Directional Statistics, k-means, NMF, EM Algorithm, Trajectory Clustering.

1. Introduction

In recent years, an exponential growth in the amount of available data for analysis has been experienced in most fields of knowledge. These huge data quantities, far beyond the scope of manual analysis, have stimulated a growing interest in automated methods. Trajectory data is no exception, thanks to the growing number of tracking devices, including GPS receivers, RFID tags, tracking cameras, and even cellphone call traces [2]. This has led to a growing interest in automatic trajectory clustering, as a means to perform activity recognition and classification. The study of traffic patterns, human mobility, and even air pollution exposure of populations [6]; the detection of common behaviors in meteorological phenomena [7]; animal movement analysis [3]; automated sports analysis; and automatic surveillance are just a few of the applications of these techniques.

Besides being influenced by many factors not directly related to the phenomenon in study, such as seasonality, in the case of hurricane tracks, trajectories introduce many challenges by being represented by a sequence of space-time data points:

1. Differences in the speed of tracked objects cause misalignments in the trajectories, as illustrated in Figure 1(a). Varying sampling rates have a similar effect.
2. Similar trajectories may have a different number of points, as illustrated in 1(b).
3. In some applications, trajectories differing only in a translation should be treated as being similar.
4. Noise and outliers: besides being affected by noise, trajectories can also have big, but sparse errors, called outliers.



(a) Objects moving at different speeds, leading to misaligned, but similar tracks.

(b) Similar tracks with a different number of points.

Figure 1: Some of the challenges faced with trajectory data.

In this work, a novel approach is introduced for clustering $2D$ trajectories, based exclusively on

their shapes. This means trajectories are clustered independently of their location and of a global change of scale.

2. Previous works

Other authors have studied the problem of trajectory clustering. Their approaches can be divided into two main groups: the distance metric approach and the model based (generative) approach.

2.1. Distance Metric Approaches

These approaches aim to find (dis)similarity measures between trajectories, and then cluster them based on those metrics. One such metric is the Euclidean distance, used in [10, 12]. Methods based on this metric require trajectories with the same number of points and uniformly sampled, so a pre-processing step is needed.

To avoid the need for uniformly sampled trajectories, methods that try to find an optimal alignment between time-varying sequences, such as *dynamic time warping* [22] and *longest common subsequence* [23] have been successfully applied to trajectory clustering.

Another important method is TRACLUS [16]. This method allows sub-trajectory clustering, instead of clustering the trajectories as a whole, which may be useful in some applications. The way it works is by partitioning trajectories in sub-trajectories, and then performing density-based clustering, using a specially defined metric.

2.2. Model-based Approaches

Model-based methods, on the other hand, attempt to find a model capable of describing the whole dataset, instead of directly comparing trajectories.

In [11], trajectories are represented as polynomial functions of time, and the parameters of these functions are modeled as a finite mixture. A similar approach is used in [24], but trajectories are modeled using B-splines.

Another common approach to trajectory clustering models trajectories as vector fields. Some of these works are [8], in which a variant of k-means for vector fields was proposed, and [20], where a mixture of vector fields model allowing transitions between fields was proposed.

3. Data Pre-processing

Trajectories are usually given as sequences of points, so a pre-processing step is needed in order to capture their shapes. In our case, shape is characterized as a sequence of angles: the angles of the tangents to the trajectory, sampled at uniformly spaced points, as illustrated in Figure 2. As seen below, the sequence of angles is invariant under changes in location and scale of the trajectory.

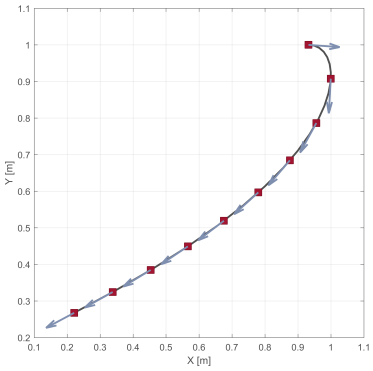


Figure 2: Uniformly resampled trajectory and respective tangents. Shape is characterized by the angles of the tangent vectors w.r.t. the horizontal.

This pre-processing step solves the problem of trajectories having different lengths and the problem of misalignment, by assuring all trajectories are similarly sampled. In this work, the resampling is performed using cubic splines, fitted separately for both the x and y coordinates. In order to fit splines, an independent variable is needed, and it must be the same for each coordinate. A natural choice is time, but many datasets only have spatial information. In these cases, the independent variable must be estimated from data, using

$$\tau_{p+1} = \tau_p + \sqrt{(x_{p+1}^{(i)} - x_p^{(i)})^2 + (y_{p+1}^{(i)} - y_p^{(i)})^2}, \quad (1)$$

with $\tau_1 = 0$, where we assume that the speed is uniform.

To reduce the influence of noise, we use *smoothing splines*, which are fitted by minimizing the cost [5]

$$p \sum_{i=1}^{m} w_i (x(\tau_i) - f(\tau_i))^2 + (1-p) \int (D^2 f(\tau))^2 d\tau, \quad (2)$$

where p is a parameter that controls the trade-off between smoothness and accuracy, w_i are weights we set to 1, and $D^2 f(\tau)$ is the second derivative of function f. Many implementations of these algorithms are readily available, such as the Curve Fitting toolbox in `MATLAB`, which was used in this work.

For a uniform resampling to d points, splines should be evaluated at:

$$\tau_i = \tau_1 + \frac{\tau_m - \tau_1}{d-1}(i-1) \ , \quad i = 1, \ldots, d,$$

where τ_1 and τ_m are the τ's calculated according to Eq. (1).

The angles of the tangents are obtained by

$$\omega_p^{(i)} = \text{atan2}(Df_y^{(i)}(\tau_p), Df_x^{(i)}(\tau_p)) \ , \quad \omega_p^{(i)} \in [-\pi, \pi] \, ,$$

where atan2(·) is the 4-quadrant inverse tangent function and $Df_y^{(i)}(\tau_p)$ and $Df_x^{(i)}(\tau_p)$ refer to the first derivatives of the fitted splines, evaluated at τ_p. This way, we get a d-dimensional vector $\boldsymbol{\omega}^{(i)}$ describing the trajectory.

Besides being invariant to spatial translation and scaling, another interesting property of this formulation is that it allows trajectories to be invariant to rotation with a small modification: by working with the differences between consecutive angles, instead of the angles themselves.

3.1. Choosing Parameters p and d

The pre-processing step requires two input parameters: p and d. For choosing p, no systematic procedure is available, and so we try several values until a good one is found. The choice of d, on the other hand, can be automated. This is done by finding the smallest number of points that can accurately describe a trajectory, called the characteristic points, which provides a lower bound for d, as the characteristic points may not be equally spaced. In order to fully characterize the trajectories, we use d approximately 5 times the maximum number of characteristic points found in the given dataset.

In [16], a method for finding the characteristic points of a trajectory was proposed, which uses the minimum description length (MDL) principle. The idea is to loop though all the points in a trajectory and to compute, $MDL_{par}(p_{start}, p_{curr})$, the MDL cost assuming p_{start} and p_{curr} are the only characteristic points in the interval $p_{start}, \ldots, p_{curr}$, and $MDL_{nopar}(p_{start\_index}, p_{curr\_index})$, the MDL cost preserving the original trajectory. If $MDL_{par} \leq MDL_{nopar}$, then choosing p_{curr} as a characteristic point makes the MDL cost lower. This procedure is shown in algorithm 1.

The MDL cost, or MDL length, of a given partitioning consists of two components, $L(H)$ and $L(D|H)$, given by

$$L(H) = \sum_{j=1}^{par_i - 1} \log_2(length(p_{c_j} p_{c_{j+1}})),$$

$$L(D|H) = \sum_{j=1}^{par_i - 1} \sum_{k=c_j}^{c_{j+1}-1} \left[\log_2(d_\perp(p_{c_j}p_{c_{j+1}}, p_k p_{k+1})) + \log_2(d_\theta(p_{c_j}p_{c_{j+1}}, p_k p_{k+1}))\right],$$

where par_i is the number of characteristic points in trajectory i, $length(\cdot)$ is the euclidean length of a segment, $p_a p_b$ is the segment defined by points p_a and p_b and $d_\perp$ and d_θ are the perpendicular and angular distances, defined as:

$$d_\perp(s_i e_i, s_j e_j) = \frac{l_{\perp 1}^2 + l_{\perp 2}^2}{l_{\perp 1} + l_{\perp 2}}$$

Algorithm 1: Approximate Trajectory Partitioning.

Input: A trajectory, as a sequence of points, $p_1, \ldots, p_m$
Output: N_c, the number of characteristic points

```
1  N_c = 1  // Add the starting point
2  start_index = 1
3  length = 1
4  while start_index + length ≤ m do
5      curr_index = start_index + length
6      cost_par = MDL_par(p_start_index, p_curr_index)
7      cost_nopar = MDL_nopar(p_start_index, p_curr_index)
8      if cost_par > cost_nopar then
           // Partition at the previous point
9          N_c = N_c + 1
10         start_index = curr_index − 1
11         length = 1
12     else
13         length = length + 1
14     end
15 end
16 N_c = N_c + 1  // Add the ending point
```

$$d_\theta(s_i e_i, s_j e_j) = \begin{cases} length(s_j e_j) \times \sin\theta & 0 \leq \theta < \pi/2 \\ length(s_j e_j), & \pi/2 \leq \theta \leq \pi \end{cases}$$

where $l_{\perp 1}$ and $l_{\perp 2}$ are the distances illustrated in Figure 3.

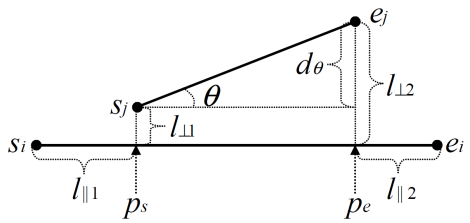


Figure 3: Definition of distance functions from TRACLUS. Image from [16].

To reduce the influence of noise in the procedure above, we only apply it to estimate d after having smoothed the trajectories with the p given by the user and using d twice the length of a trajectory, to assure its shape is well captured.

From this point on, whenever we mention trajectories we mean the trajectories *after* pre-processing, i.e., the set $\{\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(m)}\}$, where each trajectory $\boldsymbol{\omega}^{(i)}$ is a sequence of angles $\{\omega_1^{(i)}, \ldots, \omega_d^{(i)}\}$.

4. K-means

4.1. Introduction

The k-means algorithm is one of the most famous clustering techniques [13], despite offering no global optimality guarantees. In fact, it has been shown to give arbitrarily bad results in certain conditions. Arguably, its popularity is due to the fact that it is a very simple and fast approach to clustering.

Given a set of points and the number of desired clusters, k-means stores a set of k centroids, $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$, and alternates between two steps: assignment and centroid update.

1. Assignment step: set $c^{(i)} := \arg\min_j \|\boldsymbol{x}^{(i)} - \boldsymbol{\mu}_j\|^2$ for all i, where $c^{(i)} = j$ if the ith input sample belongs to cluster j. In other words, assign each sample to the closest centroid. Possible ties are broken by some arbitrary rule.
2. Centroid update step: set $\boldsymbol{\mu}_j := \frac{\sum_{i=1}^{m} 1\{c^{(i)}=j\}\boldsymbol{x}^{(i)}}{\sum_{i=1}^{m} 1\{c^{(i)}=j\}}$ for all j, where $1\{c^{(i)} = j\}$ is an indicator function, taking the value 1 if $c^{(i)} = j$ and 0 otherwise, that is, move each centroid to the mean of the points assigned to it.

Originally, k-means was developed to work with Euclidean distances, although many other versions have been proposed in the literature. In our case, we need to adapt it to handle the sequences of angles that describe each trajectory.

4.2. Measuring Distances

A classical measure of the distance between two angles is the length of the chord between them, as illustrated in Figure 4. It can be shown that, if the

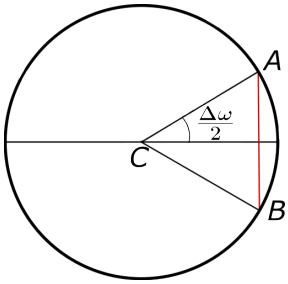


Figure 4: Relationship between angular difference and chord (in red) in a unit circle.

difference between two angles is $\Delta\omega$, the length of the corresponding chord (red line in Fig. 4) is given by $2\sin\frac{\Delta\omega}{2}$. Since sin is an odd function, it returns a negative length if the angular difference is negative. This is unintended, so we take the squares,

$$\left(2\sin\frac{\Delta\omega}{2}\right)^2 = 2\left(1-\cos\Delta\omega\right) \propto 1-\cos\Delta\omega.$$

The constant "2" is irrelevant and can be dropped, since this function will only be used to compare distances. Thus, the final metric is $1-\cos\Delta\omega$. The extension of this expression to d-dimensional data is trivial:

$$D^2(\boldsymbol{\omega}^{(1)}, \boldsymbol{\omega}^{(2)}) = \sum_{p=1}^{d}\left(1-\cos(\omega_p^{(1)} - \omega_p^{(2)})\right), \quad (3)$$

where the $\boldsymbol{\omega}^{(i)}$'s are angular vectors and $\omega_p^{(i)}$ is the pth component of vector i. The notation D^2 is used to indicate this metric is a squared value (of the chord).

4.3. Updating the Centroids

The arithmetic mean cannot be used for updating the coordinates of a centroid, as it does not work well with circular data. In this work, we define the mean of a sequence of angles $\{\omega^{(1)}, \ldots, \omega^{(m)}\}$ as:

$$\bar{\omega} = \operatorname{atan2}\left(\frac{\frac{1}{m}\sum_{i=1}^{m}\sin\omega^{(i)}}{\frac{1}{m}\sum_{i=1}^{m}\cos\omega^{(i)}}\right) \quad (4)$$

The extension of this result to vectorial data is straightforward: the mean of an angular vector is the angular mean of each of its coordinates.

4.4. Algorithm Description

Algorithm 2: *circular* $k-means$ algorithm.

Input: Data set, $\{\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(m)}\}$, and number of clusters, k
Output: Cluster centroids, $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$, and cluster assignments, $c^{(1)}, \ldots, c^{(m)}$

Initialize k cluster centroids: $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k$
repeat
  foreach i **do**
    Set $c^{(i)} := \arg\min_j D(\boldsymbol{\omega}^{(i)}, \boldsymbol{\mu}_j)$ `// Assign centroids to samples`
  end
  foreach j **do**
    Set $\mu_j^p = \arg\left(\frac{\sum_{i=1}^{m} 1\{c^{(i)}=j\}e^{j\omega_p^{(i)}}}{\sum_{i=1}^{m} 1\{c^{(i)}=j\}}\right)$ for $p = 1, \ldots, d$ `// Update each coordinate of the centroids`
  end
until *convergence*

For the algorithm to be completely described, initialization and stopping criteria must be specified. Regarding initialization, k-means++ [1] is used, but using Eq. (3) instead of the Euclidean distance. The termination condition is when the assignments no longer change between consecutive iterations.

4.5. Model Selection

In real-world situations, the number of clusters, k, is not given, and it is necessary to estimate it. One common choice used in conjunction with k-means is the so-called elbow method. This method involves computing a distortion function for several values of k and finding an elbow, that is, a point after which the rate of decrease is significantly smaller. The distortion function is just the sum of the distances of each data sample to the respective

centroid. For circular k-means, this is given by

$$J(\boldsymbol{\omega}^{(1)},\ldots,\boldsymbol{\omega}^{(m)},c^{(1)},\ldots,c^{(m)},\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_k) = \sum_{i=1}^{m}\sum_{p=1}^{d}\left(1-\cos(\omega_p^{(i)}-\mu_{c^{(i)}}^p)\right).$$

5. Finite Mixture Models

Given a set of d-dimensional trajectories, $\{\boldsymbol{\omega}^{(1)},\ldots,\boldsymbol{\omega}^{(m)}\}$, which is to be partitioned into k clusters, we can model this set as i.i.d. samples of a finite mixture,

$$p(\boldsymbol{\omega}^{(i)}|\boldsymbol{\theta}) = \sum_{j=1}^{k}\alpha_j p(\boldsymbol{\omega}^{(i)}|z^{(i)}=j,\boldsymbol{\theta}),$$

where the $z^{(i)}$'s are latent variables, with $z^{(i)}=j$ meaning the ith trajectory belongs to cluster j. Each parameter $\alpha_j = p(z^{(i)}=j)$ is the mixing probability, and $\boldsymbol{\theta}$ a set of parameters characterizing the distributions. The likelihood function of the given set of trajectories can be written as

$$p(\boldsymbol{\omega}^{(1)},\ldots,\boldsymbol{\omega}^{(m)}|\boldsymbol{\alpha},\boldsymbol{\theta}) = \prod_{i=1}^{m}p(\boldsymbol{\omega}^{(i)}|\boldsymbol{\alpha},\boldsymbol{\theta}) = \prod_{i=1}^{m}\sum_{j=1}^{k}\alpha_j p(\boldsymbol{\omega}^{(i)}|z^{(i)}=j,\boldsymbol{\theta}). \quad (5)$$

A popular distribution for modeling circular data is the Von Mises distribution, making it a natural choice for our mixture model. This way, the $p(\boldsymbol{\omega}^{(i)}|z^{(i)}=j,\boldsymbol{\theta})$ are Von Mises pdf's. The pdf of a multivariate Von Mises is [19]

$$p(\boldsymbol{\omega};\boldsymbol{\mu},\boldsymbol{\kappa},\boldsymbol{\Lambda}) = T(\boldsymbol{\kappa},\boldsymbol{\Lambda})^{-1}\exp\left(\boldsymbol{\kappa}^T c(\boldsymbol{\omega},\boldsymbol{\mu}) + \frac{1}{2}s(\boldsymbol{\omega},\boldsymbol{\mu})^T\boldsymbol{\Lambda}\ s(\boldsymbol{\omega},\boldsymbol{\mu})\right), \quad (6)$$

where

$$c(\boldsymbol{\omega},\boldsymbol{\mu})^T = [\cos(\omega_1-\mu_1),\ \ldots\ ,\ \cos(\omega_d-\mu_d)],$$

$$s(\boldsymbol{\omega},\boldsymbol{\mu})^T = [\sin(\omega_1-\mu_1),\ \ldots\ ,\ \sin(\omega_d-\mu_d)],$$

and $T(\boldsymbol{\kappa},\boldsymbol{\Lambda})^{-1}$ is a normalizing constant, which is unknown in explicit form for $d>2$. Parameters $\boldsymbol{\mu}$ and $\boldsymbol{\kappa}$ are the d-dimensional mean and concentration of the distribution; $\boldsymbol{\Lambda}$ is a $d\times d$ symmetric matrix (with zeros in the diagonal), where entry Λ_{ij} measures the dependence between ω_i and ω_j. Matrix $\boldsymbol{\Lambda}$ can be seen as the analogous to the covariance matrix in a Gaussian distribution.

The usual choice for obtaining maximum likelihood (ML) or maximum a posteriori (MAP) estimates of the mixture parameters is the EM algorithm, which is an iterative procedure with two steps [21]:

- E-step: Set $w_j^{(i)} = p(z^{(i)}=j|\boldsymbol{\omega}^{(i)};\boldsymbol{\alpha},\boldsymbol{M},\boldsymbol{K},\boldsymbol{L})$, where $\boldsymbol{M}$, $\boldsymbol{K}$, and $\boldsymbol{L}$ are used in place of $\{\boldsymbol{\mu}_1,\ldots,\boldsymbol{\mu}_k\}$, $\{\boldsymbol{\kappa}_1,\ldots,\boldsymbol{\kappa}_k\}$, and $\{\boldsymbol{\Lambda}_1,\ldots,\boldsymbol{\Lambda}_k\}$, respectively.

  Using Bayes' Theorem, the E-step reduces to computing, for every i and j,

$$w_j^{(i)} = \frac{\alpha_j p(\boldsymbol{\omega}^{(i)}|z^{(i)}=j;\boldsymbol{\mu}_j,\boldsymbol{\kappa}_j,\boldsymbol{\Lambda}_j)}{\sum_{l=1}^{k}\alpha_l p(\boldsymbol{\omega}^{(i)}|z^{(i)}=l;\boldsymbol{\mu}_l,\boldsymbol{\kappa}_l,\boldsymbol{\Lambda}_l)}, \quad (7)$$

  where $p(\boldsymbol{\omega}^{(i)}|z^{(i)}=j;\boldsymbol{\mu}_j,\boldsymbol{\kappa}_j,\boldsymbol{\Lambda}_j)$ is a multivariate Von Mises pdf.

- M-step: this step requires maximizing the following function, with respect to the parameters of the model:

$$\sum_{i=1}^{m}\sum_{j=1}^{k}w_j^{(i)}\ln\frac{p(\boldsymbol{\omega}^{(i)}|z^{(i)}=j;\boldsymbol{\mu}_j,\boldsymbol{\kappa}_j,\boldsymbol{\Lambda}_j)p(z^{(i)}=j;\boldsymbol{\alpha})}{w_j^{(i)}}$$

  The multivariate Von Mises pdf depends on $T(\boldsymbol{\kappa}_j,\boldsymbol{\Lambda}_j)$, an unknown constant, which presents a challenge for ML or MAP estimation. To avoid having to use special numeric procedures, we assume that the components of the $\boldsymbol{\omega}^{(i)}$ vector are *independent* of the others, which makes the multivariate Von Mises distribution equivalent to the product of d (the number of features) univariate Von Mises distributions:

$$p(\boldsymbol{\omega}^{(i)}|z^{(i)}=j;\boldsymbol{\mu}_j,\boldsymbol{\kappa}_j,\boldsymbol{\Lambda}_j) = \prod_{p=1}^{d}\frac{e^{\kappa_j^p\cos(\omega_p^{(i)}-\mu_j^p)}}{2\pi I_0(\kappa_j^p)}, \quad (8)$$

  where μ_j^p, κ_j^p, and $\omega_p^{(i)}$ refer to the pth component of vectors $\boldsymbol{\mu}_j$, $\boldsymbol{\kappa}_j$ and $\boldsymbol{\omega}^{(i)}$, respectively, and I_0 is the modified Bessel function of first kind and order 0.

  Using ML estimation, the update equations for μ_l^q (the qth component of $\boldsymbol{\mu}_l$), α_l and κ_l^q are as follows:

$$\mu_l^q = \operatorname{atan2}\left(\frac{\sum_{i=1}^{m}w_l^{(i)}\sin\omega_q^{(i)}}{\sum_{i=1}^{m}w_l^{(i)}\cos\omega_q^{(i)}}\right), \quad (9)$$

$$\alpha_l = \frac{1}{m}\sum_{i=1}^{m}w_l^{(i)}, \quad (10)$$

$$\kappa_l^q = A_1^{-1}\left(\frac{\sum_{i=1}^{m}w_l^{(i)}\cos(\omega_q^{(i)}-\mu_l^q)}{\sum_{i=1}^{m}w_l^{(i)}}\right), \quad (11)$$

  where $A_1(\cdot)=\frac{I_1(\cdot)}{I_0(\cdot)}$, with I_1 and I_0 denoting the modified Bessel functions of the first kind and orders 1 and 0, respectively.

In some cases, it is of interest to restrict $\boldsymbol{\kappa}_l$, in order to avoid overfitting small datasets. We refer to this variant as a constrained Von Mises mixture (VMM), as opposed to the unconstrained (previous) case. To constrain the concentration parameter, the components of the $\boldsymbol{\kappa}_l$ are forced to be the same, i.e., $\kappa_l^1 = \kappa_l^2 = \ldots = \kappa_l^d = \kappa_l$, so (11) is replaced by

$$\kappa_l = A_1^{-1}\left(\frac{1}{d}\sum_{p=1}^{d}\frac{\sum_{i=1}^{m} w_l^{(i)}\cos(\omega_p^{(i)} - \mu_l^p)}{\sum_{i=1}^{m} w_l^{(i)}}\right). \quad (12)$$

5.1. Prior on the Concentration Parameter

Due to the asymptotic behavior of the A_1^{-1} function, the estimation of the concentration parameter is prone to numerical problems. One way to deal with this is to introduce a prior, limiting its range of possible values. In this work, the conjugate prior of the Von Mises distribution is used [9],

$$p(\kappa; c, R_0) = \begin{cases} \frac{1}{K}\frac{e^{\kappa R_0}}{I_0(\kappa)^c} & , \quad \kappa > 0 \\ 0 & , \quad \text{otherwise,} \end{cases} \quad (13)$$

where $K = \int_0^\infty \frac{e^{\kappa R_0}}{I_0(\kappa)^c}$ is a normalizing constant, and $c > 0$ and $R_0 \leq c$ are parameters that control the shape of the prior. Using this prior, the update rule for κ_l^q is:

$$\kappa_l^q = A_1^{-1}\left(\frac{1}{1+c}\left[\frac{\sum_{i=1}^{m} w_l^{(i)}\cos(\omega_q^{(i)} - \mu_l^q)}{\sum_{i=1}^{m} w_l^{(i)}} + R_0\right]\right). \quad (14)$$

5.2. Algorithm Description

Algorithm 3: Mixture of Von Mises for trajectory clustering algorithm.

Input: $\{\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(m)}\}$ and the number of clusters, k
Output: $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k\}$, $\{\boldsymbol{\kappa}_1, \ldots, \boldsymbol{\kappa}_k\}$ and $\boldsymbol{\alpha}$

Initialize $\boldsymbol{\alpha}$, $\{\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_k\}$, $\{\boldsymbol{\kappa}_1, \ldots, \boldsymbol{\kappa}_k\}$
repeat
  `// E-step`
  foreach i, j **do**
    Set $w_j^{(i)} := \frac{\alpha_j p(\boldsymbol{\omega}^{(i)}|z^{(i)}=j;\boldsymbol{\mu}_j,\boldsymbol{\kappa}_j)}{\sum_{j=1}^{k}\alpha_j p(\boldsymbol{\omega}^{(i)}|z^{(i)}=l;\boldsymbol{\mu}_j,\boldsymbol{\kappa}_j)}$
  end
  `// M-step`
  Update α_j with Eq. (10) for $j = 1 \ldots, k$
  Update μ_j^p with Eq. (9) for $j = 1 \ldots, k$ and $p = 1, \ldots, d$
  Update κ_j^p with Eq. (11), (12), or (14) for $j = 1 \ldots, k$ and $p = 1, \ldots, d$
until *convergence condition is satisfied*
`// Compute cluster assignments`
foreach $i = 1, \ldots, m$ **do**
  $c^{(i)} = \arg\max_{j=1,\ldots,k} w_j^{(i)}$
end

To completely described the algorithm, an initialization procedure and a termination condition are needed. Regarding initialization, we set $\alpha_j = {}^1/_k, \forall j$, that is, we assume all clusters are equally likely; $\boldsymbol{\kappa}_j$ is initialized with the global κ over all dimensions and all samples; the centroids are initialized with the results of running k-means with k-means++ initialization. Regarding termination, the algorithms are executed until the change in likelihood is less than a fraction $\epsilon = 10^{-4}$ of its current value.

5.3. Model Selection

One advantage of using probabilistic methods is the ability to use formal approaches for model selection, such as the MDL principle [17].

The MDL cost has two components: the length of the model, $L(H)$, and the length of data encoded with the model, $L(D|H)$. In our case, the MDL cost can be shown to be

$$MDL_{cost} = -\sum_{i=1}^{m}\ln\sum_{j=1}^{k}\alpha_j p(\boldsymbol{\omega}^{(i)}|z^{(i)}=j,\boldsymbol{\mu}_j,\boldsymbol{\kappa}_j) + \frac{c}{2}\ln m,$$

where c is the number of parameters of the model. For unconstrained VMM's, $c = k \times (2d+1)$; for constrained, $c = k \times (d+2)$.

For selecting the number of clusters, several values are tried, and the one minimizing the MDL cost is chosen. To avoid local optima, EM is run several times for each k, and only the highest likelihood solution is considered [17].

6. Matrix Factorization

Nonnegative matrix factorization (NMF) is a special type of matrix factorization where a nonnegative data matrix, V, is decomposed as the product of two low-rank matrices, W and H, also nonnegative [15]. One of the reasons for the recent interest NMF has received is its close relationship with k-means. In fact, it has been shown that it is equivalent to k-means, with appropriate constraints. Since in many cases we are interested in factorizing matrices with negative entries, variants such as semi-NMF have been proposed [4].

Unfortunately, the constraints that make NMF equivalent to k-means lead to an intractable problem, thus they must be relaxed. One way to do this is by using sparse semi-NMF (SSNMF) [18], which solves the following optimization problem:

$$\min_{W,H\geq 0}\left[\|V - WH\|_F^2 + \eta\|W\|_F^2 + \beta\sum_{i=1}^{m}|\boldsymbol{h}_i|^2\right], \quad (15)$$

where $|\boldsymbol{h}_i|$ is the ℓ_1-norm of the ith column of H, while η and β are tuning parameters: $\eta > 0$ controls

the size of the elements of W, while $\beta > 0$ controls the trade-off between sparseness and accuracy of the approximation.

6.1. Application to Circular Data

Unfortunately, the standard factorization techniques do not directly work with circular data. One way to deal with this is to convert the angles to unit vectors, which do not have the problems of circular data. This can be done by stacking the set of trajectories, $\{\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(m)}\}$, as columns of the $\boldsymbol{\Omega}$ matrix, and use the duality between $2D$ vectors and complex numbers. This way, we can define matrix V as:

$$V = e^{j\boldsymbol{\Omega}} = \begin{bmatrix} | & | & & | \\ e^{j\boldsymbol{\omega}^{(1)}} & e^{j\boldsymbol{\omega}^{(2)}} & \cdots & e^{j\boldsymbol{\omega}^{(m)}} \\ | & | & & | \end{bmatrix},$$

where j is the imaginary unit and $e^{j\boldsymbol{\omega}^{(i)}}$ is the complex vector:

$$e^{j\boldsymbol{\omega}^{(i)}} = \left[e^{j\omega_1^{(i)}}, \ldots, e^{j\omega_d^{(i)}}\right]^T,$$

where $\omega_p^{(i)}$ is the pth entry of trajectory $\boldsymbol{\omega}^{(i)}$. In this work we use the exponential of a matrix as the exponential of its entries. It can be shown that factorizing this matrix is approximately equivalent to minimizing the squares of the chords corresponding to the differences between angles.

Using this matrix, we may use SSNMF for clustering. One last step can be applied to avoid working with complex valued matrices: since the real and imaginary parts of matrix V are treated independently, one can get the same results by factorizing V' instead, where

$$V' = \begin{bmatrix} \Re\{e^{j\boldsymbol{\Omega}}\} \\ \Im\{e^{j\boldsymbol{\Omega}}\} \end{bmatrix}. \tag{16}$$

The W matrix will change by a similar transformation. Since we are only interested in the angles, and not the vectors themselves, the matrix of the shape centroids is

$$W = \arg\left(W_1' + jW_2'\right), \tag{17}$$

where W_1' and W_2' are, respectively, the top and bottom halves of the W' matrix obtained by factorizing V'. Function $\arg(\cdot)$ returns the argument of each entry of the input matrix. For the purpose of this work, "The NMF MATLAB Toolbox" was used [18].

6.2. Algorithm Description

The complete clustering algorithm using SSNMF is described in Algorithm 4.

The η parameter is set to 0 in our tests, as there is no interest in constraining the W matrix. For β, the default value is 0.1.

Algorithm 4: Sparse NMF applied to clustering of trajectories.

Input: Set of trajectories, $\{\boldsymbol{\omega}^{(1)}, \ldots, \boldsymbol{\omega}^{(m)}\}$, number of desired clusters, k and η and β parameters

Output: Clustering assignments, $c^{(i)}$'s, and, if needed, centroid matrix, W

Compute V' matrix from Eq. (16).
Using "The NMF MATLAB Toolbox", find W' and H minimizing Eq. (15).
Compute W from Eq. (17). `// Compute cluster assignments`
foreach $i = 1, \ldots, m$ **do**
    $c^{(i)} = \arg\max_{j=1,\ldots,k} H_{ji}$
end

6.3. Model Selection

The MDL principle can not be directly applied to SSNMF, but other methods can be used. The elbow method works the same way as for k-means, but uses a different cost function: $J(V', W', H) = \|V' - W'H\|_F^2$. The so-called consistency method works as follows [14]:

1. For each k, compute the consistency matrix:

   $$C_k(i,j) = 1 \iff i \text{ and } j \text{ are in the same cluster}$$

2. Repeat the previous step multiple times, and compute the mean consistency matrix.

3. Finally, compute the consistency of each clustering using:

   $$\rho_k = \frac{1}{n^2} \sum_{i=1}^{m} \sum_{j=1}^{m} 4 \left(\hat{C}_k(i,j) - \frac{1}{2}\right)^2, \quad 0 \le \rho_k \le 1,$$

   and choose k where ρ_k drops.

The reasoning behind this choice for k is that, for the right k, SSNMF should be consistent, but not for higher k's. For smaller ones, it may or may not be so.

7. Results

In this section, k-means, unconstrained and constrained VMM's, and SSNMF are tested and compared, on both synthetic and real datasets.

7.1. Synthetic Datasets

The performance of the proposed algorithms is analyzed using the following two datasets:

- The roundabout dataset (Figure 5) models a roundabout where vehicles enter at the bottom and circulate counterclockwise. There are 4 possible exits, and each of these 4 clusters is composed of 20 noiseless trajectories. The

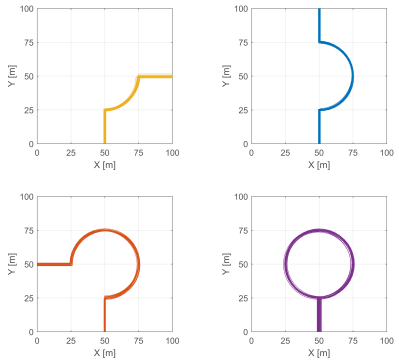


Figure 5: Roundabout synthetic dataset and its 4 clusters.

radius and the center of the roundabout were randomly generated for each trajectory, making their shapes a little different.

- The circles dataset (Figure 6) shows 100 circular trajectories, in 4 equal size clusters. The top two clusters correspond to objects circulating counterclockwise, while the bottom two, to objects circulating clockwise. The radius and center of each trajectory change between samples, and each trajectory starts at a random orientation in the interval $[0, 2\pi[$. To ignore the changes in orientation, we cluster the differences between angles of the tangents, instead of the angles themselves. This way, there are two distinguishable clusters in the dataset, composed by the top and bottom circles.

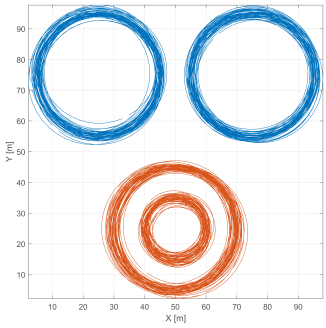
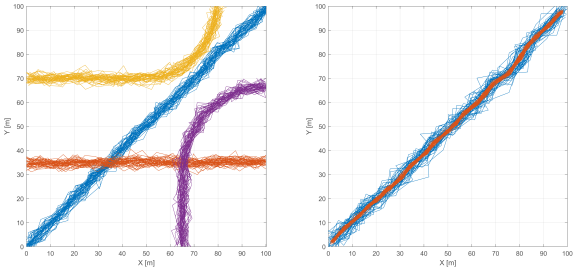


Figure 6: Circles dataset and its 2 clusters. The two top circles rotate counterclockwise, while the bottom two rotate clockwise.

- The Noisy dataset (Figure 7(a)) was designed for testing the effect of noise on the performance of the algorithms. This dataset has 4 clusters, with 50 trajectories each, and is corrupted by zero mean Gaussian noise, with a standard deviation of 1.5 m.
- The Concentration dataset (Figure 7(b)) was created to illustrate the advantages of probabilistic modeling (VMM's) over the other techniques. This dataset contains 100 trajectories, equally distributed between the two clusters, and was generated by the same mean trajectory, but with different variances (standard deviations of 0.5 m and 2 m).



(a) Noisy Tracks synthetic dataset, with 4 clusters.

(b) Concentration dataset and its 2 clusters.

Figure 7: Noisy datasets.

For the noiseless and Concentration datasets, no smoothing was used. For the Noisy dataset, several possible values for p were tested. In the noiseless datasets, 50 features were used, and 30 in the other two. Each of the 4 clustering algorithms was run on these datasets and given the right number of clusters. This was repeated 1000 times, and the percentage of trials the correct clustering was found is shown in Table 1.

Table 1: Percentage of times the optimal assignment was achieved out of 1000 trials in the synthetic datasets. The '*' indicate numerical issues.

Dataset	k-means	unconstrained VMM	constrained VMM	SSNMF
Roundabout	96.7%	96.7%	96.7%	100.0%
Circles	100.0%	100.0%*	100.0%*	100.0%
Noisy ($p = 1$)	58.1%	58.4%	58.4%	42.6%
Noisy ($p = 0.5$)	58.8%	59.0%	59.0%	64.3%
Noisy ($p = 0.1$)	65.4%	65.4%	65.4%	83.6%
Noisy ($p = 0.01$)	82.4%	82.4%	82.4%	92.4%
Noisy ($p = 1e-5$)	99.1%	99.1%	99.1%	35.7%
Noisy ($p = 0$)	99.7%	99.7%	99.7%	34.6%
Concentration	0.0%	68.9%	79.4%	0.0%

In both noiseless datasets, the true clusterings were found most of the time. There were some numerical issues with the circles dataset, as the trajectories are too similar. To avoid this, a prior with $c = -R_0 = 5e - 5$ was introduced and used in this and all the following tests. In the Noisy dataset, it was observed that below a certain level of smoothing, all algorithms performed better with the decrease in p. Stronger smoothing led to poorer results for SSNMF, but better for the other methods. It is important to note, however, that shape information is lost with too much smoothing, so it is not advised. Overall, the best results were obtained for

$p = 0.01$. In the Concentration dataset, only the VMM's were able to find the right clusters, as expected. It is interesting to note that the constrained version outperformed the unconstrained one, probably due to the latter having too many degrees of freedom for the available dataset.

7.2. Influence of β on SSNMF Results

In the previous tests, the β parameter of SSNMF was set to 0.1. To determine its effect on the results, a parametric study was conducted, using the Noisy dataset. With $p = 0.01$, β was varied, and the tests were repeated 100 times. Table 2 summarizes the results. For $\beta \in [0.1, 0.5]$, performance is approximately constant, thus justifying the choice of β used. Low and high β's give poorer results.

Table 2: Percentage of times the optimal assignment was achieved out of 100 trials for various values for β.

β	0	10^{-3}	0.01	0.1	0.2	0.5	1	2	5	10
Results	1%	32%	92%	94%	92%	91%	82%	63%	61%	63%

7.3. Model Selection on the Synthetic Datasets

In the previous tests, the algorithms were given the correct number of clusters. Since this is not the case in real problems, we now evaluate the model selection criteria. For this, k was varied in the range $1, \ldots, 10$ and the tests were repeated 20 times for each k, to avoid local optima, for each dataset and algorithm. A parametric study of the influence of p on model selection was also performed. Results are shown in tables 3 and 4.

In the noiseless datasets, the elbow method (using k-means and SSNMF) found the right number of clusters. MDL minimization using an unconstrained VMM also found the right k in these cases and in the Concentration dataset. In the noisy dataset, only MDL minimization using an unconstrained VMM found the true k, except for very small p's. This is because too much smoothing makes local patterns emerge, and also causes higher likelihoods. The other two criteria, MDL minimization using constrained VMM's and consistency performed poorly. The former performed poorly because the VMM uses very few parameters, and so too complex models are not penalized enough. The latter, on the other hand, performed badly because SSNMF was consistent even for values of k higher than the true one.

7.4. Real Datasets

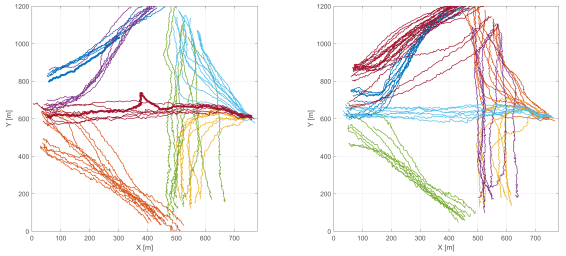
The campus dataset is shown in Fig. 8, and is composed of 134 trajectories. Since the number of clusters is unknown, the methods for model selection were executed 20 times for each k between 1 and 20. By *trial-and-error*, we set $p = 0.001$ and 100 features were used.

Table 3: Results of automatic model selection criteria.

Method	Roundabout	Circles	Concentration
Unconstrained VMM	4	2	4
Constrained VMM	10	2	5
k-means	4	2	–
SSNMF (elbow/consistency)	4/5	2/3	–/–

Table 4: Influence of p on model selection in the noisy tracks dataset. The true number of clusters is 4 and the '*' indicates no clear k could be chosen.

Algorithm	$p = 1$	$p = 0.5$	$p = 0.1$	$p = 0.01$	$p = 10^{-5}$	$p = 0$
Unconstrained VMM	4	4	4	4	9	7
Constrained VMM	7	5	5	7	10	10
k-means	4	4/5*	4/5*	4/5*	4	4
SSNMF (elbow method)	3	3	3	3/4*	4	2
SSNMF (consistency)	3	3	3	3	2	2



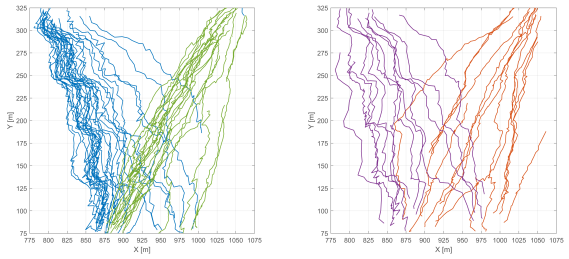
(a) 7 of the 14 clusters found in the IST Campus dataset.

(b) The other 7 clusters.

Figure 8: The 14 clusters found in IST Campus dataset. They are separated over 2 plots for facility of visualization.

Only the MDL minimization using the unconstrained VMM, which found the 14 clusters shown in Fig. 8, gave meaningful results. For having few parameters, MDL minimization with the constrained VMM chooses the biggest k tested. Using the elbow method, on the other hand, no elbow was found for k-means, and only a "weak" elbow was found at $k = 3$ using SSNMF, but it gives meaningless clusters. SSNMF was highly consistent for all k's, and the only sharp decrease happened at $k = 4$, which gives poor clusters.

The staircase dataset is shown in Fig. 9. It shows a series of people going up and down the stairs, tracked from two different views, totaling 90 different trajectories. By trial an error, $p = 0.01$ was found, and 50 features were used. The number of clusters was varied from 1 to 12 and the tests were repeated 20 times, to avoid bad local optima. In this case, all methods except MDL minimization using the constrained VMM found 4 clusters, shown in Fig. 9. This criteria failed for not penalizing enough more complex models.



(a) 2 clusters of people going down the stairs.

(b) The other 2 clusters, with people going up the stairs.

Figure 9: Clusters found in the Staircase dataset.

8. Conclusions

In this work, we have presented a novel approach to trajectory clustering, based on shape, and 3 algorithms were developed for this purposed. These methods were then tested on both synthetic and real datasets. In both cases, all algorithms could find good cluster assignments when given the desired number of clusters. With model selection, the results differed much between techniques. Overall, the best method was MDL minimization using unconstrained VMM's, which found good clusters most of the time.

Possible future work includes an extension of this methodology to 3-dimensional trajectories and an adaptation of this methodology to clustering parts of trajectories, for example.

References

[1] D. Arthur and S. Vassilvitskii. K-means++: The Advantages of Careful Seeding. In *Proceedings of the Eighteenth Annual ACM-SIAM Symposium on Discrete Algorithms*, SODA '07, pages 1027–1035, Philadelphia, PA, USA, 2007. Society for Industrial and Applied Mathematics.

[2] R. A. Becker, R. Caceres, K. Hanson, J. M. Loh, S. Urbanek, A. Varshavsky, and C. Volinsky. Route Classification Using Cellular Handoff Patterns. In *Proceedings of the 13th International Conference on Ubiquitous Computing*, UbiComp '11, pages 123–132, New York, NY, USA, 2011. ACM.

[3] D. Brillinger, H. K. Preisler, A. A. Ager, and J. G. Kie. An exploratory data analysis (EDA) of paths of moving animals. In *Journal of Statistical Planning and Inference*, pages 43–63, May 2004.

[4] A. Caner Türkmen. A Review of Nonnegative Matrix Factorization Methods for Clustering. *ArXiv e-prints*, July 2015.

[5] C. De Boor. *A Practical Guide to Splines*. Applied Mathematical Sciences. Springer, Berlin, 2001.

[6] M. Demirbas, C. Rudra, A. Rudra, and M. A. Bayir. iMAP: Indirect measurement of air pollution with cellphones. In *Pervasive Computing and Communications, 2009. PerCom 2009. IEEE International Conference on*, pages 1–6, March 2009.

[7] J. B. Elsner. Tracking Hurricanes. *Bulletin of the American Meteorological Society*, 84(3):353–356, 2003.

[8] N. Ferreira, J. T. Klosowski, C. E. Scheidegger, and C. T. Silva. Vector Field k-Means: Clustering Trajectories by Fitting Multiple Vector Fields. *CoRR*, abs/1208.5801, 2012.

[9] D. Fink. A compendium of conjugate priors, 1997.

[10] Z. Fu, W. Hu, and T. Tan. Similarity based vehicle trajectory clustering and anomaly detection. In *IEEE International Conference on Image Processing 2005*, volume 2, pages II–602–5, Sept 2005.

[11] S. Gaffney and P. Smyth. Trajectory Clustering with Mixtures of Regression Models. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '99, pages 63–72, New York, NY, USA, 1999. ACM.

[12] W. Hu, D. Xie, Z. Fu, W. Zeng, and S. Maybank. Semantic-Based Surveillance Video Retrieval. *IEEE Transactions on Image Processing*, 16(4):1168–1181, April 2007.

[13] A. K. Jain. Data Clustering: 50 Years Beyond K-means. *Pattern Recognition Letters*, 31(8):651–666, Jun 2010.

[14] J. Kim and H. Park. Sparse Nonnegative Matrix Factorization for Clustering. 2008.

[15] D. D. Lee and H. S. Seung. Learning the parts of objects by non-negative matrix factorization. *Nature*, 401:788–791, Oct 1999.

[16] J. Lee, J. Han, and K. Whang. Trajectory Clustering: A Partition-and-Group Framework. In *In SIGMOD*, pages 593–604, 2007.

[17] T. C. M. Lee. An Introduction to Coding Theory and the Two-Part Minimum Description Length Principle. *International Statistical Review*, 69(2):169–183, 2001.

[18] Y. Li and A. Ngom. The non-negative matrix factorization toolbox for biological data mining. *Source Code for Biology and Medicine*, 8(1):1–15, 2013.

[19] K. V. Mardia, G. Hughes, C. C. Taylor, and H. Singh. A Multivariate Von Mises Distribution with Applications to Bioinformatics. *The Canadian Journal of Statistics / La Revue Canadienne de Statistique*, 36(1):99–109, 2008.

[20] J. C. Nascimento, M. A. T. Figueiredo, and J. S. Marques. Activity Recognition Using a Mixture of Vector Fields. *IEEE Transactions on Image Processing*, 22(5):1712–1725, May 2013.

[21] A. Ng. Lecture Notes. CS229: Machine Learning. *Stanford University*, 2003.

[22] M. Pierobon, M. Marcon, A. Sarti, and S. Tubaro. Clustering of human actions using invariant body shape descriptor and dynamic time warping. In *IEEE Conference on Advanced Video and Signal Based Surveillance, 2005.*, pages 22–27, Sept 2005.

[23] M. Vlachos, G. Kollios, and D. Gunopulos. Discovering similar multidimensional trajectories. In *Data Engineering, 2002. Proceedings. 18th International Conference on*, pages 673–684, 2002.

[24] J. Wei, H. Yu, J. H. Chen, and K. L. Ma. Parallel clustering for visualizing large scientific line data. In *Large Data Analysis and Visualization (LDAV), 2011 IEEE Symposium on*, pages 47–55, Oct 2011.