Universiteit Leiden
The Netherlands

Opleiding Informatica

Identifying anomalous trajectories
from bird tracking data sets.

Thomas Vink

Supervisors:
Dr. Mitra Baratchi, Dr. Yali Si

BACHELOR THESIS

Leiden Institute of Advanced Computer Science (LIACS)
www.liacs.leidenuniv.nl

27/01/2021

Abstract

Trajectories of birds collected with GPS sensors can be compared with each other to find trajectory patterns. There are a number of well-known algorithms available for finding similar trajectories that can be used to find common fly patterns of birds with their seasonal migration. However, uncommon fly patterns can also be investigated to find reasons why these fly patterns are uncommon. To find uncommon fly patterns, we investigated similarity algorithms to detect anomalous trajectories and chose *SSPD* algorithm due to its capability to define a distance-based similarity on the whole structure of the two trajectories. This similarity algorithm will not only consider the route a bird took, but it will also look at the positions a bird has flown. We have also chosen another algorithm, called *t2vec*, as it offers calculation of the vector representation of trajectories. These vector representation consist of numbers that do not define a geo-spatial point, unlike GPS representation, and can then be used for clustering, such that potential different clusters of birds can be made. A cluster of trajectories is a group of different trajectories that all represent the same route a bird has flown. The clusters from both *SSPD* and *t2vec*, as well as the computational time complexity of two approaches, were compared to identify which algorithm is the most effective for anomalous trajectories. Our results indicated that *t2vec* should be preferred over *SSPD* because *t2vec* was able to obtain similar clusters when applied to a down-sampled dataset. The Euclidean distance for *t2vec* between the clusters of the down-sampled dataset and the complete dataset was much lower than the Euclidean distance for *SSPD*. Both the computational time and the run time was lower for *t2vec* compared to *SSPD*. Further research with *t2vec* was performed to investigate the underlying reason for anomalous trajectories by narrowing down to several possible causes (including the failures of the tracking devices, or behavior changes). Further research must be done to be able to conclude such hypotheses.

Contents

1 Introduction

1.1 behavior of animals

Animals are complex creatures. Some tend to behave in groups, others tend to act on their own. The behavior of animals is mostly linked to a means of survival [sea20]. A flock of birds tends to have more chance to make their full migration and a herd of deer sticks together such that it will be harder for any predator to eat one of the deer. However, there are always some animals that deviate from the whole group. They act on their own or fly to a different place compared to the group of the same species. These animals show anomalous behavior. While behavior of groups of animals is observed and understood correctly, people have much yet to learn about animals with anomalous behavior. A better understanding of these animals can provide more information about the life of certain animals. Some might not be able to complete the migration, while other animals might find a more suitable environment for survival. A change in climate or available resources could also be a reason why animals show anomalous behavior. Therefore, we can conduct more research on environmental changes if we can find animals with anomalous behavior.

1.2 Manipulation of animal trajectories

Animal trajectories themselves are recorded through the format called Global Positioning System (GPS). A trajectory may consist of a lot of data points. Most trajectories are recorded through a tracking device that records the position during different time frames for a certain period. A dedicated algorithm is necessary to do any analysis on different trajectories. Quite some different comparison algorithms exist that define the similarity between two trajectories. These algorithms can cluster big data and categorize similarity groups, such as Longest Common Sub Sequence (*LCSS*) [MGD02] or Symmetrized Segment-Path Distance (*SSPD*) [PBJMF15]. *LCSS* will try to find the longest path of two trajectories that are the same. Figure 1 shows two possible bird trajectories and their data points. *LCSS* would not be able to find a part of the trajectories that are the same, even though they start and end at the same position. *SSPD* is able to calculate a distance between these two trajectories. The distance between two bird trajectories is the total amount of difference. The bigger the distance, the more both trajectories differ from each other. The distance in *SSPD* depends on the amount of data points as well as the structure of the two trajectories. Most similarity algorithms will calculate such a distance of two trajectories. If that number is fairly small, these two trajectories belong to the same group of trajectories. If there are many trajectories in the dataset, the algorithm will divide all the trajectories into groups of trajectories. The groups can tell us which route is the most popular route, or what route is the most popular route to take.

Birds that take a completely different route compared to the most popular route are considered anomalous trajectories. They are hard to distinguish from other trajectories. While two trajectories may start and end on the same position, the route to the endpoint may differ to a large extent from the other trajectory. The bird trajectories in Figure 1 show the same start-and-end point, although it is undefined which route is the most natural route. The natural route is the most popular route that a flock of birds takes for their migration. Because there is no way to define

the natural route in Figure 1, the difference between anomalous and similar trajectories remains unclear. Similarity algorithms could still be useful to detect possible anomalous trajectories. While similarity algorithms cannot define an anomalous trajectory, recent studies show that the use of a Recurrent Neural Network (*RNN*) may work better for anomaly detection [LZC+18] [MMSY18]. *RNN* is able to find common patterns of trajectories better than other similarity algorithms, even when the data is heavily altered. Standard rules for anomaly detection do not exist. However, there are some general consensuses [LKGZ20] in which trajectories could be considered anomalous such as the direction of the trajectory and the structure of the trajectory as a whole. If a bird flew towards the East while the flock of birds flew towards the West, then the bird that flew to the East has flown an anomalous trajectory. Algorithms that calculate a distance between two trajectories use a highest threshold to define the border between similar or not. This is the maximum amount of distance between two trajectories. Trajectories that have a distance number higher than the threshold could be considered anomalous. However, similar trajectories could still contain outliers in their data points. Outliers are points in the dataset that has nothing to do with the trajectory. Outliers could be the case of a faulty GPS signal. Most algorithms are capable to detect outliers. However, the distance number may become higher due to these outliers.

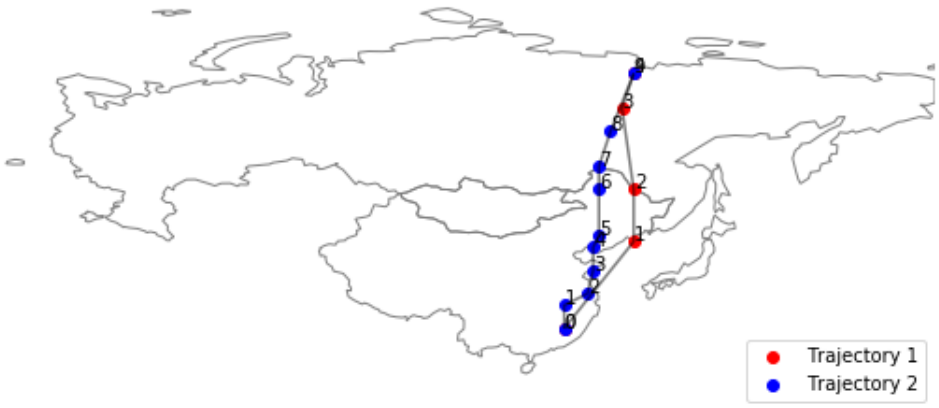

Figure 1: Two trajectories that both have the same start-and-end point.

1.3 Thesis contribution

Anomaly detection with animal tracking datasets could be very useful to obtain information about the life of the animals. To do so, we first study several different similarity algorithms to find the most appropriate algorithms to use for clustering trajectories. We will choose two algorithms that use different methods to obtain similar trajectories. After both algorithms made clusters of trajectories, they will be compared to find the highest accuracy between the two algorithms. The most useful of the two will be used for further anomaly detection on modifications of the used bird tracking dataset. A modified dataset could lead to better detection of anomalous trajectories. Therefore, we pose two research questions that are focused on GPS data.

- Which algorithm will provide the highest accuracy for detecting similar trajectories?
- When is a trajectory anomalous and how will an anomalous trajectory be useful?

1.4 Thesis overview

An explanation will be given inside Section 2 providing necessary details about trajectories, algorithms, and anomalies. The algorithms will be thoroughly explained inside Section 4 to share an understanding of both algorithms used. Section 3 will cover all research done by other authors and what was learned from this research. A complete overview of the results is given inside Section 5, including all useful graphs inside Section B. Some results will be taken out of Section B and inserted in Section 5, to assist the explanation of the results. Section A contains numerical results of the clustering of the Fall and Spring datasets separated. All conclusions are made inside Section 6. All created code that is used in this research can be found on https://github.com/ThootjeV/anomaly-detection.

2 Definitions

2.1 Trajectories

First of all, it is of great importance to define a trajectory exactly. If we do not understand what a trajectory is, we are unable to do any research on trajectories. Trajectories are known to have several data points in a sequenced order. Those data points consist of a Longitude and Latitude. Latitude is known as the displacement on earth compared to the equatorial line. The equatorial line is at 0 degrees, while the North Pole is 90 degrees and the South Pole is -90 degrees. The latitude can also be seen as the vertical place on a world map. Longitude is then the horizontal displacement compared to the prime meridian. The prime meridian is at 0 degrees, while the 180^{th} meridian is at 180 degrees to the west (-180) and east (+180) of the prime meridian.

A combination of latitude and longitude can determine the exact place in the world. To get a trajectory, one would need to record more than one combination of latitude and longitude, together with the times that those were measured. If all data points have a designated time, then a complete trajectory is measured for an investigated object. If not all data points have a designated time, or some data points are missing, the trajectory is known as incomplete. With an incomplete trajectory, it is difficult to determine what happened between two known data points. A more formal definition of a trajectory is described in Equation 1:

$$T = ((t_{x,1}, t_{y,1}), ..., (t_{x,n}, t_{y,n})) \tag{1}$$

where T is defined as a trajectory T, n is the size, x is the longitude and y is the latitude. There may be other data files that contain data points similar to the ones stated above. However, these are not considered trajectories because they are not in a sequenced order. They are merely individual data points that can be registered by other means than a GPS signal.

In this research, we will use a dataset that is stored on Movebank [oAB20]. The dataset contains all data points from trajectories of birds. It is collected by Si et al. [SXX+18]. The dataset itself contains 80 trajectories of birds. All trajectories are from the white-fronted goose in Asian territory. Every data point has a timestamp to see which date and time the data point was collected through GPS.

2.2 Similarity algorithms

Comparing two trajectories can be quite difficult. As seen in Figure 1, trajectories might differ a lot while both trajectories end and start at the same position. Therefore, most comparison algorithms will calculate a resulting distance between the two trajectories. An algorithm that calculates a certain distance between two trajectories is called a similarity algorithm. Every similarity algorithm calculates distances slightly differently. Most algorithms use a calculation measure called Euclidean distance. Euclidean distance is calculated through Equation 2.

$$d(\mathbf{p}, \mathbf{q}) = d(\mathbf{q}, \mathbf{p}) = \sqrt{(q_x - p_x)^2 + (q_y - p_y)^2} \tag{2}$$

Where d is the distance between two points $\boldsymbol{p}$ and $\boldsymbol{q}$, and the points are in a 2-dimensional space

with x longitude of a certain point and y the latitude of a certain point. Similarity algorithms have different approaches for the use of Euclidean distance. The two most notable ways to calculate a resulting distance are warping-based distance algorithms and shape-based distance algorithms. Warping-based distance algorithms will calculate a distance through similar patterns of trajectories. Shape-based distance algorithms will calculate a distance through the data points and the structure of two trajectories.

2.3 Deep Representation learning

Some more recent algorithms make use of a technique called deep learning to predict if an unknown trajectory is similar to one of the trajectories the algorithm has learned. Deep representation learning itself is a learning method for an algorithm to identify features from data input (i.e., the longitude and latitude features from the bird dataset). Through several layers, which are called hidden layers, the data input will be transformed into a certain pattern. This pattern can then be compared to known patterns that the algorithm has learned. To learn from data input, the algorithm will make use of a training dataset. This is a certain percentage of all the data input. Widely used percentages are 80% for the training set and 20% for the test set.

There are different types of deep representation learning. *T2vec*, for example, uses the unsupervised type of deep representation learning. It is a type that uses unlabeled data. Unlabeled data does not contain identifying classifications. With unlabeled data, the algorithm can easily use different sorts of data input. Unsupervised learning has to learn potential patterns inside the training dataset by itself. The test dataset is then used to test if the recognized patterns are correct patterns on the input data. This is in contrast to supervised learning [HKH+13], which uses labeled data to essentially learn by an already established example of patterns. Unsupervised learning is a bit slower compared to supervised learning. It will, however, be more useful than supervised learning [RA13] with anomaly detection, because unsupervised learning algorithms are useful for the categorization of different unknown patterns. Supervised learning algorithms, on the other hand, will only recognize known patterns to optimize already existing patterns.

The learning process of unsupervised learning results in a vector representation of the trajectories. The vector representation is 1-dimensional. Different deep representation learning algorithms calculate the vector representation in different ways. Therefore, a vector representation is hard to understand. There is not one globally accepted representation, unlike GPS data. To be able to show results, we will use IDs such that every vector representation is unique and can be analyzed when the algorithm created vectors.

2.4 Hierarchical clustering

Algorithms that compare two trajectories will not be completely useful without a way to cluster. Through clustering, we can group trajectories with high similarity. Most clustering methods cannot cluster the data only with the results. These methods need every distance that is calculated with a similarity algorithm to be able to define clusters. Therefore, the comparison of two trajectories must take place inside the clustering algorithm. The clustering must be done on all trajectories. Since the clusters must contain all similar trajectories, the used clustering algorithm must run until

trajectories or groups of trajectories are not similar. The used clustering algorithm in this paper is called Hierarchical Clustering (*HCA*) [BH12] [PB15]. This method will run through iterations until the specified similarity threshold cannot be obtained with all the sub-clusters. The threshold is the highest number the comparison result may be to be considered similar. The threshold is calculated through Equation 3.

$$Threshold = \frac{sum(results)}{length(results)} \tag{3}$$

Where *results* is a list containing all the different similarity values for all pair of points from trajectories of the first iteration.

HCA is a process in which the algorithm will begin with all trajectories in a separate cluster. It will take two clusters or trajectories and their similarity value, compare the sub-clusters and see if the result is higher or lower than the highest threshold. This threshold must be specified by the user. Whenever the result is lower than the threshold, the algorithm will combine the two sub-clusters into a new cluster. This process will go on until all clusters and sub-clusters have a similarity value higher than the threshold.

2.5 Comparison between similarity algorithms

The two algorithms that will be used must be compared with each other to see which of the two performs better. The comparison will be based on Experiment 2 by Li et al. [LZC+18]. This experiment takes down-sampled datasets and compares the clustering of the down-sampled datasets with the clustering of the complete dataset. It is not possible to calculate the accuracy of anomalous trajectories since bird trajectories do not have a clear distinction between anomalous and regular trajectories. Therefore, the comparison of the two algorithms will be done through a variation of Equation 2. Equation 2 will calculate a Euclidean distance for two points. However, if all the points in two trajectories are calculated with Equation 2, the sum of all the Euclidean distances will be the total Euclidean distance between the two trajectories. The distance between two trajectories is defined by Equation 4.

$$d(\mathbf{T}_1, \mathbf{T}_2) = \sum_{i=1}^{N} d_i(\mathbf{p}_i, \mathbf{q}_i) \tag{4}$$

Where N is the number of data points from the smallest trajectory, $d(\mathbf{p}_i, \mathbf{q}_i)$ the distance between two data points, $\mathbf{p}_i$ an individual data point of trajectory 1 and $\mathbf{q}_i$ an individual data point of trajectory 2.

To be able to correctly use Equation 4 for algorithm comparison, the dataset will be down-sampled to obtain another dataset. This can be achieved through a function that will randomly remove data points based on the amount of down-sampling. A pseudo-code can be seen in Algorithm 1.

Algorithm 1: Down-sampling algorithm

Input A trajectory with several data points, down-sampling rate
Output The same trajectory with fewer data points

```
for i = 2, ... , trajectory-size-1 do
    random-number ← generate random number between 0 and 1
    if random-number bigger than down-sampling rate then
        new-trajectory.append(data point i)
    end if
end for
```

With two datasets, we can calculate the distance of the clusters made with the down-sampled dataset and the complete dataset. Similarity algorithms, as well as deep representation learning, will group similar trajectories. The down-sampled dataset still contains the same trajectories, albeit with fewer data points. A distance will be calculated between the trajectory categories of the complete dataset and the down-sampled dataset, for both algorithms. A lower distance between categories from the complete dataset and the down-sampled dataset would result in a more accurate algorithm.

3 Related Work

3.1 Warping-based distance algorithms

Throughout the years, there are already lots of algorithms established that need trajectories as their input. All of them do something unique that others do not do. However, it is possible to categorize lots of algorithms in certain groups. One such group is called the warping-based distance group. These algorithms try to find patterns with all the different data points of two trajectories. Warping distance algorithms not only compare the same index of a trajectory point blue(i.e., the first or the second data point), but will also compare all different indices of the compared trajectories with each other. This is achieved by creating a grid cell of all the different combinations of indices. Warping distance is essential when the trajectories are not the same length. The most notable are Longest Common Sub Sequence (*LCSS*) [MGD02], Edit Distance on Real sequences (*EDR*) [LMV05], Edit distance with Real Penalty (*ERP*) [LR04] and Dynamic Time Wrapping (*DTW*) [BKHC98].

While most warping-based distance algorithms use Euclidean distance, they differ in their approach. *LCSS* takes a look at two trajectories and their timestamps, to see if any sub-sets of these two trajectories are the same. The algorithm will search for the longest of those subsets. *LCSS* is mostly useful if all the trajectories have the same timestamps and all of the timestamp intervals are the same. *EDR*, on the other hand, looks at the differences between all data points. *LCSS* and *ERP* are completely different warping-based distance algorithms. *DTW* and *ERP* will put additional weight on the differences and similarities between all data points from two trajectories. With this weight, *DTW* and *ERP* can be considered to be more accurate than *LCSS* and *EDR*. The computational cost that they take is $O(n^2)$ for all algorithms.

3.2 Shape-based distance algorithms

A whole different approach for the comparison of two trajectories is the shape-based distance algorithm. Compared to warping distance, shape distance will calculate a distance over two whole trajectories and their geometries. Notable algorithms that use a shape-based distance are *Hausdorff* [F.14] and *Fréchet* [M.06]. *Hausdorff* takes the greatest distance from a data point in Trajectory 1 that is smaller or equal to all data points in Trajectory 2 and does this for every data point in Trajectory 1. Then, *Hausdorff* has a list of distances from every data point in Trajectory 1. The final distance of *Hausdorff* is decided by the smallest distance in this list that is greater or equal to all other distances. However, *Hausdorff* will also calculate distances from data points to line segments. Therefore, the algorithm will obtain a distance that is based on the geometry of two trajectories and will not take any timestamps into account. *Fréchet* distance is calculated through similarity between curve segments of trajectories. Both have a computational cost of $O(n^2)$ and perform exceptionally well when time series will not be taken into account.

Another shape-based distance algorithm is Symmetrized Segment-Path Distance (*SSPD*). Compared to *Hausdorff* and *Fréchet*, *SSPD* will look at all possible sub-trajectories that two trajectories can have. Therefore, the trajectories as a whole will also be used when calculating the final distance. The distances that *SSPD* calculates are called the *Segment-Path* distances. If the segment of Trajectory 1 is part of the path of Trajectory 2, then the distance will be zero. To make the algorithm symmetric,

the algorithm will take the mean of all *Segment-Path* distances from both trajectories. *SSPD* is time-insensitive, compares the whole trajectory, and compares the physical distance between two trajectories. This has not been achieved with all the aforementioned algorithms [PBJMF15]. This leads to the decision to use *SSPD* as the traditional similarity algorithm.

To use *SSPD*, a combination of different algorithms must be made. There are sufficient Python libraries that work with trajectory data [JR19] [PBJMF15]. These source codes are easy to re-implement for computation of distances. During the calculation of all the distances, we will also make use of a clustering algorithm that can cluster data with only their distances as a parameter [A.19].

3.3 Deep Representation Learning algorithms

In more recent years, using a deep representation learning algorithm has seen to have many advantages over the more traditional similarity functions [MMSY18]. Some algorithms have proven to be more useful with trajectory clustering than the more traditional distance calculation, using traffic data [HKH+13] [MMSY18]. These algorithms work with labeled GPS data, so their source code would not be useful. However, their conclusions offered a more in-depth look at what the difference could be between a more traditional algorithm and a deep representation learning algorithm. Deep representation learning algorithms are capable to distinguish similar trajectories better than more traditional algorithms, when the labeled GPS data tells which trajectories are anomalous. Even if the dataset is configured, the deep representation learning algorithm can correctly distinguish similar trajectories. Therefore, if we have labeled trajectories, *RNN* algorithms perform exceptionally well for anomaly detection.

Another deep representation learning algorithm is the trajectory to vector (*t2vec*) algorithm, composed by Li et al. [LZC+18]. This algorithm will use a training dataset to create vector representations of the input data through a recurrent neural network, and then it will use the test dataset to see if the vector representation can be transformed to a GPS-represented trajectory. This results in a *best-model* of vectors that represent the trajectories. The test dataset will also be used to see how well the algorithm has learned from the training dataset, through various modifications of the dataset. Li et al. found that the computational cost of *t2vec* is $O(n + |v|)$ with $|v|$ being the length of the vector. While there may be a plethora of other deep representation learning algorithms to use, *t2vec* is resistant to possible outlier data points and low-sampling rates.

4 Methodology

4.1 Choice of algorithms

Several algorithms may be useful for this research. We have chosen a shape-based distance, *SSPD*, for the more traditional algorithm. We have chosen *t2vec* as an example of an RNN based algorithm. With animal trajectory data, the general geometry will be very important for anomaly detection. Therefore, a shape-based distance algorithm is more useful than a warping-based distance algorithm. We also experimented with a warping-based distance algorithm, *LCSS*. However, we were not able to cluster the trajectories with *LCSS*. The similarity numbers of all trajectories were zero. *LCSS* can only calculate a distance if the timestamps and their intervals are the same. However, every timestamp is different as well as the timestamp interval from all data points. *SSPD* has been chosen specifically because the algorithm is relatively new [PBJMF15], and it will look into subsets of trajectories as well as the whole trajectory itself. Li et al. [LZC+18] used the *t2vec* algorithm with a taxi trajectory dataset. Therefore, the use of *t2vec* for anomaly detection of bird trajectories may be useful. *t2vec* also has the potential for further research in trajectory detection.

4.2 The data

Before the algorithms will be used, one must have a thorough understanding of the data that is provided. First of all, the GPS data consists of several data points for one species and all organisms have their unique ID. The species provided is the white-fronted goose, in a time frame from 2015 until 2019. All individual organisms had their own tracker for a certain period in time. The length of this period depends on the duration of the tracking and the system that the tracking company used. The duration ranges from several days to several months. While GPS data may not always be accurate, the algorithms consider the trajectories as complete and without missing values. Otherwise, the algorithms are not able to compare trajectories.

Furthermore, the timestamps that are provided in the GPS data are not consistent. While some data points have data every 20 minutes, others have time jumps for more than a few hours. Therefore, the algorithms used will not take timestamps into account or consider them all equally divided. While this decision may seem to be a bit dubious, the first implementation of *LCSS* did take all different timestamps into account. The resulted similarities for all of the trajectories were zero. However, it may be interesting to divide the data by timeframe, such that the spring migration and fall migration will be separated in different files. This is useful because most birds will have a northern migration during spring and a southern migration during fall.

The data collected by GPS is inside a CSV file format, organized alphabetically by the bird ID [SXX+18]. The sum of all unique animal IDs is 80, which means that the algorithms will work with a maximum of 80 trajectories. Reading the data from the CSV file can be done through the pandas library [pdt20]. However, the type of all longitudes and latitudes are in string format. This must be correctly changed to floats when using the data.

4.3 *SSPD*

The implementation of *SSPD* is used inside a hierarchical cluster algorithm. First of all, all trajectories are considered separate clusters. Then, the first round of comparisons is done through all 80 clusters, comparing them one by one. All subsequent rounds will then compare a cluster of 1 or more trajectories with another cluster until none of the comparisons hit below the threshold number. Comparing clusters is done through *SSPD*, an implemented function inside the python library trajectory-distance by Guillouet et al. [PBJMF15].

The implemented function will take two trajectories or clusters and run the *SSPD* algorithm through these two trajectories. Figure 2 shows two example trajectories. When *SSPD* is run with these trajectories, it will calculate distances for all possible combinations of sub-trajectories. Trajectory 1 and Trajectory 2 both contain five data points, such that 1.0 and 2.0 are the start data points of the trajectories. 1.4 and 2.4 are the end data points of the trajectories. *SSPD* will calculate a distance for all individual points, and all of the possible combinations of points. This means that *SSPD* will also calculate a distance between the sub-trajectories 1.0 until 1.3 and 2.0 until 2.3. Once all distances are calculated, they are summed. The resulting value is called D_{SPD}. *SPD* means Segment-Path Distance. The final distance will be calculated by Equation 5.

$$D_{SSPD} = \frac{D_{SPD}(T_1, T_2) + D_{SPD}(T_2, T_1)}{2} \tag{5}$$

If there is any sub-trajectory of Trajectory 1 that is a sub-trajectory of Trajectory 2, the distance will always be zero. A proof of this is given by Besse et al. [PBJMF15].

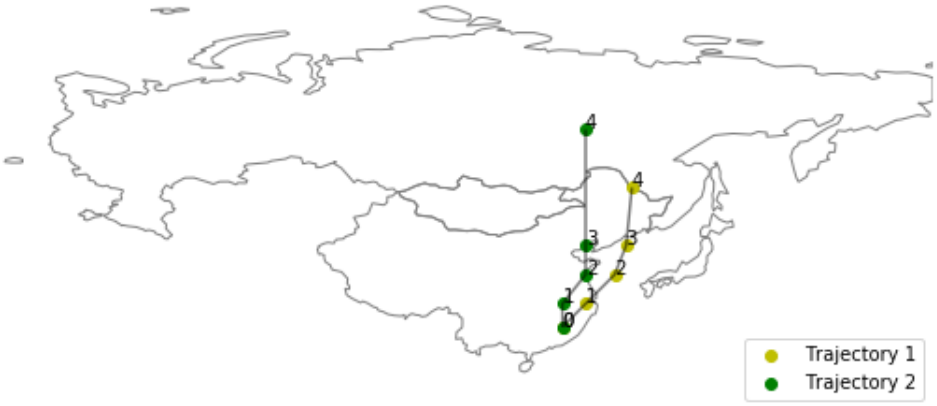

Figure 2: Two trajectories with different mid and endpoints.

4.4 t2vec

The base algorithm of *t2vec* is defined by Li et al. [LZC+18]. While Li et al. use taxi data from the city of Porto, the algorithm can be used on every other data since it is a neural network based on unlabeled data. Therefore, the GPS data from birds must be preprocessed into the same format as the Porto dataset. The format used is a polyline format. This means that all longitude and latitude points must be on the same list, where a combination of longitude and lat-

itude is one element in this list. As stated before, the dataset is complete when no points are missing.

T2vec uses a special encoder-decoder model for representation learning. The encoder-decoder model takes a trajectory $\mathbf{p}$ and will build a vector v from the trajectory $\mathbf{p}$. All sequential information inside $\mathbf{p}$ still exist inside the newly made vector v. The decoder will be able to convert vector v back into a trajectory $\mathbf{q}$, where $\mathbf{q}$ is a representation trajectory of trajectory $\mathbf{p}$. To learn trajectory representations using a *RNN*, the components of a trajectory $\mathbf{p}$ will first be changed to an embedded vector (i.e., all embedded vectors contain the necessary information to represent trajectory $\mathbf{p}$) before all embedded vectors will be converted to a single vector v. To learn a reliable representation through these vectors, a conditional probability function (Equation 6) will be used to calculate the most reliable route R from an input trajectory $\mathbf{p}$ and a vector v.

$$\mathbb{P}(p_t = u | p_{1:t-1}, v) = \frac{exp(W_u^\top h_t)}{\sum_{v \in V} exp(W_v^\top h_t)} \tag{6}$$

Here, p_t is the t-th trajectory data point, h_t is the t-th hidden state in the *RNN* which is used to decode the vector $\mathbf{v}$, $W^\top$ is the projection matrix that can output a positional value with the corresponding hidden state and the input vector $\mathbf{v}$, V is the complete Vocabulary that contains all the possible latitude and longitude positions and u is the u-th row.

A route R is a collection of data points that indicates the travel of the bird, defined by Li et al. [LZC+18]. Due to the probability function, several different routes R can be generated from the input trajectory and the corresponding vector. All these routes must be considered similar to be able to cluster trajectories. To address this issue, the coordinates of R are changed into cells. While coordinates have a continuous range, cells have a discrete range. A discrete range is useful when we want to cluster trajectories. All components from several generated trajectories with route R will be put inside the same cells.

5 Experiments

5.1 Individual algorithm results

5.1.1 Data manipulation

Before both algorithms can be run, the order of the data must be changed to fit both algorithms. For both, the individual data points must be grouped by their respective animal ID. For a faster configuration, the Python library package Traja [JR19] can be used to do manipulations on all trajectories in a small number of functions. Then, the data must be saved to a different CSV file format to ensure the same type of columns that are used with the initial *t2vec* algorithm. To run the *SSPD* algorithm, the saved data from Traja needs to be changed to a NumPy [Oli06] array. This is the only accepted format inside the *SSPD* algorithm, as this algorithm uses NumPy-only functions. Figure 11 shows every trajectory from the used dataset. While one may see several outlying data points, there is no certainty as to which trajectories may be anomalous or not.

5.1.2 *SSPD* results

First of all, the threshold for clustering similar trajectories must be decided. This is done by running the *SSPD* algorithm through all the trajectories once. To calculate the threshold, Equation 3 will be used. All the distances are summed up and divided by the total number of distances, which is 6400. This resulted in a threshold around 2, which is rounded down to be sure the algorithm takes only the most similar trajectories. The maximum and minimum length of a trajectory must also be set in *SSPD*, as well as the number of trajectories. Table 1 contains the numbers that have been used for the *SSPD* algorithm with the bird dataset.

Hyperparameters *SSPD*	Bird dataset settings
Number of trajectories	80
Minimum length	100
Maximum length	10,000
Threshold	2

Table 1: Parameter settings for *SSPD*

Secondly, the clustering algorithm needs to run past all possible clusters. Most of the clustering algorithms themselves consist of code that will combine the two clusters when the threshold criteria are met. Then inside this algorithm, the distances between the two clusters are calculated through the *SSPD* function. This process took roughly 3 whole days. It is, therefore, useful to note that this algorithm is expensive. blueThe clustering resulted in eight clusters, which can be seen from Figure 13 until Figure 27. The algorithm is built in such a way that individual trajectory numbers are lost. This could be a problem if the algorithm will be used for further research since all the clusters now only contain a lot of different data points.

As a final step, some clusters look quite vague. It would be better to take an average trajectory and visualize that. While Figure 14 until Figure 28 are not one specific trajectory each, they show a possible average of the trajectories. To calculate every average trajectory, we followed the following

process: For every latitude point, all longitudes of that point will be put inside one list. Afterward, for every latitude point, the mean of all the longitudes will be calculated. The result will be a list of 0 or 1 longitude, where the list ranges between 0 and 90.

Figure 13 until Figure 27 show that *SSPD* also clusters on different routes. When Figure 13 is compared with 17, they both contain a large number of similar data points. However, the distance calculated through *SSPD* also depends on northern or southern migration. Therefore, northern and southern migration is of influence with this algorithm. Normally during spring, the birds will migrate to the northern region. This can be related to what is presented in Figure 23. Normally during fall, on the other hand, birds will migrate to the southern region. Figure 13 shows the migration to the south. Since this algorithm does not take timestamps into account, the results only show northern and southern migration. From these results, there is no way to deduct whether a cluster contains only fall or only spring migrations. Every figure has a representative start point for all the trajectories inside the cluster, to illustrate where the trajectories began.

SSPD also separates trajectories on their endpoints. The figures show that the data is separated by whether or not the birds reached northern Siberia. Figure 27 shows a northern migration from China to Siberia. A comparison with Figure 23 shows that the birds in Figure 23 only reached up to northern China instead. This could mean a failed GPS signal, the death of those birds, or anomalous trajectories.

The biggest advantage of *SSPD* is the separation of data through all the possible trajectories. Many or few data points, northern or southern migration, and completed or unfinished migration all define the distance value for a trajectory or sub-cluster. However, *SSPD* does not separate anomalous trajectories enough to be able to tell which trajectories are anomalous.

5.1.3 *T2vec* results

Several hyper-parameters need to be set to use this algorithm. These hyper-parameters need to be changed to use your dataset. The dataset of Porto only focuses on the city of Porto, whereas our dataset will contain data points all over Asia. The dataset of Porto also contains an enormous set of small taxi trajectories, while the GPS dataset contains a small number of trajectories that are enormous. The algorithm will read a CSV file. However, any trajectory that is too big to fit in the CSV file will be seen as incomplete. The implementation cannot be changed, as the processed format is strict and vulnerable to errors. As a consequence, the only data that can be used will be the accepted data by the algorithm. This means that our implementation can only use 38 of all 80 trajectories that are available in the complete dataset. This must be taken into consideration when comparing both algorithms. A possible solution to this problem would involve dividing the dataset into separate datasets or reducing the amount of time interval points.

Before running the algorithm, the hyper-parameters need to be set for the bird GPS data. The minimum and maximum longitude and latitude will be set appropriately to fit every possible point around Asia since the data ranges from China to Russia. See Table 2 for a comparison between the Porto parameters and the Asia parameters. Li et al. reads a CSV file and copy the content into an HDF5 file. An HDF5 file is called a Hierarchical Data Format. Li et al. use this type of format

as their primary format. Only the content that can be copied to the HDF5 file will be used. As explained before, not all trajectories can be used. Therefore, only 38 of all 80 trajectories will be converted into the HDF5 format. To identify these trajectories, the prepossessing step will also make a new CSV file which only contains the trajectories that are used. After the conversion, the data inside the HDF5 file will be separated for a training dataset and a test dataset.

Hyperparameters *t2vec*	Porto settings	Asia settings
Minimum Longitude	-8.735152	0.15
Minimum Latitude	40.953673	0.15
Maximum Longitude	-8.156309	171.44
Maximum Latitude	41.307945	80.12

Table 2: Parameter settings for Porto and Asia datasets

If all prepossessing steps are done, the neural network must train to learn and recognize vector patterns. This will be done through the created training dataset inside the HDF5 file. The test dataset is used to calculate the amount of error that the vector represented trajectory has compared to the actual GPS-represented trajectory. The whole dataset is split by given numbers. A ratio of 80% of the dataset is used to train, whereas 20% of the dataset is tested. This is a widely used ratio for any learning algorithm. The training was done for about twelve hours, and every thousand trained iterations are saved inside a checkpoint file. The best checkpoint file will be saved as a file named the best-model. After twelve hours, the error margin for every thousand iterations was the same. The error margin is the difference between the output of the vector representation and the actual input. Therefore, the best-model contains the best possible vector representation of the training dataset. To link every vector representation to their respective GPS representation, another CSV file will be created which contains a combination of these two representations. This CSV file will also be used to visualize the clusters in the final step. Since the training dataset is used in the training process, the CSV file contains only those vector representations of the trained trajectories. The vector representation of the test data will not be stored, because it was only needed to obtain the best possible vector representations of the trajectories.

The vector representation can now be used to calculate distances and to cluster the different trajectories. Like with *SSPD*, the threshold must be decided first. The distance calculation will be done with DTW, such that all the different vectors of two trajectories can be compared. The threshold calculated through Equation 3 resulted in around 25000. This might seem like a very high number. However, it will not matter, since all distances are scaled to high numbers around the threshold. If all distances in *results* in Equation 3 are high, the scaling of the threshold is done automatically. The clustering algorithm itself will be the same as with *SSPD*. This process took around one hour, which is less expensive compared to *SSPD*. The results can be seen from Figure 29 until Figure 45. The average trajectories of the clusters can be seen from Figure 47 until Figure 55. This average is calculated in the same way that has been done with the clusters from *SSPD*.

While *t2vec* does not separate quite as rigorously as *SSPD*, trajectories found in Figure 35 as well as Figure 45 could be considered anomalous trajectories. These trajectories are separated from other clusters, meaning that this algorithm is able to detect possible anomalous trajectories.

5.2 Evaluating the performance of algorithms

When the two algorithms have finished clustering and calculating distances between all the trajectories, the two algorithms need to be compared to see which of them is more effective. *t2vec* has several comparison methods that have proven to be the most effective [SPD+15] [HKH+13]. The results already show a high advantage to *t2vec* over more traditional algorithms, like *LCSS* or *EDR*. These results have been obtained with the Porto dataset. However, *t2vec* uses datasets that contain more than 1 million trajectories. One method, proposed by Li et al. [LZC+18] (Experiment 1), takes a random sample varying between 10.000 and 100.000 trajectories. Since our datasets only contain 80 different trajectories, this method will not be used. The other methods that are used by Li et al. will change the data itself. We will use the method that will remove a certain percentage of data points from the data (Experiment 2), to see if the clustering stays the same with a smaller-sized trajectory. We will use various percentages, to find whether or not the algorithms can group the same trajectories inside one cluster. The computational cost will also be taken into account.

5.3 Results of the evaluation

5.3.1 Manipulation of dataset

SSPD is able to run on all 80 trajectories, while *t2vec* was only able to run on 38 full trajectories of the 80 trajectories in total. Figure 12 shows all trajectories from the dataset that is used initially with *t2vec*. Therefore, both results will always differ from each other. To compare which algorithm could detect anomalous trajectories more easily, the algorithms must be run with the same dataset. We will use a re-sampled dataset for the comparison. The complete dataset contains 80 trajectories with some trajectories that contain an enormous amount of data points. Most of these trajectories cannot be used with the *t2vec* algorithm. Therefore, the dataset must be resized to use all 80 trajectories with both algorithms. The time between two data points ranges between every few minutes and several hours. This wide range of time frames will be reduced such that every data point will at least have 2 hours between two data points. The re-sized dataset contains less than 30% of all the data points in the whole dataset, without losing much information. *t2vec* is able to run all 80 trajectories with the re-sampled dataset.

We will also create subsets of the re-sampled dataset to use Equation 4. Different reduction percentages will be used to compare both algorithms. If the algorithms ran with the reduced datasets, we will compare the clusters made by both algorithms for the re-sampled dataset and the reduced dataset. Equation 4 is then used to calculate the distance between all clusters from one algorithm with the down-sampled dataset and the re-sampled dataset. If the distance is low, the algorithm was able to correctly cluster the same trajectories, even if the dataset is reduced. If the distance is high, the algorithm made completely different clusters and is therefore vulnerable to changes in the dataset. Since only the dataset changes, the setup to run both algorithms is the same as in Sections 5.1.2 and 5.1.3.

5.3.2 *SSPD* results

The results of the 38 trajectories dataset used with *SSPD* can be found in Figure 56 until Figure 58. While the results are completely different from the results in Section 5.1.2, the figures show that the algorithm did do the same separations. It did not take into account whether or not the migration was completed and whether there were many or only a few data points. Some birds finished or started their migration in northern China (Figure 57), while other birds finished or started their migration in Siberia (Figure 56). However, we cannot deduce which cluster has the most natural route for the birds. We also cannot deduce any anomalous trajectory from these figures alone. *SSPD* was able to successfully cluster the trajectories, though it was not able to filter any possible anomalous trajectories. The algorithm was a whole lot faster compared to 5.1.2, with only 30 minutes of runtime before the clusters were made.

5.3.3 *T2vec* results

The results of the 38 trajectories down-sampled dataset used with *t2vec* can be found in Figure 30 until Figure 46. Most clusters have many similarities with the clusters found in Section 5.1.3. Figure 46 and Figure 40 contains data previously seen in Figure 43 and Figure 37. It is unclear why *t2vec* has filtered these trajectories into different clusters. Even if the data is manipulated, *t2vec* is able to identify possible anomalous trajectories. The algorithm was only a bit faster compared to Section 5.1.3, with 30 minutes of runtime.

5.3.4 Computational cost

The computational cost is also very important for the difference between the two algorithms. For *t2vec*, the computational cost is linear, $O(n + |v|)$. For *SSPD*, the computational cost is quadratic, $O(n^2)$. Therefore, *t2vec* is better to use. Furthermore, the total runtime of *SSPD* with the original bird dataset was three days. The runtime of *SSPD* with the re-sampled dataset was around one hour. The learning process of *t2vec* consisted of twelve hours of learning. The clustering process took another twelve hours of runtime.

5.3.5 Comparison results

We cannot compare the algorithms with the use of figures only, as these are used mainly for visualization. The Euclidean distances of the down-sampled clusters have been calculated comparing the clusters of their original counterpart. The results can be seen in Table 3. The distance numbers for *SSPD* do show that *SSPD* will treat the same trajectories differently when the dataset is down-sampled, as the distance numbers are in a high range. There will always be a difference in distance, as one dataset contains more data points than the other. However, the distances that *t2vec* was able to acquire are far more suitable for anomaly detection, since a manipulated dataset will result in almost the same clusters of trajectories.

	20%	30%	40%	50%	60%
SSPD	5229.49	3807.55	4705.17	4968.53	3777.19
t2vec	14.625	17.5	16.625	15.375	20

Table 3: Euclidean distances for *SSPD* and *t2vec*

While *SSPD* is a great algorithm to cluster several trajectories into different groups and identify their traits, *SSPD* is not useful to identify possible anomalous trajectories. The possible anomalous trajectories are still grouped into several big clusters. *T2vec* on the other hand shows clusters that could be anomalous trajectories. Therefore, *t2vec* has the biggest advantage when identifying anomalous trajectories. Similar algorithms to *SSPD* have also been compared with *t2vec* by Li et al. [LZC+18]. When identifying traits, *t2vec* has clusters with indistinct traits and therefore *SSPD* would be more useful to apply with this dataset. For our research of anomalous trajectories, however, *t2vec* will be the most useful algorithm to use.

5.4 Further results from the re-sampled dataset with *t2vec*

5.4.1 More manipulation of the dataset

The re-sampled dataset can also be filtered such that one subset only contains fall migration and another dataset spring migration. Since *t2vec* will not cluster distinct traits, there will not be a way to separate fall migration from spring migration with *t2vec*. In the data pre-processing step, we will split the data into a CSV file containing all the data points that have dates between July and December for fall migration, and a CSV file containing all the data points that have dates between January and June. While the total number of trajectories was 80, the Spring dataset contains 67 trajectories and the Fall dataset contains 70 trajectories. This is a confirmation that some trajectories were recorded in one long period where some geese made one or multiple northern and southern migrations with the recorded data.

5.4.2 Results of *t2vec* on the re-sampled dataset

The re-sampled dataset, which does not have spring and fall separated, contains all 80 trajectories. *t2vec* can now run on all 80 trajectories. The learning process took around 12 hours. The learning process only learns with 80% of all the data, since the other 20% is used as test data. Therefore, all clustered trajectories together contain 80% of the total number of trajectories inside the dataset. After obtaining the best possible vector representation, the clustering process also took around 12 hours. The results are shown in Figure 3, Figure 4, and Figure 59 until Figure 83. While there are many different clusters, the clusters themselves show that some trajectories contain possible anomalous trajectories. The average trajectories of the clusters can be seen in Figure 5, Figure 6, and Figure 60 until Figure 84. The average trajectories show that some clusters only have subtle differences in their representative trajectory. This can be seen in Figure 5 and Figure 6, where the representative trajectory only differs in north Russia. However, Figures 3 and 4 show more noticeable differences in Northern Russia. To get the best possible clusters, *t2vec* must also be run with fall and spring migration separated.

Figure 3: *t2vec* cluster 1 re-sampled

Figure 4: *t2vec* cluster 2 re-sampled

Figure 5: *t2vec* cluster 1 re-sampled average

Figure 6: *t2vec* cluster 2 re-sampled average

5.4.3 Results of *t2vec* on Spring and Fall datasets

The final results, with spring and fall migrations separated, can be seen in different figures. For the Fall-only dataset, the clusters can be seen in Figure 7, Figure 8, and from Figure 85 until Figure 95. The Spring-only dataset can be seen in Figure 9, Figure 10, and from Figure 96 until Figure 106. These results contain trajectory numbers for each cluster. Every tracked bird in each cluster for both the Spring dataset and the Fall dataset can be found in Table 4. These tables also state whether or not every bird in the cluster had data points throughout multiple years. Many trajectories are collected inside Figure 7 and Figure 9. These figures indicate that the clusters show flocks of birds migrating together in the specified season. These clusters could very well be the typical trajectory that white-fronted geese take when migrating between Russia and China and are therefore not anomalous.

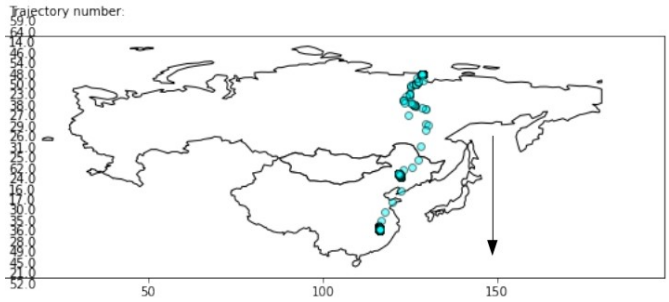

Figure 7: *t2vec* cluster 10 re-sampled fall

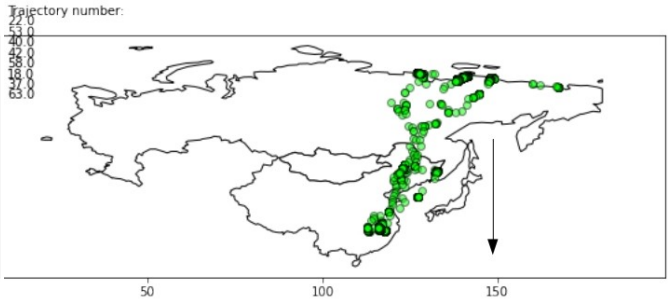

Figure 8: *t2vec* cluster 11 re-sampled fall

There are no birds that traveled in the opposite direction, as birds will have natural survival instincts. They will not travel to places where they cannot survive due to cold weather. While analyzing the clusters, there was one bird tracking company (**D**ruid) that registered their data points in reverse chronological order. This can also be seen in Table 4, as cluster 11 only contains trajectories from this company. The algorithm was not able to label them correctly, as the algorithm stated that these birds traversed in opposite directions. Therefore, most of the trajectories from the company Druid have been left out. Further manual analysis of cluster 11 (Figure 105) found out that these birds traveled northward. Some trajectories, like in Figure 88, made the full migration with very few data points. These clusters contain data points that registered only one migration. Others, like the birds in Figure 95, have fully completed their migration with a lot of data points. These clusters contain data points of multiple migrations. In Figure 103, the goose either died before it could start migrating or the GPS signal failed to register this specimen.

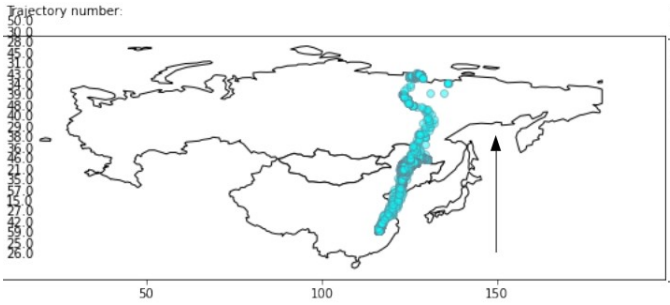

Figure 9: *t2vec* cluster 10 re-sampled spring

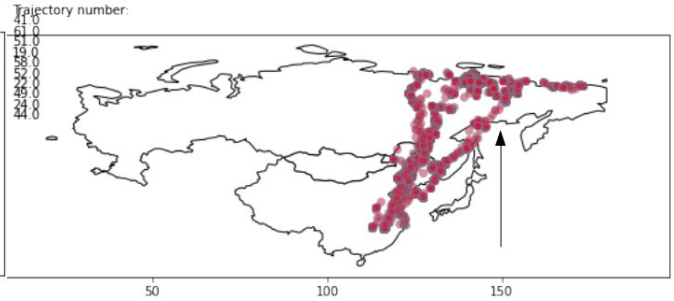

Figure 10: *t2vec* cluster 12 re-sampled spring

6 Conclusions and Further Research

6.1 Conclusion

Anomalous trajectories can be detected more easily through the vector representation of *t2vec* compared to *SSPD*. While *SSPD* is useful to find common traits for each cluster, the vector representation of *t2vec* can separate the anomalous trajectories from the typical trajectories. If the data itself is separated by season, *t2vec* can run through all the data with relative ease to see which trajectories could be anomalous. While none of the results can definitively show anomalous trajectories, *t2vec* shows a better understanding of which trajectories may be anomalous. *t2vec* is able to filter any possible anomalous trajectory from the cluster with regular trajectories, even if the data is heavily down-sampled. *t2vec* can also separate one migration with multiple migrations. These trajectories could be analyzed further to see why they differ from the typical trajectory.

6.2 Discussion

The results of the comparison between *SSPD* and *t2vec* show that *t2vec* has an overall better performance compared to *SSPD*. While *SSPD* may cluster on common features that cannot be found through *t2vec*, *t2vec* clusters all regular trajectories together. This cannot be obtained through *SSPD*. The computational cost is also in favor of *t2vec*, as *t2vec* can compute clusters much faster compared to *SSPD*. However, *t2vec* will not work if the dataset contains too many data points for one trajectory. Therefore, every dataset that will be used for *t2vec* must be re-sampled first before any computation can be done. The results of the Fall and Spring datasets indicate possible reasons why some trajectories are not clustered with other regular trajectories. The GPS signal could have failed, the bird could have died or there could be fewer resources in the environment. However, every nature-based reason behind the anomalous trajectory should be analyzed further before we can make any conclusion. *t2vec* considered every bird ID as an individual trajectory. However, Table 4 and Table 5 contain trajectories of multiple migrations. This indicates that multi-year migrations are considered as a single trajectory. We will lose information about the possible change in behavior of birds over multiple years. Therefore, a better approach would be to separate such trajectories.

6.3 Future Implementation

The *t2vec* algorithm, as well as the plots, could both be improved in future work. *T2vec* could be altered to be able to use full-fledged datasets. This would mean an altered vector length and size such that *t2vec* accepts trajectories that consist of more data points. Future plots could contain the coordinates of every point, preferably stating which data point belongs to which trajectory. Future research could also be done with other species of birds. Because *t2vec* is an unsupervised learning algorithm, the algorithm could be used for any kind of anomalous data detection throughout the whole world. *SSPD* could also be improved for possible anomaly detection. This could be achieved by defining a function that defines which trajectories could be anomalous. If we are able to obtain labeled data, a *RNN* supervised learning algorithm might find interesting patterns among the available data.

References

[A.19] Vladimirovich A. Time series hierarchical clustering using dynamic time warping in python, November 2019.

[BH12] Frank B. Baker and Lawrence J. Hubert. Measuring the power of hierarchical cluster analysis, 2012.

[BKHC98] Yi B.-K., Jagadish H., and Faloutsos C. Efficient retrieval of similar time sequences under time warping, 1998.

[F.14] Hausdorff F. Grundz uge der mengenlehre, 1914.

[HKH+13] Su H., Zheng K., Wang H., Huang J., and Zhou X. Calibrating trajectory data for similarity-based analysis, 2013.

[JR19] Shenk J. and Busche R. justinshenk/traja: v0.1.1, June 2019.

[LKGZ20] Yiding L., Zhao K., Cong G., and Bao Z. Online anomalous trajectory detection with deep generative sequence modeling, 2020.

[LMV05] Chen L., Ozsu M.T., and Oria V. Robust and fast similarity search for moving object trajectories, 2005.

[LR04] Chen L. and Ng R. On the marriage of lp-norms and edit distance, 2004.

[LZC+18] Xiucheng Li, Kaiqi Zhao, Gao Cong, Christian S. Jensen, and Wei Wei. Deep representation learning for trajectory similarity computation. In *34th IEEE International Conference on Data Engineering, ICDE 2018, Paris, France, April 16-19, 2018*, pages 617–628, 2018.

[M.06] Fréchet M. M. Sur quelques points du calcul fonctionnel, 1906.

[MGD02] Vlachos M., Kollios G., and Gunopulos D. Discovering similar multidimensional trajectories, 2002.

[MMSY18] Cong Ma, Zhenjiang Miao, Shaoyue Song, and Ming-Hsuan Yang. Detecting anomalous trajectories via recurrent neural networks, 2018.

[oAB20] Max Planck Institute of Animal Behavior. Gps bird tracking data sets. , 2020.

[Oli06] Travis E Oliphant. *A guide to NumPy*, volume 1. Trelgol Publishing USA, 2006.

[PB15] Besse P. and Guillouet B. Review perspective for distance based trajectory clustering, 2015.

[PBJMF15] Besse P., Guillouet B., Loubes J-M., and Royer F. Review perspective for distance based trajectory clustering, 2015.

[pdt20] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

[RA13] Sathya R. and Abraham A. Comparison of supervised and unsupervised learning algorithms for pattern classification, 2013.

[sea20] seaworld.org. Animal behaviour and learning. , 2020.

[SPD+15] Ranu S., Deepak P., Telang A. D., Deshpande P., and Raghavan S. Indexing and matching trajectories under inconsistent sampling rates, 2015.

[SXX+18] Yali Si, Yanjie Xu, Fei Xu, Xueyan Li, Wenyuan Zhang, Ben Wielstra, Jie Wei, Guanhua Liu, Hao Luo, John Takekawa, Sivananintha Balachandran, Tao Zhang, Willem F. de Boer, Herbert H. T. Prins, and Peng Gong. Spring migration patterns, habitat use, and stopover site protection status for two declining waterfowl species wintering in china as revealed by satellite tracking. *Ecology and Evolution*, 8(12):6280–6289, 2018.

A Tables

Table 4: Spring dataset clusters

Cluster ID	Trajectory numbers	Bird ID's	Multi-migration?	Years
1	16	E002(GWFG)	Yes	2015-2019
2	17	E003(GWFG)	Yes	2015-2019
3	20	E013(GWFG)	Yes	2016-2019
4	32	H024_B(GWFG)	Yes	2015-2016
5	33	H027_B(GWFG)	Yes	2015-2017
6	47	H088(GWFG)	Yes	2018-2019
7	53	H100(GWFG)	Yes	2018-2019
8	55	H117(GWFG)	No	2018
9	56, 11	H121(GWFG), D3725(GWFG)	No	2018, 2019
10	50, 30, 28, 45, 31, 43, 34, 39, 48, 40, 29, 38, 36, 46, 21, 35, 57, 15, 27, 42, 59, 25, 26	H093(GWFG), H018(GWFG), H013(GWFG), H081(GWFG), H021(GWFG), H074(GWFG), H047(GWFG), H066(GWFG), H089(GWFG), H067(GWFG), H016(GWFG), H064(GWFG), H057(GWFG), H086(GWFG), E017(GWFG), H050(GWFG), H124(GWFG), E001(GWFG), H006(GWFG), H072(GWFG), H127(GWFG), H003(GWFG), H005(GWFG)	No	2018, 2016, 2016, 2018, 2016, 2018, 2017, 2018, 2018, 2018-2019, 2016, 2018, 2018, 2018, 2016, 2017, 2018, 2015, 2016, 2018, 2018, 2016, 2016
11	12, 13, 14	D3731(GWFG), D3749(GWFG), D3827(GWFG)	No	2019, 2019, 2019
12	41, 61, 51, 19, 58, 52, 22, 49, 24, 44	H071(GWFG), H129(GWFG), H094(GWFG), E010(GWFG), H125(GWFG), H097(GWFG), E018(GWFG), H092(GWFG), E022(GWFG), H079(GWFG)	Yes	2018-2019, 2018-2019, 2018-2019, 2016-2017, 2018, 2018-2019, 2016-2017, 2018-2019, 2016-2017, 2018-2019
13	18, 37, 54, 60	E005(GWFG), H063(GWFG), H115(GWFG), H128(GWFG)	Yes	2015-2019, 2018-2019, 2018-2019, 2018-2019

Table 5: Fall dataset clusters

Cluster ID	Trajectory numbers	Bird ID's	Multi-migration?	Years
1	11	E003(GWFG)	Yes	2015-2019
2	15	E013(GWFG)	Yes	2016-2019
3	34	H027_B(GWFG)	Yes	2015-2016
4	39	H063(GWFG)	No	2017-2018
5	44	H071(GWFG)	Yes	2017-2019
6	47	H079(GWFG)	Yes	2017-2019
7	51	H088(GWFG)	Yes	2017-2019
8	56	H097(GWFG)	No	2017-2018
9	57	H100(GWFG)	Yes	2017-2019
10	59, 64, 14, 46, 54, 48, 50, 23, 38, 27, 29, 26, 31, 25, 62, 24, 16, 17, 30, 35, 36, 28, 49, 45, 21, 52	H117(GWFG), H127(GWFG), E010(GWFG), H074(GWFG), H093(GWFG), H081(GWFG), H087(GWFG), H003(GWFG), H061(GWFG), H009(GWFG), H016(GWFG), H007(GWFG), H020(GWFG), H006(GWFG), H124(GWFG), H005(GWFG), E015(GWFG), E017(GWFG), H018(GWFG), H047(GWFG), H050(GWFG), H013(GWFG), H086(GWFG), H072(GWFG), E021(GWFG), H089(GWFG)	No, Data starts end of year, no completed migration	2017, 2017, 2015-2016, 2017, 2017, 2017, 2017, 2015, 2018, 2015, 2015, 2015, 2015, 2015, 2017, 2015, 2015, 2015, 2015, 2016, 2015, 2015, 2017, 2017, 2015, 2019
11	22, 53, 40, 42, 58, 18, 37, 63	E022(GWFG), H092(GWFG), H064(GWFG), H066(GWFG), H115(GWFG), E018(GWFG), H057(GWFG), H125(GWFG)	Yes	2015-2016, 2017-2018, 2018, 2017, 2017-2018, 2015-2016, 2015, 2017
12	61, 41, 32	H121(GWFG), H065(GWFG), H021(GWFG)	No	2017, 2017, 2015-2016
13	12, 43, 33, 55	E005(GWFG), H067(GWFG), H024_B(GWFG), H094(GWFG)	Yes	2015-2019, 2017-2018, 2015-2016, 2017-2018

B Graphs

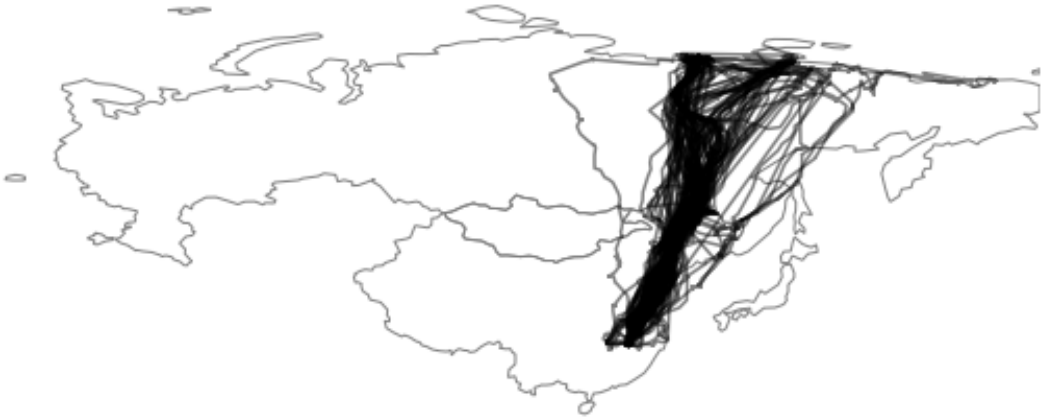

Figure 11: Line segments showing all different trajectories of the complete dataset

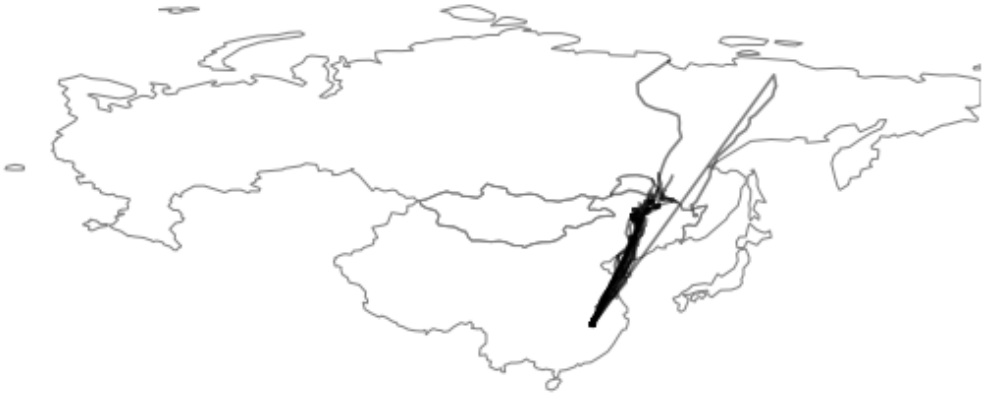

Figure 12: Line segments showing all different trajectories of the 38 trajectories dataset

Figure 13: *SSPD* cluster 1 with start point [120.6, 43.3]

Figure 14: *SSPD* cluster 1 average

Figure 15: *SSPD* cluster 2 with start point [128.2, 49.5]

Figure 16: *SSPD* cluster 2 average

Figure 17: *SSPD* cluster 3 with start point [116.2, 29.1]

Figure 18: *SSPD* cluster 3 average

Figure 19: *SSPD* cluster 4 with start point [115.9, 29.1]

Figure 20: *SSPD* cluster 4 average

Figure 21: *SSPD* cluster 5 with start point [68.9, 51.1]

Figure 22: *SSPD* cluster 5 average

Figure 23: *SSPD* cluster 6 with start point [116.4, 29.0]

Figure 24: *SSPD* cluster 6 average

Figure 25: *SSPD* cluster 7 with start point [116.4, 28.9]

Figure 26: *SSPD* cluster 7 average

Figure 27: *SSPD* cluster 8 with start point [116.1, 28.8]

Figure 28: *SSPD* cluster 8 average

Figure 29: *t2vec* cluster 1

Figure 30: *t2vec* down-sampled cluster 1

Figure 31: *t2vec* cluster 2

Figure 32: *t2vec* down-sampled cluster 2

Figure 33: *t2vec* cluster 3

Figure 34: *t2vec* down-sampled cluster 3

Figure 35: *t2vec* cluster 4

Figure 36: *t2vec* down-sampled cluster 4

Figure 37: *t2vec* cluster 5

Figure 38: *t2vec* down-sampled cluster 5

Figure 39: *t2vec* cluster 6

Figure 40: *t2vec* down-sampled cluster 6

Figure 41: *t2vec* cluster 7

Figure 42: *t2vec* down-sampled cluster 7

Figure 43: *t2vec* cluster 8

Figure 44: *t2vec* down-sampled cluster 8

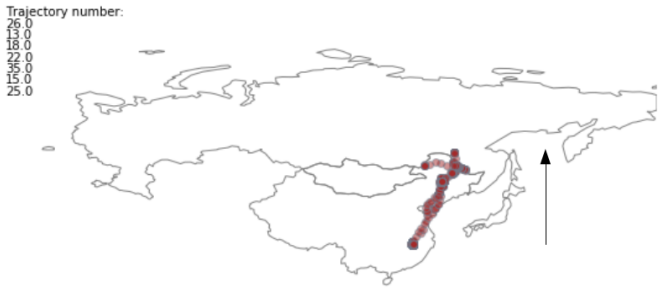

Figure 45: *t2vec* cluster 9

Figure 46: *t2vec* down-sampled cluster 9

Figure 47: *t2vec* cluster 1 average

Figure 48: *t2vec* cluster 2 average

Figure 49: *t2vec* cluster 3 average

Figure 50: *t2vec* cluster 4 average

Figure 51: *t2vec* cluster 5 average

Figure 52: *t2vec* cluster 6 average

Figure 53: *t2vec* cluster 7 average

Figure 54: *t2vec* cluster 8 average

Figure 55: *t2vec* cluster 9 average

Figure 56: *SSPD* cluster 1 with 38 trajectory dataset

Figure 57: *SSPD* cluster 2 with 38 trajectory dataset

Figure 58: *SSPD* cluster 3 with 38 trajectory dataset

Figure 59: *t2vec* cluster 3 re-sampled

Figure 60: *t2vec* cluster 3 re-sampled average

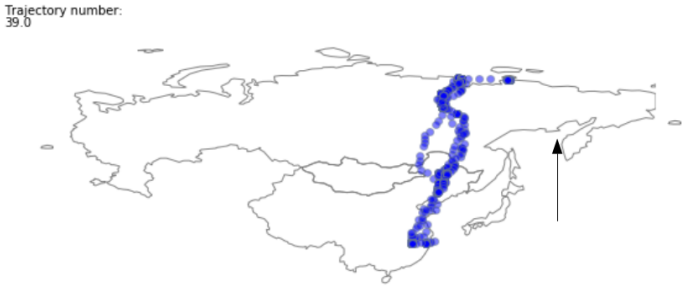

Figure 61: *t2vec* cluster 4 re-sampled

Figure 62: *t2vec* cluster 4 re-sampled average

Figure 63: *t2vec* cluster 5 re-sampled

Figure 64: *t2vec* cluster 5 re-sampled average

Figure 65: *t2vec* cluster 6 re-sampled

Figure 66: *t2vec* cluster 6 re-sampled average

Figure 67: *t2vec* cluster 7 re-sampled

Figure 68: *t2vec* cluster 7 re-sampled average

Figure 69: *t2vec* cluster 8 re-sampled

Figure 70: *t2vec* cluster 8 re-sampled average

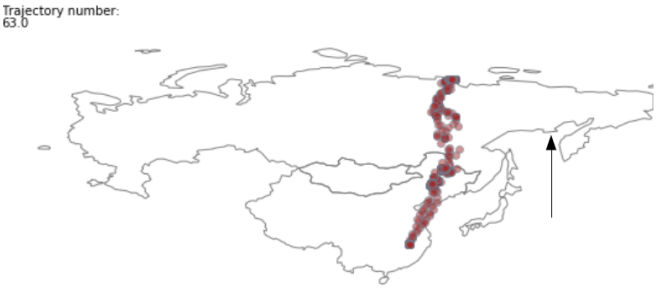

Figure 71: *t2vec* cluster 9 re-sampled

Figure 72: *t2vec* cluster 9 re-sampled average

Figure 73: *t2vec* cluster 10 re-sampled

Figure 74: *t2vec* cluster 10 re-sampled average

Figure 75: *t2vec* cluster 11 re-sampled

Figure 76: *t2vec* cluster 11 re-sampled average

Figure 77: *t2vec* cluster 12 re-sampled

Figure 78: *t2vec* cluster 12 re-sampled average

Figure 79: *t2vec* cluster 13 re-sampled

Figure 80: *t2vec* cluster 13 re-sampled average

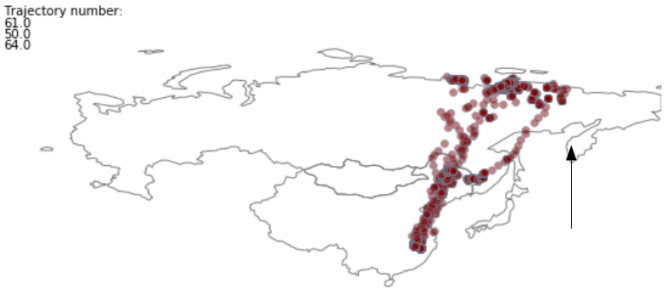

Figure 81: *t2vec* cluster 14 re-sampled

Figure 82: *t2vec* cluster 14 re-sampled average

Figure 83: *t2vec* cluster 15 re-sampled

Figure 84: *t2vec* cluster 15 re-sampled average

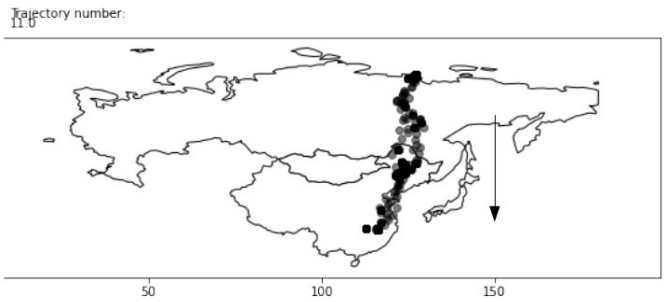

Figure 85: *t2vec* cluster 1 re-sampled fall

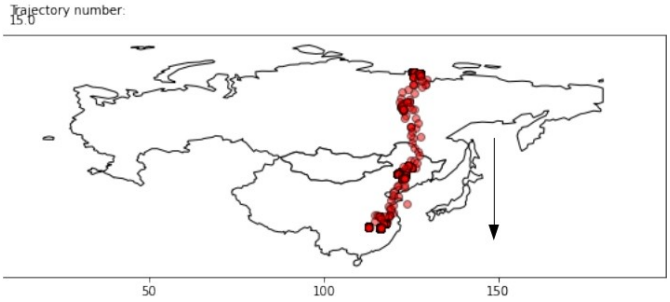

Figure 86: *t2vec* cluster 2 re-sampled fall

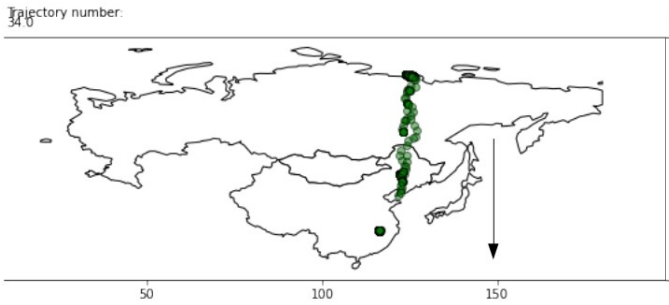

Figure 87: *t2vec* cluster 3 re-sampled fall

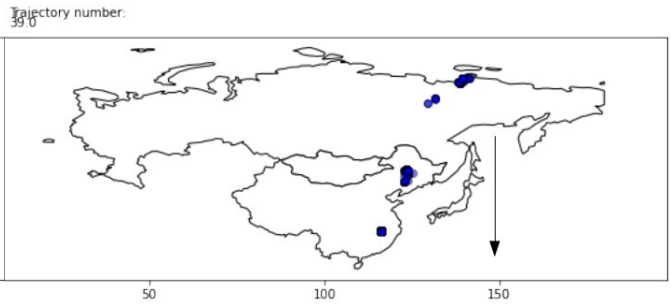

Figure 88: *t2vec* cluster 4 re-sampled fall

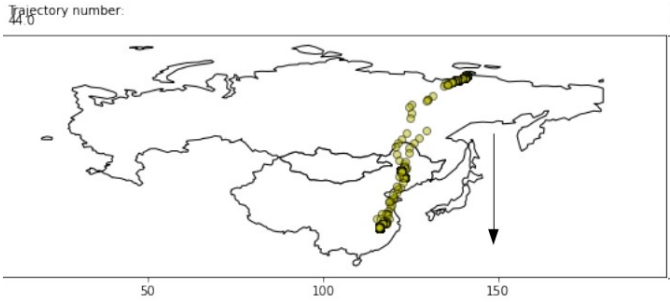

Figure 89: *t2vec* cluster 5 re-sampled fall

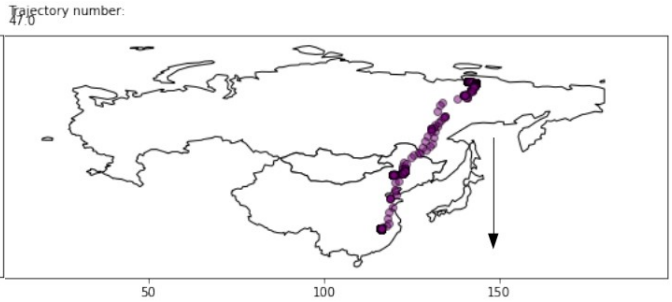

Figure 90: *t2vec* cluster 6 re-sampled fall

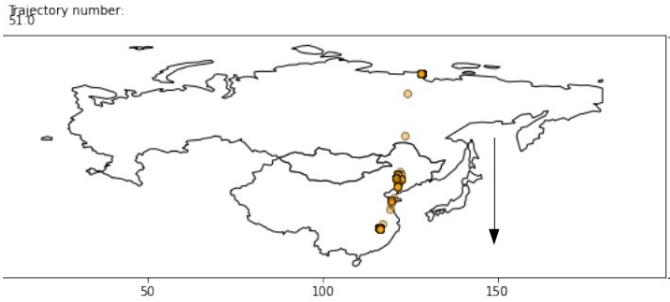

Figure 91: *t2vec* cluster 7 re-sampled fall

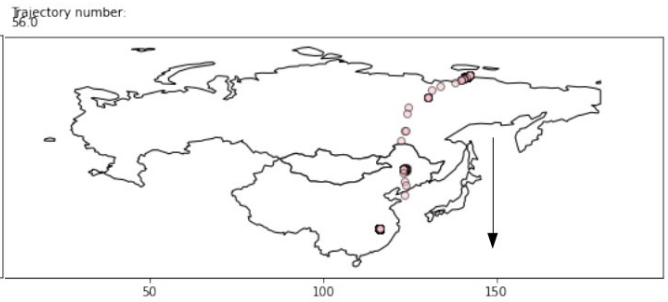

Figure 92: *t2vec* cluster 8 re-sampled fall

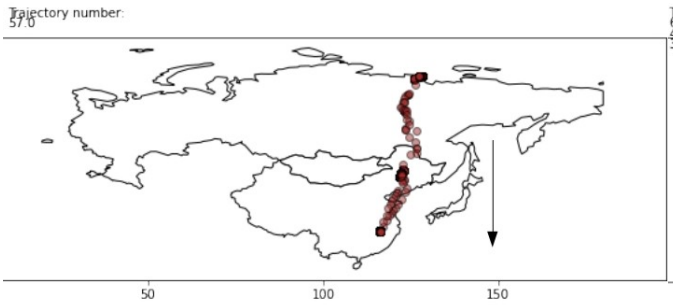

Figure 93: *t2vec* cluster 9 re-sampled fall

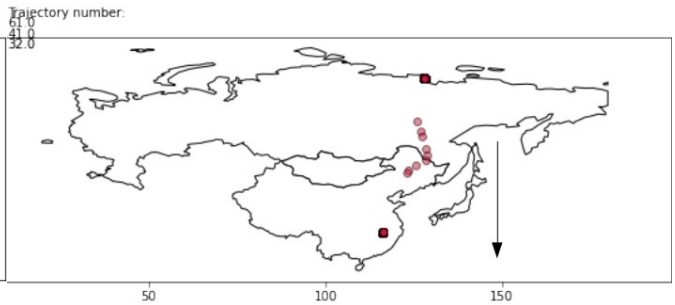

Figure 94: *t2vec* cluster 12 re-sampled fall

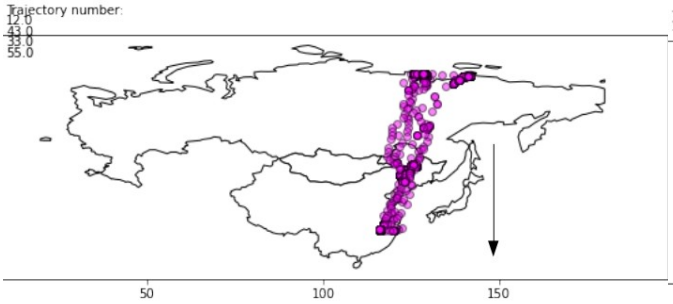

Figure 95: *t2vec* cluster 13 re-sampled fall

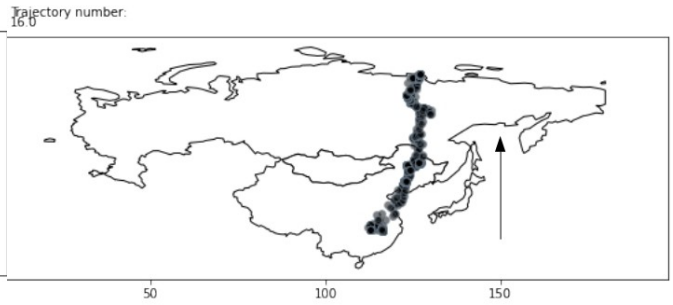

Figure 96: *t2vec* cluster 1 re-sampled spring

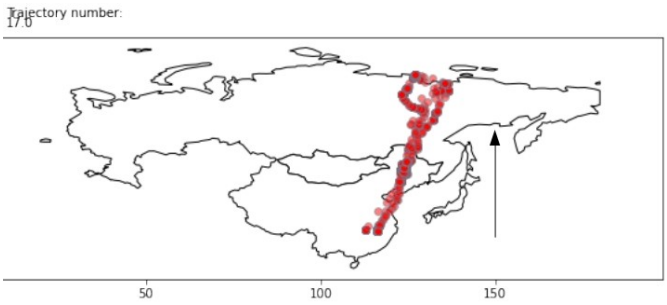

Figure 97: *t2vec* cluster 2 re-sampled spring

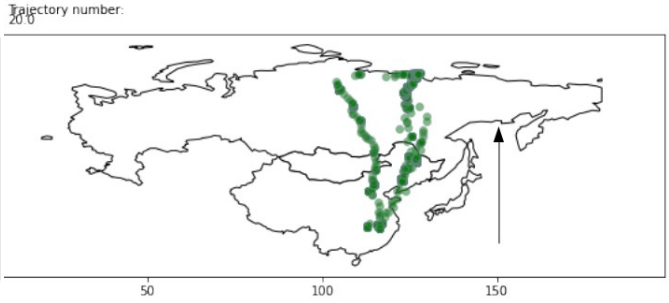

Figure 98: *t2vec* cluster 3 re-sampled spring

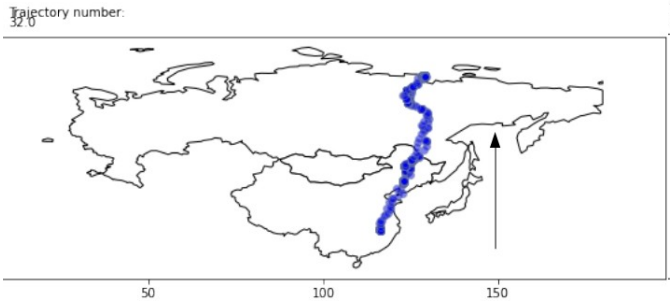

Figure 99: *t2vec* cluster 4 re-sampled spring

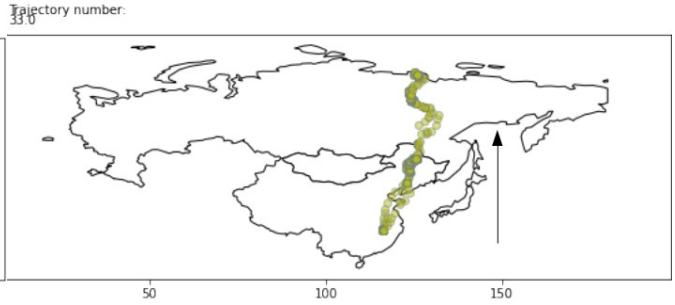

Figure 100: *t2vec* cluster 5 re-sampled spring

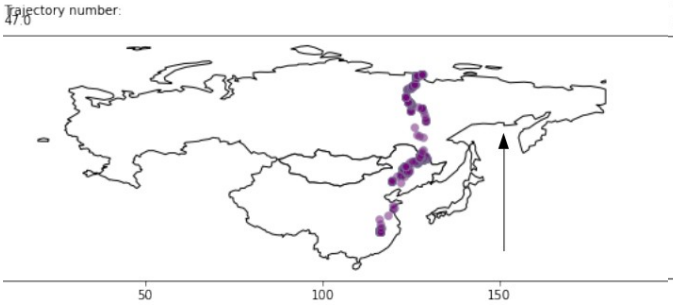

Figure 101: *t2vec* cluster 6 re-sampled spring

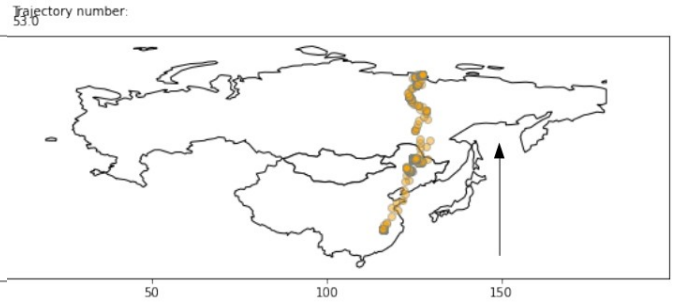

Figure 102: *t2vec* cluster 7 re-sampled spring

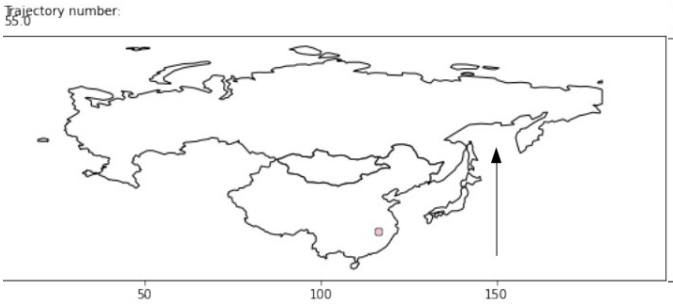

Figure 103: *t2vec* cluster 8 re-sampled spring

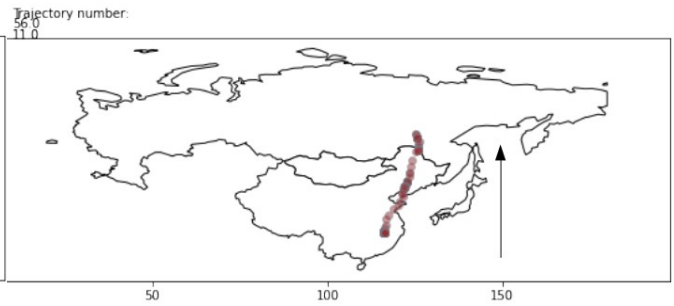

Figure 104: *t2vec* cluster 9 re-sampled spring

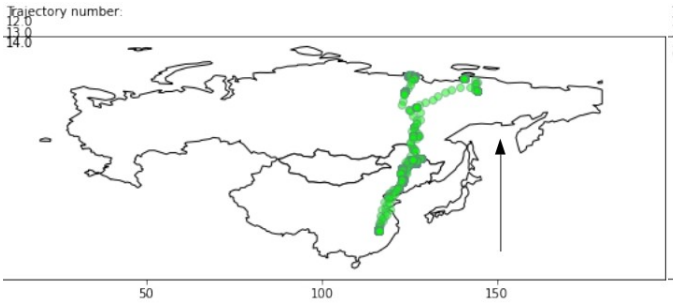

Figure 105: *t2vec* cluster 11 re-sampled spring

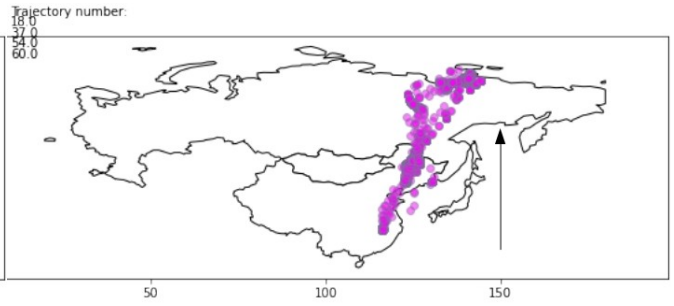

Figure 106: *t2vec* cluster 13 re-sampled spring