TOHOKU UNIVERSITY
Graduate School of Information Sciences

Trajectory Mining for Movement Ecology of Animals
(動物移動生態学のためのトラジェクトリマイニング)

A dissertation submitted to the department of
System Information Sciences in partial fulfillment of
the requirements for the degree of

Doctor of Philosophy
in
Information Sciences

by

Ilya SADEGHI ARDAKANI

July 05, 2019

Trajectory Mining for Movement Ecology of Animals

Ilya SADEGHI ARDAKANI

Abstract

This work for the most part involves applications of data science and trajectory data mining in movement ecology of animal. The main objective is to present methods and techniques that could be applied in modeling or extracting information from animal movement in spatial, spectral and temporal domains at large scales. A small part of this work is also allocated to applications of computer science and vision in data acquisition and visualization of movement. In Chapter 2, a multi-stage motion based stereo tracking method presented for non-invasive tracking of elusive animals like bats in their natural habitats. Since imaging these objects in their environments requires near infrared cameras with maximum aperture to collect maximum allowable light, images often lack sharpness. In addition, they also possess very low observable visual features. Thereby, to overcome these challenges, a method proposed which estimates the moving components of the scene and based on their dynamics and a prior knowledge about the dynamics of the target objects, it is able to identify the desired targets and reconstruct their 3D trajectories. It is demonstrated that the proposed method could provide reasonable estimates of the trajectories given the poor image conditions. This however comes at a considerable computational cost which remains as a follow-up objective. In Chapter 3, the focus moves towards modeling latent states of animal movement using their geospatial trajectory data. In this chapter, objective is to propose methods which could extract information about behavioral trends from animal trajectories at large scales. In other words, rather than focusing on an individual, the movement of masses like genders, colonies or whole species over long periods of time was the objective. Here, the similarities between language and navigation rooted in cognitive abilities of organisms exploited to discriminate between influence of behaviors on the trajectories using probabilistic models. In Chapter 4, the contextual semantical relationships between geospatial locations were modeled to classify behavioral responses of focal organisms. These contextual relationships could disentangle large groups of trajectories based on the underlying states which influenced the arrangement of trajectory points. This information is also embedded in numerical representations of the trajectory key points. These numerical representations then could be used for modeling or classification of trajectories based on their latent states. It is experimented that utilization of these input features im-

prove the classification results comparing to other common techniques. In Chapter 5, novel approaches in modeling of the dynamics of animal movement using variants of recurrent neural networks are presented. It is shown that these powerful models are able to learn and generate animal trajectories with the given prior information about the organism itself and geospatial features of environment. In this chapter probabilistic models used to discover the similarities or disparities between latent states influencing geometry of trajectories. Finally, in Chapter 6, a software solution for analysis and visualization of animal trajectories with interactive and integrated environment data management system is proposed. This tool is to help researchers to reconstruct a more realistic environment for visualization of trajectories in order to better understand the movement and navigational behaviors. Over the course of this study, the importance and necessity of integration of environmental factors in study of animal movement become more relevant. With the proposed suite, it became easier to analyse and comprehend individual and collective movement behaviors and features particularly in case of marine avians. After all, this work attempted to provide multiple paths forward for the researchers interested in the newly minted discipline of bio-navigation science. Topics discussed and presented in this work introduce various perspectives in data-driven ecology of animal movement. These are also compelling candidates for follow-up research in ecology of animal movement within the framework of data science.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

As mankind becomes technologically more advanced, it gains expanding capabilities to collect and analyse a greater volume of information about its environment. The significance of this data in improving, or in many cases saving, human life renders more evident as we progress in time. An obvious example is our ability to forecast severe weather systems which salvaged numerous human lives in recent decades. A major portion of the environmental data is collected via in-situ and remote sensing machinery such as ocean weather buoys and remote sensing satellites respectively. As one may imagine, there is a trade-off between the scale and resolution for each of the collection method as buoys data are local and remote data comes with limited resolution. Of course, increasing the number of sensors would alleviate the problem to some degree, but, up to a breaking point at which the navigational and logistical complexity makes it impractical and expensive to operate. A candidate solution to this problem is to employ animal kingdom as expert surveyors of the environment. This is done by large scale collection of animal movements. In return, this also supplies us with information about the possible adverse effects of human activities on other species and their ecosystems. In other words, by scientific and objective study

of animal movements, or movement ecology, mankind could obtain clues about environment and the interaction of its inhabitants with it. Another major advantage in studying animal movement is that it could guide us towards design of smart logging devices and GPS-less navigation systems which have applications in autonomous robots, logger robots and autonomous vehicles.

Animal movement data could be collected in both invasive and non-invasive manners. Invasive methods involve affixing a foreign object to the species of interest, given the assurance that it would not have any effects on their movement. In contrast, in non-invasive methods animals are observed from distance without necessity for making any contact. Each method has inherent limitations that makes them unsuitable for certain applications. For instance, in case of tracking bats, imaging is more suitable due to their small size, taking into account that the spatial boundaries of the trajectories would be significantly restricted. On the other hand, GPS sensors or telemetry devices are widely used for tracking larger birds.

Traditionally, free ranging animals' movements were documented by recapturing the tags or wires that had been attached to them. There was little to none information about the detailed path of their travel. Animal trajectories or movements are results of interaction between internal states, motion and navigation capabilities, and external factors[2]. With the introduction and recent advances in GPS sensors and telemetry devices technologies, we were enabled to design and manufacture smaller and lighter sensors yet with higher performance, storage and operation time. As a result, researchers were presented with a continuous availability of movement data from different animal species [3]. This on the other hand, poses as a new challenge in interpreting and gaining insights into the collected data [4, 5]. Therefore, new and revolutionary methods and techniques were required to

manage, analyse and visualize the collected data [6]. The outcome provides a window into the behavior of organisms, their reactions to environment stimuli and their life phases [7, 8, 9, 10, 11, 12]. It would also present us with a more accurate information about environment as reported in [13]. This study in parts, attempts to suggest solutions for challenges regarding the modeling and representation of animal movements by adopting data mining techniques.

1.1 Basic Concepts

In this section we briefly review two fundamental concepts of this study.

1.1.1 Trajectory Mining

Trajectory mining or trajectory data mining is systematic process of analyzing movement data in large volumes to discover patterns, similarities, semantics and anomalies [14, 15, 16, 17]. This process starts right after data acquisition with preprocessing and ends with knowledge discovery. Most of trajectory data mining algorithms share these steps in some form [15, 16]. Trajectory data may be collected in numerous ways from wide variety of objects. Trajectory data might be traces of objects in an image sequence, cellular tower pings, WiFi fingerprints, geo-locations of activities in social networks or GPS locations of public transport vehicles. It also could be animal or natural phenomena traces. These data sets undergo preprocessing procedures with respect to their type or the information expected to be extracted from them. Common preprocessing procedures are noise filtering, compression, key point detection and segmentation. In addition, map-matching is also a preprocessing method mainly used with vehicle trajectories on road networks. Next stage is representation of the trajectories. This is important in order to measure similarities

or disparities between trajectories which are used in indexing, clustering and classification of trajectories. In the following stages discovery and modeling are performed.

Trajectory Compression

In certain applications, not all of the sampled trajectory coordinates are significant and due to power, storage or complexity constraints they need to be discarded. For instance, while tracking an object, only a point suffices for representation of period of time in which the object was stationary. Therefore, a spatial aggregation tools should be used to reduce the number of data points in the trajectory. This compression could be done based on shape, key points and semantics along trajectories [18, 19, 20].

Trajectory Segmentation

Trajectory segmentation is to identify the composing elements of trajectories in a way that the underlying information about the trajectories are greatly rendered. In other words, these element produce a richer knowledge about the trajectories while reducing the cost and complexity of the computation. Segmentation is application oriented and could be performed on top of compression. Segmentation and compression share a few approaches as well. Like compression, segmentation could be based on geometry and shape or semantical [21, 22, 23]. It also could be temporal in non-uniformly sampled data set where, a large time interval between to data points creates segments [16].

1.1.2 Movement Ecology

According to [2], movement of an organism is driven by multi-scale spatial and temporal processes and it characterizes fate and lifestyle of each individual. It also reveals

essential information about the structure and dynamics of populations and their ecosystems. For modeling, organismal movement could be considered as integration of 4 main processes related to an individual organism interacting with external processes. These are internal state dynamics, navigation, motion and movement propagation processes. Internal states of an organism regards to the physiological and psychological states encoding objectives of an organism. Motion capacity describes the modes of movement in an organism like flying, crawling or swimming. Navigation capacity represents the ability of an organism to orient and navigate in space and time. This involves both sensory and memory apparatus. And movement propagation process is the way modes of movement rendered in environment. As further described in [2] movement can be formulated as following:

$$\mathbf{u}_{t+1} = F(\mathbf{\Omega}, \mathbf{\Phi}, \mathbf{r}_t, \mathbf{w}_t, \mathbf{u}_t), \tag{1.1}$$

where $\mathbf{\Phi}$, $\mathbf{\Omega}$, $\mathbf{r}_t$, and $\mathbf{w}_t$ represent navigation capacity, motion capacity, external factors and internal state at location $\mathbf{u}_t$ respectively. The processes mentioned earlier could be expressed in the formulation above as following:

$$\mathbf{u}_{t+1} = f_U(f_N(\mathbf{\Phi}, f_M(\mathbf{\Omega}, \mathbf{r}_t, \mathbf{w}_t, \mathbf{u}_t), \mathbf{r}_t, \mathbf{w}_t, \mathbf{u}_t)), \tag{1.2}$$

for navigation-driven movement process and for motion-driven case is written as:

$$\mathbf{u}_{t+1} = f_U(f_M(\mathbf{\Omega}, f_N(\mathbf{\Phi}, \mathbf{r}_t, \mathbf{w}_t, \mathbf{u}_t), \mathbf{r}_t, \mathbf{w}_t, \mathbf{u}_t)). \tag{1.3}$$

In movement formulation some of processes could be ignored. For instance, for an organism in a random motion or fully under external influence is as:

$$\mathbf{u}_{t+1} = f_U(f_M(\mathbf{\Omega}, \mathbf{r}_t, \mathbf{w}_t, \mathbf{u}_t)). \tag{1.4}$$

Lastly, internal states and external factors could have their own independent dynamical processes which are described as following:

$$\mathbf{r}_{t+1} = f_R(\mathbf{r}_t, \mathbf{w}_t, \mathbf{u}_t), \tag{1.5}$$

$$\mathbf{w}_{t+1} = f_W(\mathbf{r}_t, \mathbf{w}_t, \mathbf{u}_t). \tag{1.6}$$

1.2 Problems and Contributions

1.2.1 Stereo Tracking

As mentioned previously, image-based tracking is a non-invasive method for registering animal movements. This technique is mainly used where weight, size or access limitations do not allow attachment of active sensors. An application of this is in tracking of bats in low-light environments. Due to the natural color profile of bats, their image is captured against a background with high relative contrast. Utilizing forward-looking infrared (FLIR) or near-infrared (NIR) imaging devices and infrared illuminators are also admissable [24, 25, 26, 27, 28, 29]. In controlled setups in a relatively smaller flight chambers multi-view motion capture rigs with number of active or passive wing marker configurations are also used to record a more detailed flight kinematics of bats [30, 31, 32].

Since the interest here is mainly trajectories, using two imaging devices in stereo configuration suffices 3D reconstruction of bats' trajectories. However, one requirement for depth reconstruction in stereo setup is computing disparities between corresponding points in left and right view planes. It should be noted that detecting bats in single image is challenging let alone finding the correspondences between multiple views. There are two cases considered here. The first is that the constraint on non-invasive tracking could be slightly relaxed and the case that physical contact is not an option.

In the first case, a solution is affixing ultralight infrared reflector markers to the birds which appear as tiny bright blobs in images. The correspondence in multiple views is trivial for the case of a single object, multiple objects situated on different epipolar lines or beyond

a threshold for disparity. But, given conditions except the latter, it would be challenging to measure the 3rd dimension. Another challenge in tracking markers is dealing with the occlusions. Since bat flight involves large motion of the wings, and the wing spread is larger than the main body, which leads to occlusion of the marker in motion.

In the second scenario, there should be a few model-based prior assumptions imposed in order to localize the corresponding points in each frame in both view and time sequence.

Problem Summary and Contributions

In summary, the challenges regarding the stereo tracking of bats could be enumerated as following:

- Recognition of individual bats in each frame.
- Finding the correspondence between identified objects in each view and time frame sequence.
- Dealing with occlusions and missing assignments in both view and time frame sequence.
- Localizing the points of interest in the recognized segments in each frame.

This study offers the following contributions in order to tackle aforementioned problems:

- Proposing a method for 3D reconstruction of trajectories for bats with markers which deals with occlusions and missing correspondences.
- Proposing a multi-stage motion-based recognition algorithm for marker-less tracking of bats.

1.2.2 Trajectory Features

A trajectory, specifically a spatial trajectory, is a sequence of position measurements in space ordered in time. In general, in most of geospatial applications, the position measurements are in 2 dimensions, latitude and longitude, where the elevation is ignored due to relatively small variations. Here, we mainly focus on analysis and modeling for geospatial trajectories of animals.

Trajectories may be recorded in equal or variable intervals in time. In addition, they may have the same end points while being different in length. Hence, not all of the trajectory points carry the same weight of information about underlying factors generated the trajectory. This makes measuring similarities or comparison between trajectories a non-trivial task.

Commonly, prior to analysis or modeling stages in trajectory data mining, collected trajectory data undergoes a number of application oriented preprocessing procedures other than simply noise removal. This is due to the naive nature of each geospatial coordinate that does not carry semantical information on its own. Preprocessing procedures like clustering, segmentation and stay-point detection are a few to be mentioned [16, 15]. Each of these methods attempts to assign relevant feature information to points or segments in trajectory. These features to be used in later stages to model or discriminate between trajectories or their latent states [33, 22]. Design and engineering of these features and their representations remain an area of research in trajectory mining.

Trajectory segments could be considered at different scale levels both in time and spatial dimensions [2, 34] for their respective feature information. These segments could be determined in various ways. For instance, it could be based on the geometric shape of the trajectory at a point or mode of motion along a segment of the trajectory. But, there are

points in trajectories that are determinant of the corresponding segment semantics regardless of the scale. These carry special information along trajectories and are used as features for modeling the latent states that generated the trajectories in the first place. For example, trajectories containing theater locations versus ones containing location of stadiums would provide sufficient information to distinguish between the latent states of the subjects [35, 36].

A significant distinction between animal and human trajectories is that, in the latter, mostly the utility of the points along the trajectories could be determined in advance by matching them against the known landmarks. Therefore, the key points are simpler to extract. In contrast, in case of animals, utility semantics of locations are not generally available and identifying them might be indeed an objective itself. This is even more challenging in case of animals with very dynamic environment like seabirds. Thence, methods and algorithms should be devised to identify such key points. Furthermore, the representation of these key points needs to be addressed as well. For instance, Gao et al. [37] applied spatial hierarchical clustering methods to create feature sequences of land animals' trajectories. Then to compare similarities of these sequences with each other, Least Common Sub-Sequent (LCSS) and entropy methods [38, 39] were employed. There are also other approaches in which environment related features like distances to boundaries are considered [40].

Problem Summary and Contributions

In summary the persisting challenges regarding trajectory features and representations are stated as following:

- Sampling frequency and resolution of the collected data.

- Segmentation of the collected stream of spatial points into meaningful sequence of phases.
- Extracting informative features from trajectory segments.
- Efficient and compact numerical representation of trajectory features.

The corresponding contributions to present solutions for the problems above are:

- Proposing key point extraction algorithms for animal trajectories which produce multi-level features that carry semantical information for modeling navigational behaviors and responses to the environmental events.
- Proposing context-based semantical vector representations for animal trajectories

1.2.3 Trajectory Models

Animal trajectories could be modeled as a joint result of interacting processes between components related to the focal organisms and external factors [2]. These components are related to internal states, navigation capabilities, motion characteristics, movement path and lastly environmental factors. Internal states could be physiological and neurological states affecting movements of the organism. Therefore, it is possible to infer information about those states given trajectories. The same stands for navigation apparatus and motion machineries. The question becomes how one would model these processes. Movement consist of multiple levels of spatiotemporal scale. Evident effects of each aforementioned component varies at each level. For instance, effects of motion characteristics and movement propagation process are more distinguishable in finer scales while navigation and internal states prevail in larger scales.

With reference to the framework introduced in [2], the aforementioned components are originated from addressing essential questions like: why move? how to move? when to move? what are the consequences of movement in terms of ecology? However, answering these key questions brings about following major challenges in analysis and modeling of organismal movement. One is, to produce multilevel spatiotemporal scales of movement phases which consist of canonical activity modes [41]. The other is attributed to the environment in which, the organisms reside. Environmental factors certainly have considerable effects on all components of movement ecology. For instance, vegetation and winds effectively adjust movement paths of animals in their habitat [13, 34, 42, 43].

Here, we point out some notable and relatable approaches in movement ecology literature which attempted to tackle the mentioned challenges. Analysis of Variance (ANOVA), and other statistical methods are commonly used for modelling animal movements. In [44, 45] ANOVA and general linear models were utilized to study gender segregation of a species of deer based on their response to habitat alterations. Statistical methods are also applied to analyse the behavioral trends, the correlation between environment and foraging locations, gender specific foraging strategies and the effects of ocean currents and compass orientation on foraging and migratory behaviors in seabirds [12, 46, 10, 11, 46, 47]. Brillinger et al. [48] used stochastic differential equation models and statistical inference on Fourier transforms of trajectory data, to analyse the effects of habitats' variables and behavior of other animals on movement paths of species of elk and deer. Bayesian and Hidden Markov models are widely used in predicting and identifying the behavioral states of animals as well [49, 50, 51, 52]. Another model of interest is Gaussian mixture model which for instance was used in [53] to classify flight modes. Given the availability of training labels, supervised methods are often used in applications like behavioral mode classification

and prediction [54, 55]. In [56], inverse reinforcement learning (IRL) was used to predict the missing movement data in a reward-based approach.

Problem Summary and Contributions

The challenges and problems in modeling trajectories that are deemed to be overlooked are summarized as following:

- Semantical approaches for modeling movement
- Discovery of informative low dimensional data manifolds for dynamics of animal movement

This study offers these contributions to tackle the stated challenges:

- Proposing a semantical approach for modeling cognitive components of animal movement in both spatial and spatiotemporal domains
- Proposing non-linear system identification algorithms for animal movement.
- Proposing a trajectory encoder model using recurrent neural networks.

1.3 Thesis Outline

Aside from the first chapter and last chapter, which provide basic concepts and generic introductory descriptions of addressed problems and their respective solutions, and summary remarks respectively, chapters of this thesis are categorized in three families based on the nature of the problems they approach. They are trajectory data acquisition in the second, trajectory data mining in chapters 3 to 6 and trajectory data visualization in the

last chapter. Here, the problems, motivations, proposed solutions and conclusions of each chapter are summarized.

In Chapter 2, we approach the problem of 3D trajectory reconstruction for animals using stereo imaging. Our approach, in general, focuses on the capturing of the movements of animals that are not easily distinguishable in their habitat. In particular, the ultimate goal is to be able to track bats flight in a non-invasive manner. Due to the widely availability of near infrared cameras, solution to this problem accommodates biologist and ecologist with new options for researching bats flight in their environment. They would be able to capture the trajectories without requiring specialized and sophisticated sensory machinery like thermal imaging and sonar devices. To achieve this objective, we propose a multistage motion-based algorithm which provides solutions for target acquisition, stereo point correspondence for 3D coordinates reconstruction and target identification for coherent tracking of texture-less objects. To verify the viability of our method, it was compared with flight paths captured from bats equipped with markers as reference target.

In Chapter 3, we attempt to provide a solution for modeling animal behavior based on their movement. With constantly growing volume of collected trajectory data from animals, it is necessary to devise and introduce techniques which are capable of dealing with larger and larger data sets. Therefore, we propose a geo-spectral approach which relaxes the sequential dependencies between trajectory points and using conditional independence models the latent states of responsible for generation of the trajectories. This approach consists of three main stages: extraction of geo-spectral features from trajectories, representation of trajectories based on extracted features and finally modeling behaviors based on such representations. In Chapter 3, we presented both generative and discriminative models constructed capable of modeling trajectories characterized by gender. A notable

advantage of our approach is its capability to deal with extensively large data sets. In fact, increase in size and balance of the data, contributes to accuracy and robustness of the constructed models. Nonetheless, a feature of this approach is inability to address temporal characteristics of behaviors. Yet, there might be remedies to tackle such shortcomings like temporal slicing. After all, we present extensive test results which evaluates the approach itself and compares the major options for its stages.

In Chapter 4, we take another approach to modeling latent structures in trajectories by introducing context in representation of points in trajectories. This is to model behaviors based on local structures in trajectories. This resembles temporal slicing as referred to in Chapter 3. With this approach, we have also a less space demanding and more informative alternative to n-gram structures for modeling sequences. Moreover, It is possible to embed the trajectory points in metric semantical subspaces which is beneficial in measuring embedded semantical similarities or dissimilarities between trajectory points. Our approach also provides a compact representation space for trajectory points by taking advantage of local embeddings. In addition, utilizing candidate sampling methods, it is possible to deal with large data sets. We have presented results of experiments in both data exploration and classification that support the viability of the proposed method.

In Chapter 5, recurrent neural networks were used to model the sequential dynamics of the trajectories. With employing autoencoder models, we attempted to discover data manifolds that are responsible for generation of trajectory paths. These manifolds could represent a combinatory state of internal and external components of animal movement. This provides researchers a tool to compare sequential dynamics of trajectory segments or study their dynamical relationship with environment. The presented networks including multimodal and generative models that are capable of reconstruction of parts of trajecto-

ries at different scales. A number of experiments were performed to closely examine the information capacity and generalization of the trained models. It was shown that recurrent models with complex structure like LSTMs could learn to distinguish between state dynamics of path procession. These states could be trained conditionally on terrain or environment features which in return could predict the future movement of animals given such features.

In Chapter 6, we attempt to approach the shortcomings in visualization of movement data. In fact, visualization techniques and tools play a prominent role in analyzing and exploring movement data and their models. Certainly, not to be overlooked, this hugely involves relevant atmospherical and geological data. With assistance of these tools, researchers are able to have a better understanding or assessment of movement features or behavioral symptoms. In other words, placing the animal movements in the context of their surrounding information could be significantly beneficial to researchers in terms of better understanding or describing underlying phenomena. In this chapter, we proposed an architecture for a web-based application which could manage and visualize the relevant atmospheric, oceanic and geologic conditions along with movement paths of animals. To demonstrate the essential functionalities of this application, a working prototype built and used for reconstruction of trajectories of gulls and shearwaters.

1.4 Notes to Readers

In this section, there a few notes that should be stated regarding software libraries used throughout this work and reuse of media materials included in this book.

Use of Open Source Software

It is worth to be acknowledged that, the open source software libraries used for visualization in general are this book are Matplotlib [57, 58], HoloViews [59] and Google maps [60]. With regards to numerical manipulation of data, geographical mappings, image processing and data modeling GeoPy [61], NumPy [62], OpenCV [63], Pandas [64], scikit-learn [65, 66], SciPy [67] , Theano [68] and TensorFlow [69, 70] were used.

Reuse of Media Contents

The last point to mention is regarding the reuse of images in this book, in particular in Chapter 6. In accordance with copyright notes provided by Google with regards to Google maps, in case of reuse, the readers are advised to include the image captions which include copyright information as well.

Chapter 2

Stereo Tracking for Reconstruction of 3D Trajectories of Bats

2.1 Introduction

Tracking bats flight was always a challenging task for researchers as it always required special equipments like sonars or thermal imaging devices. These devices are expensive and there are limitations in their applicability in certain conditions. Yet, another group of devices that are gaining grounds in performing similar task in comparable conditions are near infrared imaging (NIR) devices. These devices possess sufficiently sensitive sensors which are able to register rays near infrared wavelength. Images collected using these devices also having specific properties like lack of color features, and artifacts caused by higher aperture size to collect more light. Thence, there are specific set of techniques to be revised in order to extract information from these images. To reconstruct 3D trajectories, a pair of these devices could be positioned at a certain baseline distance from each other and using object correspondence, the depth or distance of the identified object to cameras

could be estimated. It is worth mentioning that with regards to bats, during this process few problems arise that have been overlooked. The first is recognition, or identification of the bat itself is challenging. Since bats appearance profile is elusive in their natural habitat, there is more effort required to track them. Instance of this are demonstrated later in this chapter. Second challenge is correspondence problem between consecutive frames in time and frames of each view to compute disparity distance. Moreover, increase in baseline distance to decrease depth ambiguity, renders incongruous projections of the same object on each image plane. This exacerbates the correspondence problem further. The third issue is related to the tracking anchor. Bats' highly variable geometry during flight introduces unwanted disturbances in trajectory of the registered geometric feature like centroid. Hence, to reconstruct more accurate trajectories, a solution should be formulated to recover the anchor point from the detected segment. Here, we attempt to provide solutions to the aforementioned problems by introducing an motion-based recognition method which explains the observations in terms of model bases.

2.2 Preliminary Concepts

2.2.1 Epipolar Geometry

In stereo imaging, the relationship between projection of 3D structures could be described in terms of `epipolar geometry`. Main components of epipolar geometry are `epipolar points`, `epipolar lines` and `epipolar planes`. Given a generic stereo configuration for cameras C^l and C^r, and object point P in 3D, optical center of C^l, O^l, is considered as the origin. B, distance between two cameras along horizontal axis is called `baseline` distance. Π^l and Π^r are image planes of two cameras and $\mathbf{p}^l$ and $\mathbf{p}^r$ are corre-

sponding images of P on the image planes. Triangle $\overset{\triangle}{O^l O^r P}$ constructs the epipolar plane. Epipoles are images of each optical center on the image plane of the other camera. Epipolar lines are intersection of epipolar plane with image planes of each camera. Figure 2.1(a) shows a stereo configuration and corresponding epipolar elements. It is seen that p^l and p^r are guaranteed to lie on the corresponding epipolar lines in Π^l and Π^r. This relationship is described in a mapping matrix called `fundamental matrix`, $\mathbf{F}$, where it maps a point on an image plane to the corresponding epipolar line on the other image plane as:

$$\mathbf{l}^r = \mathbf{F}\mathbf{p}^l \tag{2.1}$$

Given points $\mathbf{p}^l$ and $\mathbf{p}^r$ coordinates on the image planes, it is possible to recover the distance of point P from the baseline of the cameras as following:

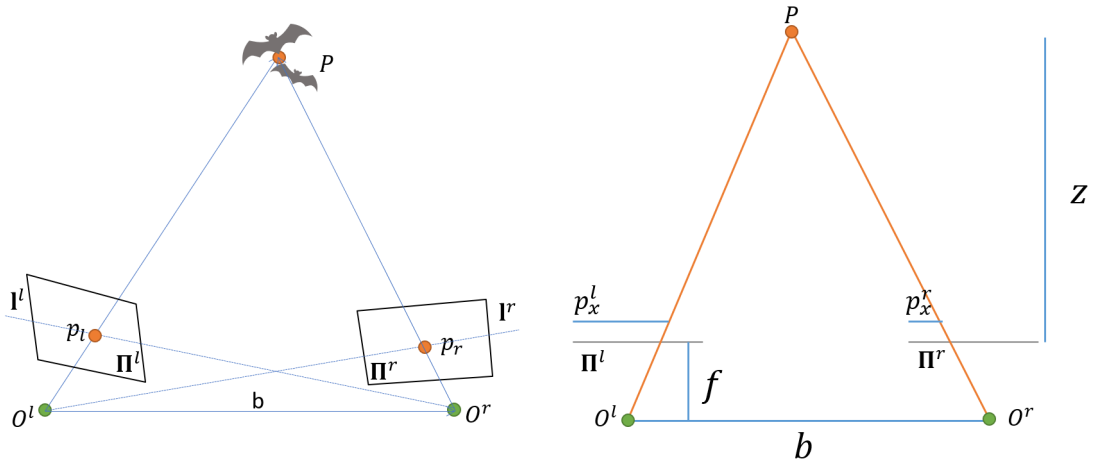
$$P_z = f\frac{b}{p_x^l - p_x^r}, \tag{2.2}$$

where $p_x^l - p_x^r$ is termed as `disparity` and it is inversely proportional to distance of the point to the optical center of the camera. Figure 2.1(b) illustrates the relationship between disparity and depth.

The major step for computing disparities is point correspondence. As a result, to compute disparities, for points in left image, corresponding points in right image is sought for. The problem could be defined as:

$$\hat{d} = \arg\max_d \psi(\mathbf{p}^l, \mathbf{p}_d^r) \tag{2.3}$$

where $\psi(\mathbf{p}^l, \mathbf{p}_d^r)$ is measure of similarity between point $\mathbf{p}^l$ in left image and point $\mathbf{p}_d^r$ in right image at disparity d in set of D disparities. Knowledge of fundamental matrix is significantly reduce the search space of the problem. It was shown that corresponding points ideally should lie on the corresponding epipolar line. The limit the search window to a line or a group of lines in contrast to whole image.



(a) (b)

Figure 2.1: (a) Epipolar lines and epipolar plane. (b) Inverse relation between depth and disparity.

There various methods available for measuring similarity between patches of stereo images. The simplest is sum of absolute differences as:

$$SAD(\mathbf{p}^l, \mathbf{p}^r) = |\mathbf{p}_I^l) - \mathbf{p}_I^r|, \tag{2.4}$$

where $\mathbf{p}_I$ determines the image intensity at point p. A more complicated measure is normalized cross correlation which is defined as:

$$NCC(\mathbf{p}^l, \mathbf{p}^r) = \frac{\sum_{\mathbf{n}\in\mathbf{N}} \mathbf{n}_i^r \mathbf{n}_i^l}{\sqrt{\sum_{\mathbf{n}\in\mathbf{N}} (\mathbf{n}_i^l)^2 \sum_{\mathbf{n}\in\mathbf{N}} (\mathbf{n}_i^r)^2}}, \tag{2.5}$$

where $\mathbf{N}$ is the set of neighboring points of $\mathbf{p}$. $NCC(.,.)$ has a zero mean variant that is denoted with $ZNCC(.,.)$ and only defers where it calculates the means of intensities and subtracts them from each pixel's intensity. More methods like census, and rank are described and benchmarked in [71].

2.2.2 Gaussian Mixture Models

Almost all of distribution functions could be approximated by superposition or linear combination of a sufficient number of Gaussian distributions. These combinations are called Gaussian mixture models [72]. These could be formulated as:

$$P(\mathbf{x}) = \sum_{c \in C} \pi_c \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c), \quad \sum_{c \in C} \pi_c = 1, \ \pi_c \geq 0 \tag{2.6}$$

where C is the number of Gaussian components with mean vector and covariance matrix of $\boldsymbol{\mu}_c$ and $\boldsymbol{\Sigma}$ respectively. Consequently, the log likelihood of independent data points in data set $\mathbb{X}$ is given as:

$$\mathcal{L}(\mathbb{X}|\boldsymbol{\pi}, \boldsymbol{\mu}, \boldsymbol{\Sigma}) = \sum_{\mathbf{x} \in \mathbb{X}} \ln \sum_{c \in C} \pi_c \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_k, \boldsymbol{\Sigma}_k) \tag{2.7}$$

Expectation-Maximization Algorithm

Expectation-maximization algorithm (EM) introduced in [73, 74, 75] provide a solution for optimizing the parameters of Gaussian mixture models using likelihood measure in eq. 2.7. This algorithm comprised of two major steps: **E** and **M**. After initialization of parameter set $\{\boldsymbol{\mu}, \boldsymbol{\Sigma}, \boldsymbol{\pi}\}$, posterior probabilities of parameters are calculated for each data point as following:

$$\hat{p}(\mathbf{x}_c) = \frac{\pi_c \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_k)}{\sum_{c' \in C} \pi_{c'} \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_{c'}, \boldsymbol{\Sigma}_{c'})}, \quad |\mathbb{X}| = N \tag{2.8}$$

where $\hat{p}(\mathbf{x}_c)$ measures component c's level of ownership of data point $\mathbf{x}$. Then using posterior values computed in **E** step, the parameter set is updated as following:

$$\hat{\boldsymbol{\mu}}_c = \frac{1}{|\mathbf{X}_c|} \sum_{\mathbf{x} \in \mathbb{X}} \hat{p}(\mathbf{x}_c)\mathbf{x} \tag{2.9}$$

$$\hat{\boldsymbol{\Sigma}}_c = \frac{1}{|\mathbf{X}_c|} \sum_{\mathbf{x} \in \mathbb{X}} \hat{p}(\mathbf{x}_c)(\mathbf{x} - \hat{\boldsymbol{\mu}}_c)(\mathbf{x} - \hat{\boldsymbol{\mu}}_c)^T \tag{2.10}$$

$$\hat{\pi}_c = \frac{|\mathbf{X}_c|}{|\mathbb{X}|}. \tag{2.11}$$

Then log likelihood is evaluated using updated parameters and compared against a threshold to determine convergence. This process is repeated until convergence or to a predefined iterations.

2.2.3 Belief Propagation

An efficient algorithm for inference on probabilistic graphical model is sum-product message passing or belief propagation method (BP) [76]. Given a tree-structured probabilistic graph, BP could achieve exact inference [75]. This algorithm involves evaluating local marginals. A marginal is computed by summing over the joint distribution as:

$$P(x) = \sum_{\mathbf{x}\backslash x} P(\mathbf{x}), \tag{2.12}$$

where $\mathbf{x}\backslash x$ means variable x is excluded from the joint. By creating a factor graph [75, 77] from the probability graph and partitioning the joint distributions, the joint distribution could be written in form of product of factors as:

$$P(\mathbf{x}) = \prod_{\mathbf{s}\subset\mathbf{x}} f_{\mathbf{s}}(\mathbf{s}), \tag{2.13}$$

where $f_{\mathbf{s}}$ is function of variables in $\mathbf{s}$ a subset of $\mathbf{x}$. By grouping factors connected to variable x, the joint distribution is:

$$P(\mathbf{x}) = \prod_{s\in S} F_s(x, X_s) \tag{2.14}$$

where S and X_s are set of neighboring factor nodes and their variables respectively. F_s is joint of factors in the group. Plugging Eq. 2.14 in to Eq. 2.12 results in:

$$\begin{aligned} P(x) &= \sum_{\mathbf{x}\backslash x} \prod_{s\in S} F_s(x, X_s) = \prod_{s\in S} \sum_{\mathbf{x}\backslash x} F_s(x, X_s) \\ &= \prod_{s\in S} \mu_{f_s\to x}(x), \end{aligned} \tag{2.15}$$

where $\mu_{f_s \to x}(x)$ is termed as `message` which is being passed from f_s to x. Message $\mu_{f_s \to x}$ is marginalized as:

$$\mu_{f_s \to x}(x) = \sum_{x_1, x_2, \ldots, x_M} f_s(x, x_1, x_2, \ldots, x_M) \prod_{m \in G \setminus x} \mu_{x_m \to f_s}(x_m), \tag{2.16}$$

where G is set of variable nodes neighboring f_s.

$$G_m(x_m, X_s) = \prod_{l \in L \setminus f_s} F_l(x_m, X_l), \tag{2.17}$$

where L is set of factor nodes neighboring variable node x_m. $\mu_{x_m \to f_s}(x_m)$ is defined as:

$$\begin{aligned} \mu_{x_m \to f_s}(x_m) &\equiv \sum_{\mathbf{x} \setminus x} G_m(x_m, X_s) \\ &= \prod_{l \in L \setminus f_s} \sum_{\mathbf{x} \setminus x} F_l(x_m, X_l) \\ &= \prod_{l \in L \setminus f_s} \mu_{fl \to x_m}(x_m). \end{aligned} \tag{2.18}$$

2.2.4 Total Variation Regularization

Total variation (TV) regularization introduced in computational image processing in [78, 79]. It has been proposed that rather than L_2-norm, TV is the proper norm for images. TV norm is in fact absolute value of derivative or L_1-norm of gradient vector. However, unlike L_2-norm, closed form solution is not trivial. In [80], it is stated that the proper class for many image processing task like denoising is bounded total variation (BV) function space. Given u is smooth, the solution to denoising problem is the result of TV-L_2 minimization below:

$$\arg\min_f \lambda \int_\Omega (f - I)^2 + \int_\Omega |\nabla f| \tag{2.19}$$

where f is estimated image, I is observed image and λ is data weight. The first term is data term and the second term is TV which encourages less oscillations and allows edges.

TV-L_1 version [81] of this equation is written as:

$$\arg\min_f \lambda \int_\Omega |f - I| + \int_\Omega |\nabla f|. \tag{2.20}$$

Another application of TV terms in image processing is in optical flow. In [82, 83] TV-L_1 was used to define the following terms to ensure piecewise smooth flow field:

$$E_s(u, v) = \int_\Omega |\nabla u| + |\nabla v|, \tag{2.21}$$

where u and v are horizontal and vertical components of optical flow. In [83] to approximate L_1 minimization, concave function $\psi(x) = \sqrt{x + \epsilon^2}$, where $x \geq 0,\ \epsilon > 0$, was applied on variation terms as:

$$E_s(u, v) = \int_\Omega \psi(|\nabla u|^2 + |\nabla v|^2). \tag{2.22}$$

The total energy functional is written as:

$$E(u, v) = E_d + \lambda E_s. \tag{2.23}$$

E_d, which assumes both intensity and gradient constancy, is described as:

$$\begin{aligned} E_d(u, v) = \int_\Omega \psi(&|I(x + u, y + v, t + 1) - I(x, y, t)|^2 \\ &+ \gamma|\nabla I(x + u, y + v, t + 1) - \nabla I(x, y, t)|^2) \end{aligned} \tag{2.24}$$

where, γ is a weighting coefficient. To minimize the total energy functional, numerical approximation of the solution to the Euler-Lagrange equations of E computed with use of fixed point iterations. Yet another approach for constructing energy functional is simply consider intensity consistency over time for data term and use Taylor expansion to linearize the nonlinear image function as:

$$\begin{aligned} E_d(u, v) = \int_\Omega &|\nabla I(x + u_0, y + v_0, t + 1)\ .\ (u - u_0, v - v_0) \\ &+ I(x + u_0, y + v_0, t + 1) - I(x, y, t)| \end{aligned} \tag{2.25}$$

where u_0 and v_0 are the values at which image function gradient approximated. With reference to [82], the total energy functional could be written with weighted data, smoothing and convex relaxation terms as:

$$E(u,v) = \int_\Omega \lambda E_d(u,v) + E_s(u_a,v_a) + \frac{1}{2\theta}|u - u_a + v - v_a|^2 \tag{2.26}$$

where θ controls penalizing disparity between u_a, v_a, and u, v respectively. Given small values for θ, the solution is accepted when the relaxation term is nearly zero. The optimization is done iteratively by alternatively fixing a pair variables constant and solve minimization problem in part. First u_a and v_a constant and minimize:

$$\min_{u,v} \int_\Omega \lambda E_d(u,v) + \frac{1}{2\theta}|u - u_a + v - v_a|. \tag{2.27}$$

Then fix u and v, solving:

$$\min_{u_a,v_a} \int_\Omega E_s(u_a,v_a) + \frac{1}{2\theta}|u - u_a + v - v_a|. \tag{2.28}$$

The first optimization problem could be solve using point-wise thresholding and the second using Chambolle's duality-based algorithm as stated in [82].

2.3 Methodology

2.3.1 Problem Formulation

There are a set of problems that should be resolved in order to reconstruct 3D trajectories of bats using stereo images. We leave out preprocessing challenges and assume that images are single channel grayscale, their histogram are adjusted sufficiently to achieve reasonable intensity and contrast levels for further procedures and stereo rectified. In the first stage we deal with recognition of the objects in the scene we are interested in. Since we

assume that objects of interest, bats, occupy a subset of image real estate, we approach the first stage as background modeling problem. This is beneficial as it significantly reduces the search space for this problem as whole. In this stage we construct model of background components and mask out pixels determined to be background from our object search. In addition, certain geometric criteria could be used to select a subset of detected foreground segments that presumed to be projected surfaces on the image plane. Set of foreground segments in camera image I_i is defined as:

$$\mathcal{F}_i = \{\pi_i(S) \mid S \in \mathbb{S}\}, \quad i \in \{l, r\}, \tag{2.29}$$

where S is a surfaces in set of surfaces in 3D space and $\pi_i : \mathbb{R}^3 \rightarrow \mathbb{Z}^2$ is the projection function of camera i in set of l and r representing left and right. Camera image is defined as mapping function $I_i : \Omega_i \rightarrow \mathbb{R}$, $\Omega_i \subset \mathbb{Z}^2$. In the next stage, to estimate the depth of qualified foreground segments, correspondences and subsequently, disparities should be determined. As stated previously, simply using intensity or color correspondence for computing disparities produces poor results. In order to resolve this we introduce new criteria and constraints to correspondence evaluation. Given rectified stereo rectified images, for pixels $\mathbf{p}_i = \{c, x, y, f\},\ i \in \{r, l\}$:

$$\begin{aligned} |c_l - c_r| &< \delta_c \\ |x_l - x_r| &< \delta_x \\ |y_l - y_r| &< \delta_y \\ f_l - f_r &= 0 \end{aligned} \tag{2.30}$$

where c, x, y are intensity, x-coordinate and y-coordinate of the pixel. f is a binary variable, identifying whether it is part of foreground segment. $\delta_c, \delta_x, \delta_y$ are intensity disparity threshold, maximum lateral disparity and vertical tolerance respectively. With these constraints, we approach this problem as inference on Markov random fields (MRF) [84]. Set

of disparity labels $\mathcal{D}$ is defined and BP algorithm is used to find the likely disparity labels. The major goal of this stage is to approximately measure the depth of the foreground segments resulted in the first stage. In the third stage, we use a decoupled scene flow algorithm to optimize u, v, and p corresponding to lateral, vertical and change in disparity [85]. Given u, v, p and d with calibration parameters of cameras we can compute the coordinates and the motion of pixels in the foreground segment in 3D. The information obtained then used to determine the dynamics of moving parts of the object and fit the appropriate model priors. This is essential for solving this problem as we do not have access to the interior points of the projected surfaces and recognition should be done on the features of the object silhouettes.

2.3.2 Background Modeling

Background modeling is a fundamental stage in our algorithm. Failure to identify the relevant and informed image segments would bring down efficiency of the results as whole. Since the main objective is to track moving objects, we could exploit this prior knowledge to increase both computational efficiency and performance of the algorithm. Simple background subtraction could provide hints about the movements in the image but, noise and multimodality of the background components could introduce redundancies. Therefore, adaptive methods with multi-component models like Gaussian mixture models [86] and kernel density estimator models [87] were introduced to represent a more robust model of background pixels. However, there was still issue of computational complexity still persisting. With introduction of online expectation maximization algorithms, adaptive methods based on Gaussian mixture models [88, 89, 90] gained more popularity. There are also principal components analysis (PCA) and factorization approaches to this problem as well

[91, 92]. Here, we choose to approach the background modeling employing the technique in [90]. This technique is an online method and is based on Gaussian mixture models. The membership of the object to foreground is decided based on $P(\mathbf{p}^t|\mathcal{B})$ value, which is probability of color components of pixel $\mathbf{p}^t_{x,y}$ at time t given background distribution model $\mathcal{B}$, being less than a threshold ω. The density components are updated periodically and represented by C components, with means and isotropic covariance matrices, $\boldsymbol{\mu}_1, \ldots, \boldsymbol{\mu}_C$ and $\sigma_1\mathbf{I}, \ldots, \sigma_C\mathbf{I}$, respectively. The estimated density for a pixel is described as:

$$P(\mathbf{p}|\mathcal{P}^T, \mathcal{I}) = \sum_{c \in C} \pi_c \mathcal{N}(\mathbf{p}|\boldsymbol{\mu}_c, \sigma_c\mathbf{I}), \tag{2.31}$$

where $\mathcal{P}^T$ is set of pixel values over time period T, and $\mathcal{I}$ is the model of both foreground $\mathcal{F}$ and background $\mathcal{B}$ components in the image.

According to [93], parameters of the components are updated as following:

$$\pi_c^t = \pi_c^{t-1} + \alpha(o_c^t - \pi_c^{t-1}), \tag{2.32}$$

$$\boldsymbol{\mu}_c^t = \boldsymbol{\mu}_c^{t-1} + o_c^t(\frac{\alpha}{\pi_c^t})\boldsymbol{\delta}_c, \tag{2.33}$$

$$\sigma_c^t = \sigma_c^{t-1} + o_c^t(\frac{\alpha}{\pi_c^t})(\boldsymbol{\delta}_c^T\boldsymbol{\delta}_c - \sigma_c^{t-1\,2}), \tag{2.34}$$

where $\alpha = 1/T$, $\boldsymbol{\delta}_c = \mathbf{p}^t - \boldsymbol{\mu}_c^{t-1}$ and o_t is binary variable which determines the membership to component c given a distance threshold compared against the distance to the component, Δ_c, which is computed as:

$$\Delta_c^2(p^t) = \frac{\boldsymbol{\delta}_c^T\boldsymbol{\delta}_c}{\sigma_c^{t-1\,2}}. \tag{2.35}$$

If there is no component withing threshold distance, a new component with following parameters is appended to C:

$$\pi = \alpha, \quad \boldsymbol{\mu} = \mathbf{p}^\mathbf{t}, \quad \sigma = \sigma_0$$

where σ_0 is an arbitrary initial value for variance. Given the trained component models, the background components are assigned to the components with largest pi_c values, or as

in [90], with sorted weight in descending order, the background components are:

$$\arg\min_{b} \sum_{c}^{b} \pi_c > (1 - c_{\mathcal{F}}) \tag{2.36}$$

where $c_{\mathcal{F}}$ is the ratio of the foreground occlusion in the image. Here, we take the approach presented in [90] to adaptively update priors of the components π_c using Dirichlet prior coefficients $\kappa_c = -k$ where maximum a posteriori solution can be written in recursive form as:

$$\pi_c^t = \pi_c^{t-1} + (o_c^t - \pi_c^{t-1})/t - k_T/t, \quad k_T = k/T \tag{2.37}$$

where T is a large constant time period and therefore k_T is a constant. However, in our case, these component models are computed updated for each of left and right images separately. In the next stage, using the probability density models learnt here, we design correspondence measures for pixels between left and right images. Every pixel $\mathbf{p}_i,\ i \in \{r, l\}$ membership component parameters are π_{p_i}, $\boldsymbol{\mu}_{p_i}$ and σ_{p_i}. The probability density for pixel $\mathbf{p}_j$ belong to the density of $\mathbf{p}_i$ is:

$$P(\mathbf{p}_j|\mathbf{p}_i) = \pi_{\mathbf{p}_i} \mathcal{N}(\mathbf{p}_j|\pi_{\mathbf{p}_i}, \boldsymbol{\mu}_{\mathbf{p}_i}, \sigma_{\mathbf{p}_i}) \tag{2.38}$$

In the final part of the first stage, pixel segments identified as foreground are passed to the next stage for disparity estimation.

2.3.3 Disparity Estimation

For disparity estimation, only the rows of the detected foreground segments in both images are selected as search regions. These search regions fall within epipolar threshold for vertical tolerance. To compute disparities we design a global loss function which consists of data or observations term E_o instead of E_d to avoid confusion with disparity d, and

smoothness term E_s as below:

$$E(d) = \sum_{p \in \hat{\mathcal{F}}} \left[E_o(p,d) + E_s(p,d) \right], \tag{2.39}$$

where $\hat{\mathcal{F}} = \mathcal{F}_l \cup \mathcal{F}_r$ is the selected foreground rows. Data cost term promotes intensity and segment consistency along with epipolar criterion. Therefore, we can write E_o as:

$$E_o(p,d) = \lambda_i ZNCC(p,d) + \lambda_b \mathcal{L}(p,d) + \lambda_e |\Delta_y(p,d)|, \tag{2.40}$$

where λ_i, λ_b and λ_e are weighting coefficients for intensity consistency, segment consistency and epipolar criterion penalties. $ZNCC(.)$ is zero mean normalized cross correlation:

$$ZNCC(p,d) = \frac{\sum_{n \in N}(n_p - \bar{n}_p)(n_{p_d} - \bar{n}_{p_d})}{\sqrt{\sum_{n \in N}(n_p - \bar{n}_p)^2 \sum_{n \in N}(n_{p_d} - \bar{n}_{p_d})^2}}, \tag{2.41}$$

where N is set of neighboring pixels, $\bar{n}_p$ and $\bar{n}_{p_d}$ are means of each neighborhood. $\Delta_y(.)$ is vertical disparity:

$$\Delta_y(p,d) = |y_p - y_{p_d}|, \tag{2.42}$$

where y_{p_d} is the vertical component of the pixel at disparity d and $\mathcal{L}(.)$ is negative log likelihood of a pixel does not belong to background component computed as:

$$\mathcal{L}(p,d) = -\log \left[1 - \sum_{c \in \mathcal{B}} \pi_c \mathcal{N}(p_d; \mu_c, \sigma_c^2) \right]. \tag{2.43}$$

With regards to smoothing term E_s, we consider penalizing spatial and temporal discontinuities by introducing spatial and temporal smoothness terms as:

$$E_s(p,d) = \lambda_s \sum_{q \in Q(p)} |d_p - d_q| + \lambda_t |d_p^t - d_p^{t-1}|, \tag{2.44}$$

where $Q(p)$ is set of spatial neighboring pixels of p. The first term discourages large disparities between neighboring pixels and the second term penalizes sharp changes in disparity of the same pixel in consequent frames. It should be noted that to avoid over-smoothing of

the results, these losses are generally truncated explicitly or robust estimator functions are used. We also use additional segmentation information to manipulate the smoothing costs to allow informed discontinuities.

To solve the minimization problem in this stage, we quantize the disparity levels and assign a label to each level. Then we approach this problem as a labeling problem inspired by [84, 94]. Here, a Markov random field (MRF) connectivity graph of image pixels is spanned. Each pixel statutes a hidden node four-way connected to neighboring pixels and data nodes consist of observation cost and temporal consistency cost. Then, we can approach the labeling problem as a maximum a posteriori problem in MRF. We use message passing algorithm with negative log likelihood minimization to solve the MAP problem.

To reach an optimum state, the graph nodes start pass messages initialized to zero to each other updating states and subsequent messages given observation nodes. Each node stores a vector of disparity costs $\mathbf{h} \in \mathbb{R}^{\mathcal{D}}$ at each disparity label. A min-sum BP update message to a neighboring node q from node $p \in \hat{\mathcal{F}}$ is computed as:

$$\boldsymbol{\mu}^{i}_{p\to q}(q) = \min_{p} \left[\mathbf{F}(p,q) + \mathbf{h}(p)\right], \tag{2.45}$$

$$\mathbf{F}_{(d_p,d_q)}(p,q) = \rho(\lambda_s |d_p - d_q|, \epsilon_s), \tag{2.46}$$

$$\mathbf{h}(p) = \mathbf{E}_o(p) + \rho(\lambda_t |\mathbf{d}_p - \mathbf{b}^{t-1}_p|, \epsilon_t) + \sum_{q' \in Q(p)\setminus q} \boldsymbol{\mu}^{i-1}_{q'\to p}(p), \tag{2.47}$$

where i is the iteration step number, $F(p,q)$ is vector of $\mathcal{D}$ disparities for node q evaluated for $\mathcal{D}$-dimension vector node p, and $\mathbf{b}^{t-1}$ is the belief of the pixel in the previous frame tiled as a vector, $\mathbf{d}$ is vector of all disparity labels, $\rho(.,\epsilon_.)$ is the robust estimator function with parameter $\epsilon_.$. This function could be simply a truncation at ϵ value. As one suspects, the computational cost of $\mathbf{F}(p,q)$ is $O(\mathcal{D}^2)$. To reduce this cost to linear time, `min convolution` algorithm introduced in [84] is used. In case of linear model for $\rho(.,\epsilon)$, the

update message equation for a disparity label d_q for node q is:

$$\mu^{i}_{p\to q}(dq) = \min_{dp}(\lambda_s|d_p - d_q| + h(d_p)), \tag{2.48}$$

and could be imagined as $\mathcal{D}$ upward cones with tips situated at point $(d_p, h(d_p))$ with slope λ_s. The solution to the minimization problem is lower envelope of the cones that is computed using linear model min convolution forward-backward algorithm in [84]. Then the truncated message update is:

$$\mu_{p\to q}(dq) = \min(\mu_{p\to q}(d_q), \min_{dp} h(d_q) + \epsilon_s) \tag{2.49}$$

Given a sufficient number of iterations, the $\mathbf{b}_p$ is belief vector for node p whose disparity is minimum of the $\mathbf{b}_p$ as:

$$\hat{d}_p = \arg\max_{d} \mathbf{b}_p. \tag{2.50}$$

It should be noted that, if a pixel was not segmented as foreground in previous stage, in this stage remains inactive and there is no message should be generated from it while they still receive messages from active neighboring pixels. At this point, the calculated disparities are passed to the next stage for estimation of scene flow.

2.3.4 Scene Flow Estimation

To create dense 3D motion fields from stereo images and the estimated disparities in the previous stage, we take the approach presented in [85] to estimate the optical flow in horizontal u and vertical v directions in addition to the change in disparities p. This flow field is the scene flow field and parameterized by $[u(x, y, t), v(x, y, t), p(x, y, t)]^T$ vector. With estimation of these parameters using the scene flow algorithm, it is possible to reconstruct the motion in 3D. Given a pair of stereo images at time t with computed disparity map

$d(x, y, t),\ I(x, y) \in \hat{\mathcal{F}}$ belonging to the foreground segments, a pair of stereo images at time $t + 1$ and intensity consistency assumption, following constraints could be imposed:

$$E_{I_l} = I_l(x, y, t) - I_l(x + u, y + v, t + 1) = 0, \tag{2.51}$$

$$E_{I_r} = I_r(x, y, t) - I_r(x + u + d + p, y + v, t + 1) = 0, \tag{2.52}$$

$$E_d = I_l(x + u, y + v, t + 1) - I_r(x + u + d + p, y + v, t + 1) = 0 \tag{2.53}$$

where $I_l(.)$ and $I_r(.)$ are the left and right image functions respectively, and (x, y, t) subscript is dropped for u, v and p for the sake of brevity.

Based on the constraints above, data term could be described as:

$$E_d = \psi(E_{I_l}^2) + c\psi(E_{I_r}^2) + c\psi(E_d^2) \tag{2.54}$$

where $\psi(.)$ is TV-L^1 approximation and c is a binary coefficient disabling the loss if disparity information is not available for the pixel. The smoothness term consists of variation penalties for all variables is as following:

$$E_s = \lambda_s \psi(|\nabla u|^2 + |\nabla v|^2) + \gamma\psi(|\nabla p|^2), \tag{2.55}$$

where λ_s and γ are weighting factors for smoothness of optical flow and disparity flow components respectively. We construct an energy functional combining the data and the smoothness terms defined above as:

$$E(u, v, p) = \int_\Omega (E_d(u, v, p) + E_s(u, v, p))\mathbf{dx}, \tag{2.56}$$

where $\mathbf{dx}$ consists of dx and dy components.

To optimize this energy functional Euler-Lagrange equations are calculated as in [85]. Similarly, the linearization technique presented in [83] are utilized to iteratively solve the nonlinear Euler-Lagrange equations. With a solution for the scene flow vector field, we are able to create the motion components of the scene in the next stage.

2.3.5 Extraction of Motion Field Components

With disparities obtained in the second stage, 3D coordinates could be derived with calibrated camera parameters as:

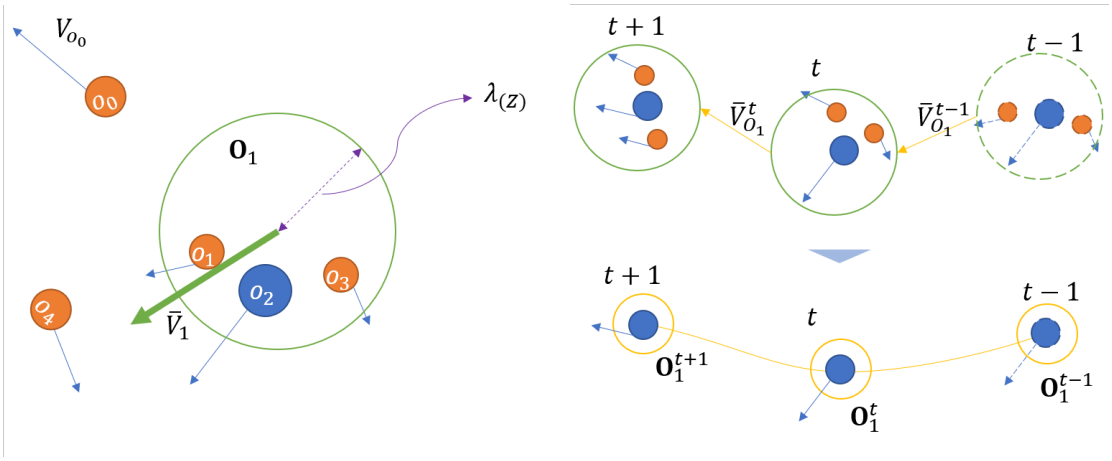
$$W\begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & -c_x \\ 0 & -1 & 0 & c_y \\ 0 & 0 & 0 & f \\ 0 & 0 & \frac{1}{b} & 0 \end{bmatrix}\begin{bmatrix} x \\ y \\ d \\ 1 \end{bmatrix} \tag{2.57}$$

where f is focal length, c_x, c_y are coordinates of principal point of left camera on its image plane and b is the baseline distance between cameras. Then with flow parameters obtained in the previous stage, it is possible to write the change in 3D coordinates as following:

$$\begin{bmatrix} \Delta X \\ \Delta Y \\ \Delta Z \end{bmatrix} = b\begin{bmatrix} \frac{x+u-c_x}{d+p} - \frac{x-c_x}{d} \\ \frac{y-c_y}{d} - \frac{y+v-c_y}{d+p} \\ \frac{f}{d+p} - \frac{f}{d} \end{bmatrix} \tag{2.58}$$

Given the 3D motion field of foreground pixels, we identify the density components of the 3D motion field using 6 dimensional feature vector $[X, Y, Z, \Delta X, \Delta Y, \Delta Z]^T$ with depth dependent minimum number $\eta_{(Z)}$ for pixels and a maximum distance $\sigma_{(Z)}$. Then, we extract the components $\mathbf{o}$ that fit the simplified model of flying bats with group of moving major components within a distance threshold $\lambda_{(Z)}$. The range and levels of these parameters are dependent on the physical properties of the bats to be tracked and camera baseline distance. Figure 2.2 demonstrates an instance of the mentioned procedures. Finally, to distinguish between moving parts of the object for tracking, we employ the dynamic model of bat flight. Here, we compare the motion vector of each individual component with the mean movement of the object and identify the component that has the least divergence from the mean. In other words, the inner products of the mean motion vector of the object with the ideal component should be maximized or formulated as:

$$\hat{\mathbf{o}} = \arg\max_{\mathbf{o}\in\mathbf{O}} \mathbf{o}^T \bar{V} \tag{2.59}$$



(a) (b)

Figure 2.2: Schematic diagram of motion component extraction and object identification process. (a) Single frame. (b). Across multiple frames.

where, $\mathbf{o}$ is a component vector of an object $\mathbf{O}$ with average motion vector $\bar{V}$. Figure 2.2(a) shows a schematic diagram of the mentioned process. Of course this technique is more robust across frames in time where the mean motion is computed across multiple frames as shown in Figure 2.2(b).

2.4 Experiments, Results and Discussion

Prior to processing the image for foreground extract, stereo images are undistorted and rectified. As a result, the correspondence search performed along rows. Then, the image intensities were transformed using the following power transform:

$$\hat{I}(x,y) = \alpha I(x,y)^{\gamma}, \quad \gamma = 0.8,\ \alpha = 2.2 \tag{2.60}$$

The results of power transformation for different values of γ and α is shown in Figure 2.3. Then images are denoised using box filter, fast non-local means (fastNLMeans) [95, 96] or TV-L^1 [97] methods. Figure 2.4 illustrates the results of denoising and corresponding

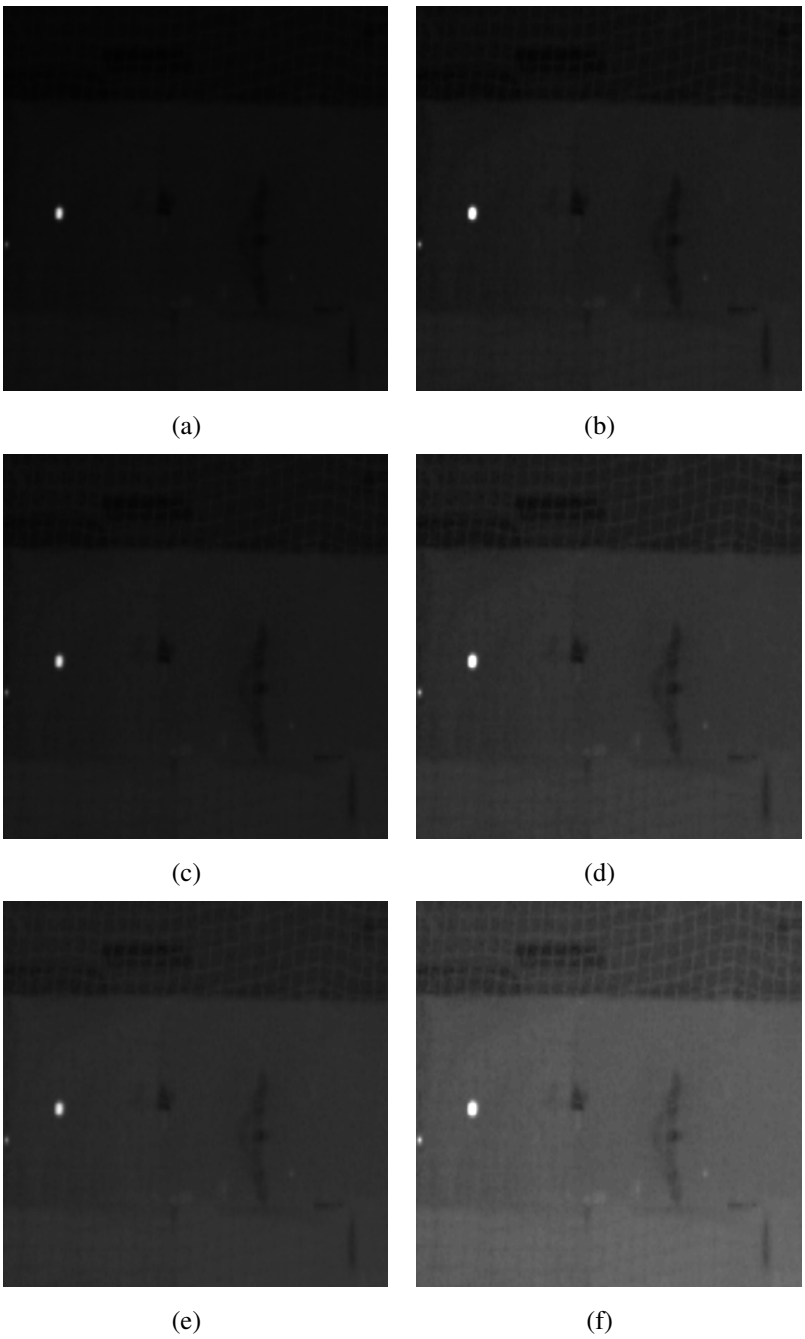


Figure 2.3: A sample image of a bat with different power transform parameters. It is seen that unlike linear transforms, non-linear power transform greatly improve contrast of the object. (a) Original Image (b) $\gamma = 1, \alpha = 3$ (c) $\gamma = 0.8, \alpha = 1$ (d) $\gamma = 0.8, \alpha = 3$ (e) $\gamma = 0.6, \alpha = 1$ (f) $\gamma = 0.6, \alpha = 3$

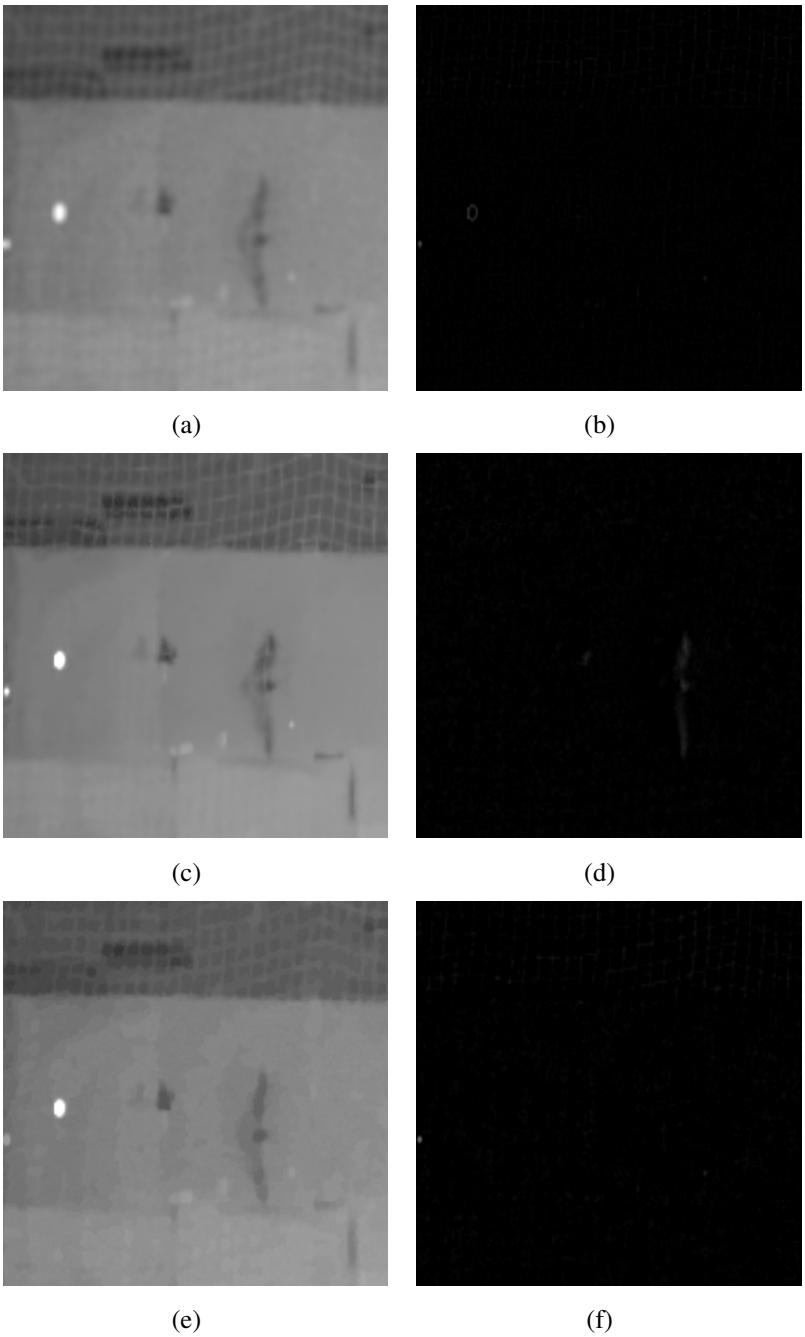
residual images. As suggested in [98] the performance of the optical flow algorithm could be improved using residual images.

To reduce computational work load and reduce noise, we examine providing different pyramid levels of the stereo images to the algorithm. However, due to the small size of the bats in the images, the down-sampling of the images could result in loss of information. We choose to work with only two pyramid scale levels, full-size and half-size. Then the images are passed to background subtraction algorithm. The results for half and full-size at different poses are shown in Figure 2.5, and Figure 2.6 respectively. Considering Sobel edges, it is seen that in most of cases the objects are properly classified as foreground but, results in case of full-size images contain more noise.

In the next stage, the foreground segments are used to compute disparity map. This greatly reduces the computational cost for subsequent stages. It is worth mentioning that, in case of lighting conditions or poor image quality causing missing corresponding segments on left or right images, we would consider intersection of foreground rows on both images with some margin.

As stated earlier, BP algorithm is used to determine disparities in foreground segments. For the sake of comparison, the estimated disparities using semi-global block matching algorithm [1] on the whole image is illustrated in Figure 2.7. In most cases, the foreground object would not appear in the foreground or its depth data is diffused with surroundings.

BP algorithm could be slower depending on the size of the foreground segment but it achieves more consistent results. Disparity maps obtained from BP algorithm for foreground row is shown in Figure 2.8. Maximum disparities were set to 256 and universal λ values 15 and 8 were used for weighting energies. Figure 2.9 show the energy function values versus iterations for the settings. It is seen that, with increase in number of iterations, a

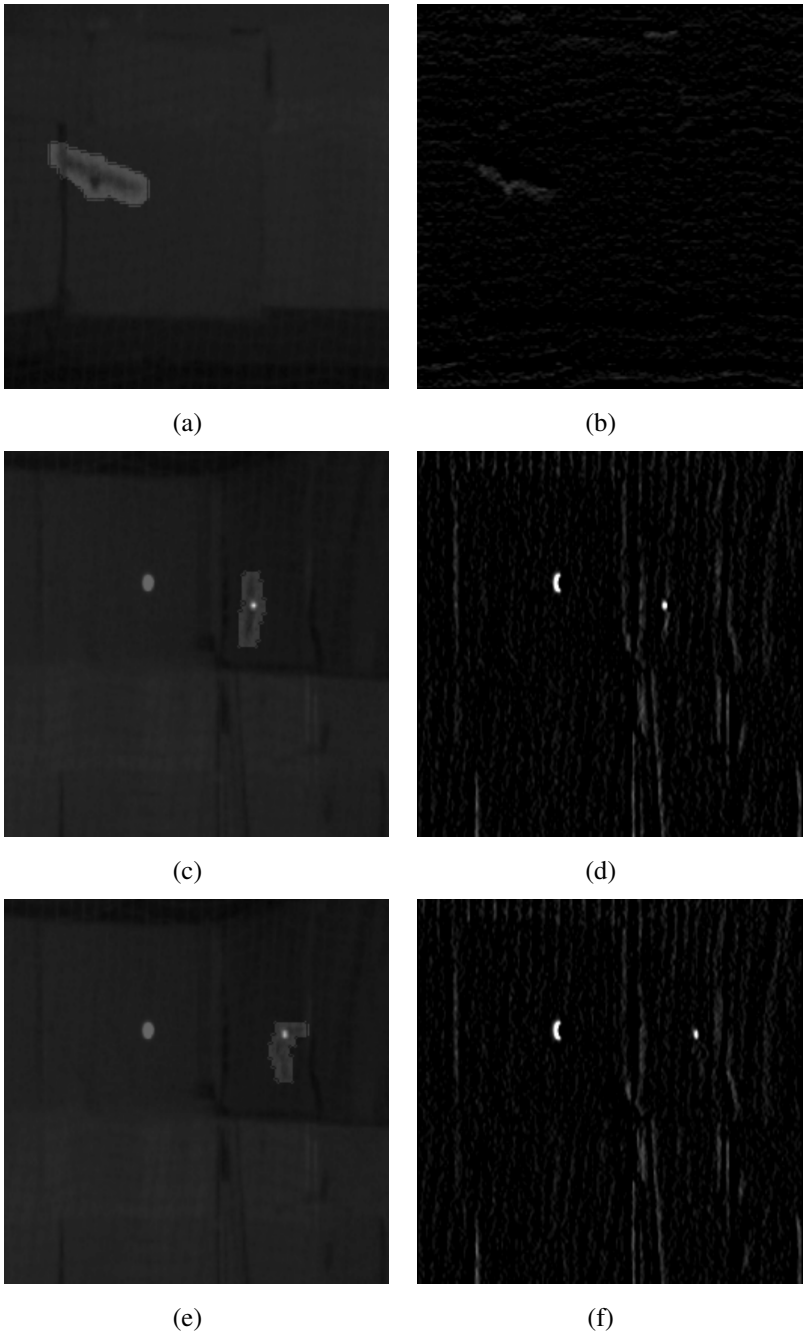


(a) (b)

(c) (d)

(e) (f)

Figure 2.4: Denoised images on the left and their corresponding residuals on the right column. The residual information is used to improve depth and sceneflow estimation. (a) , (b) Box filter. (c) , (d)FastNLMeans . (e) , (f) TV-L^1

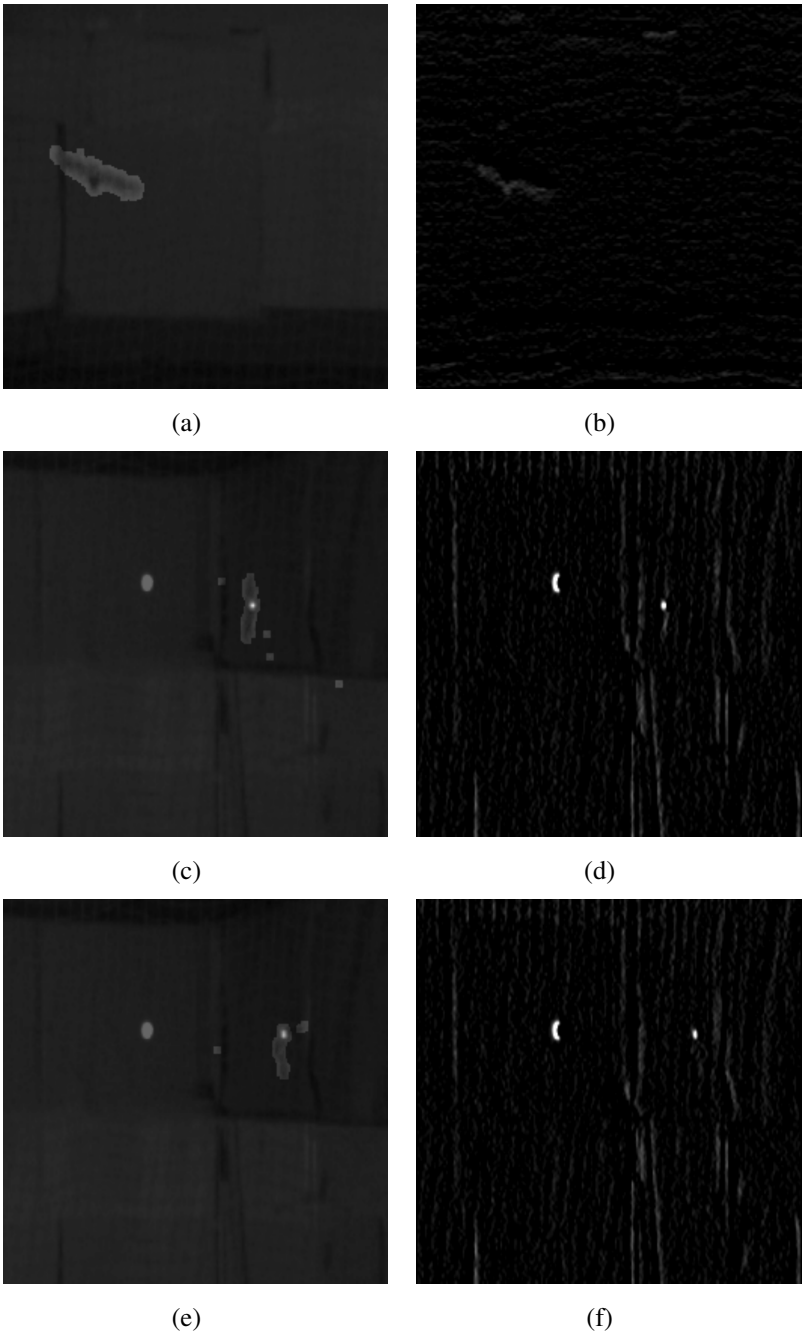


(a) (b)

(c) (d)

(e) (f)

Figure 2.5: Foreground segments extracted from half-size image on the left and $\text{Sobel}_{k=5}$ edges on the right column. Refer to text for details.



(a) (b)

(c) (d)

(e) (f)

Figure 2.6: Foreground segments extracted from full-size image on the left and $\text{Sobel}_{k=5}$ edges on the right column. Refer to text for details.

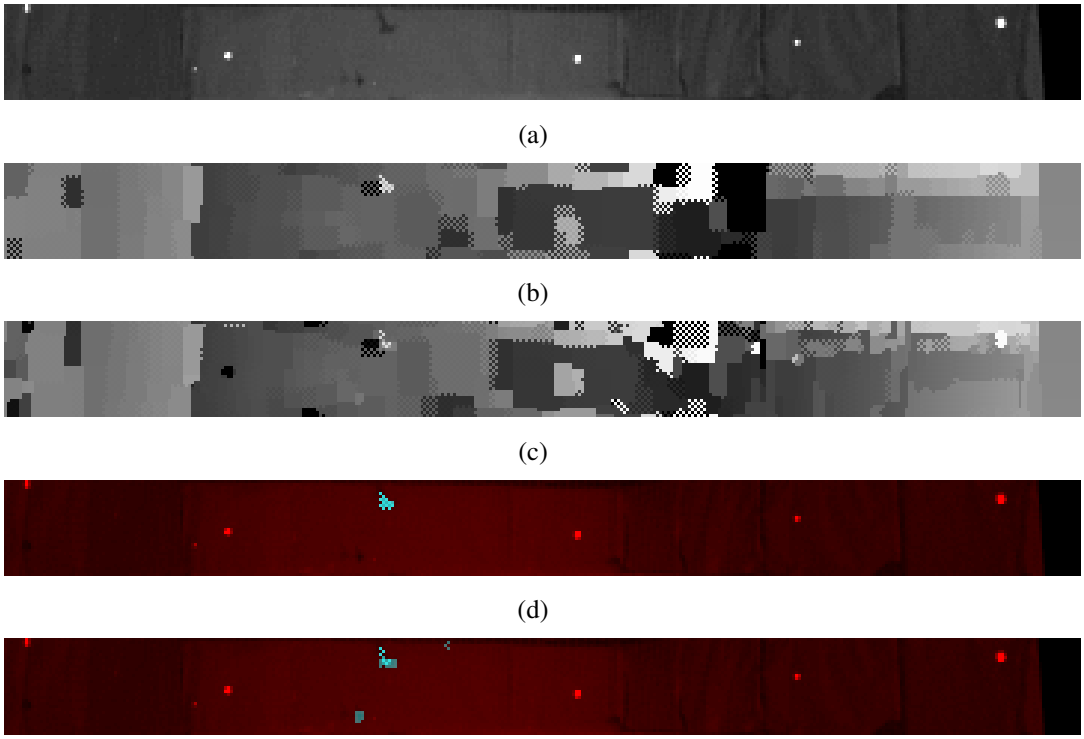
(a) (b)

Figure 2.7: Disparity map obtained using semi-global block matching algorithm [1] (a) and superposition top half of disparities on the corresponding image (b)

better results were achieved.

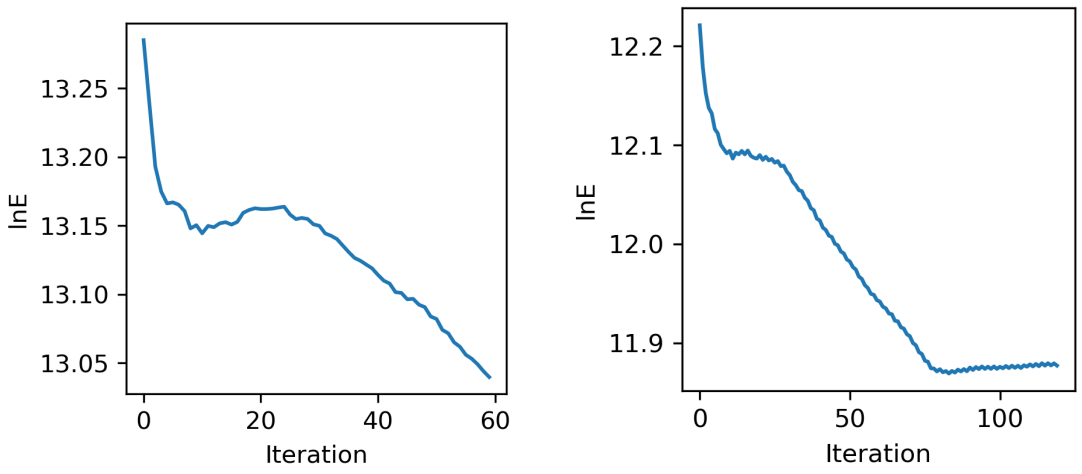
Thereafter, the obtained disparities are used to compute scene flow of the stereo images sequence. with regards to the execution of the scene flow algorithm, we perform few experiments to observe the influence of smoothing coefficient and computational efficiency of the algorithm. Even though the objective at the moment is not to optimize the execution efficiency of this algorithm, we intend to avoid intractable computation times. Of course with help of parallel processing, this would not be an issue. Here, we demonstrate the 2d results of scene flow obtained from full-size and half-size images. The angles are encoded with hue channel of HSV color model with intensities representing the magnitude. Then we adjust the coefficient of the data energy term to examine the noise artifacts. The results for coefficients 0.3 and 0.6 are shown in Figure 2.10. It is seen that with increase in the coefficient of data term moving components of the objects are become more distinguishable. However this adds unwanted noise to the results. Similar outcome is also observed in the results obtained from full-size image as well.

In the next stage, clusters of pixels based on their location and displacements in horizontal, vertical and depth components are identified. We register these component groups



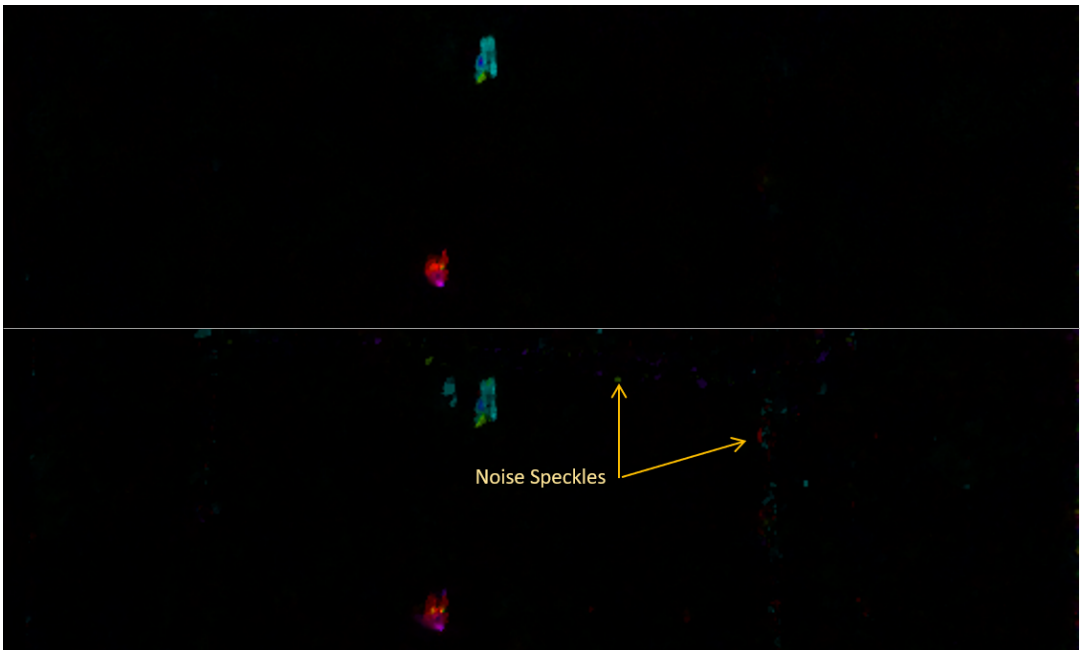
(a) (b) (c) (d) (e)

Figure 2.8: Left view image (a) Disparity maps $\lambda_{.} = 15$, $N_{iter} = 60$ (b) $\lambda_{.} = 8$, $N_{iter} = 120$ (c) $\lambda_{.} = 8$, $N_{iter} = 120$ Foreground segment disparities in cyan on the image obtained $\lambda_{.} = 15$, $N_{iter} = 60$ (d) $\lambda_{.} = 8$, $N_{iter} = 120$ (e)

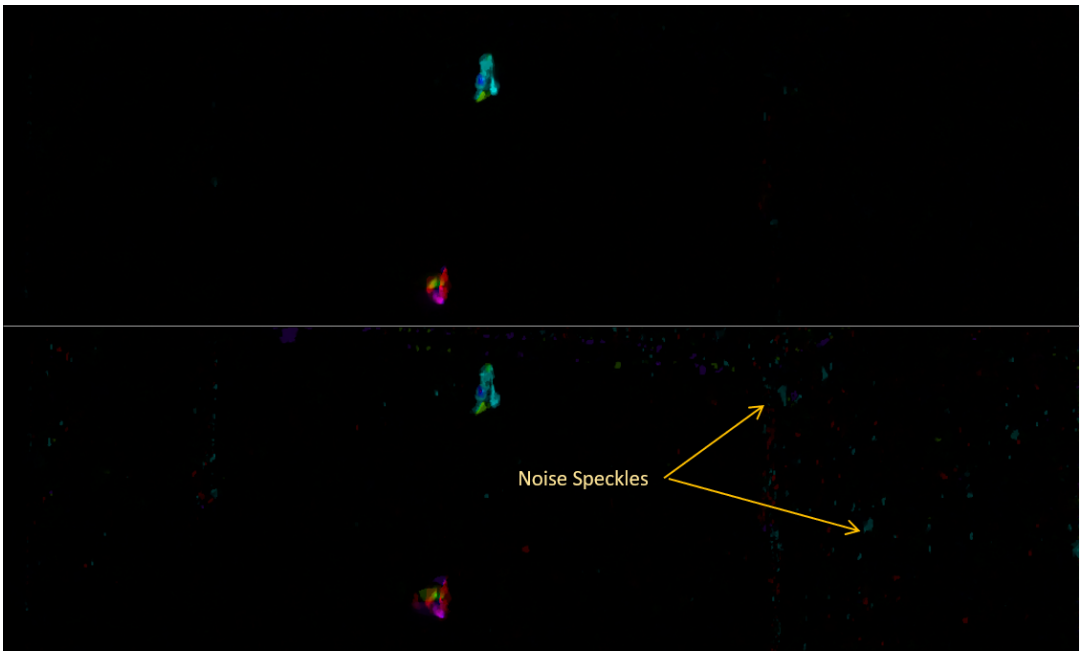


(a) (b)

Figure 2.9: Natural logarithm of energy values of the belief network versus number of iterations (b) $\lambda_{.} = 15.0$, $N_{iter} = 60$ (b) $\lambda_{.} = 8.0$, $N_{iter} = 120$

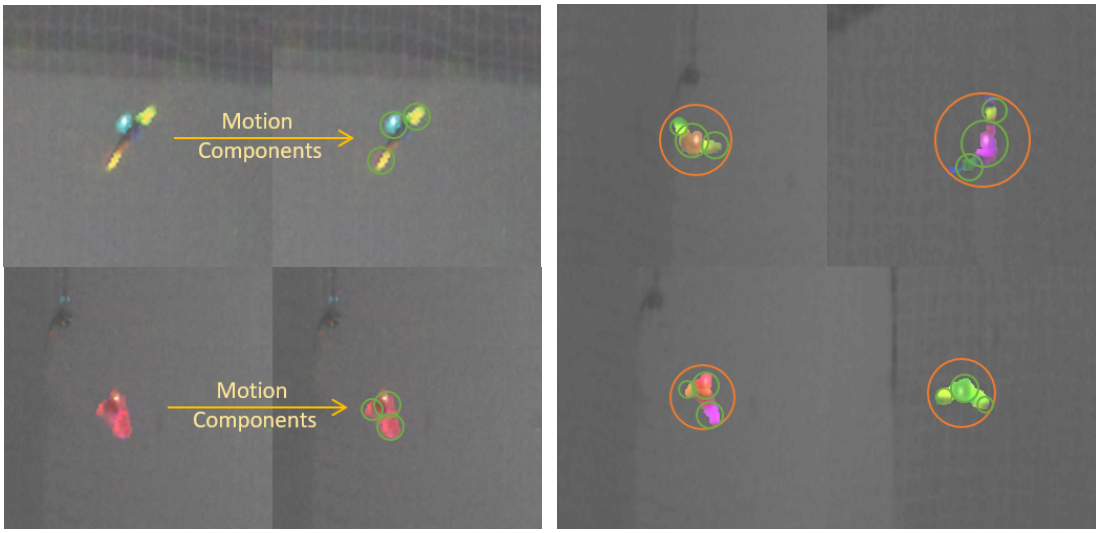


(a)



(b)

Figure 2.10: This figure shows the motion components of the full-scale and down-sampled to half-size images obtained using total variation loss with two different data coefficients 0.3 and 0.6 on top and bottom respectively. (a) Half-size image. (b) Full-size image.



(a) (b)

Figure 2.11: A demonstration of 2D motion component and groups. (a). Extraction of motion components from pixels in 6D. (b) Identification of groups of motion components.

and based on their 3D movement vectors computed using eq. 2.58. Figure 2.11 shows examples of moving components identified as a group. In Figure 2.11(a), it is shown that motion components are identified and extracted using a clustering method like KMeans. Then these components are aggregated within a range depending on the size of the bats as shown in Figure 2.11(b). These are registered as motion groups and are tracked. Given velocity and position of the groups and their components it is possible to use particle filters or ensemble Kalman filters to create a search area for subsequent frames. Given the location and movement parameters of the target object it becomes less prone to 2D occlusion failures.

To evaluate this method against other tracking techniques for this application, we have selected a number of prominent baseline tracking algorithms. These are Boosting [99], CSR-DCF (CSRT) [100], GOTURN [101], KCF [102], MedianFlow [103], MIL [104], MOSSE [105] and TLD [106]. Since the algorithm proposed here, directly measures the

depth, there is no requirement for depth estimation from tracked objects. However, the tracking algorithms track objects in both left and right images and the centers of the proposed rectangles are used to compute depth of the tracked objects. In addition, for directly comparing the tracking results of the proposed algorithm with the others in images, we projected the obtained 3D components on to left and right images and benchmark all algorithms in a same way. The evaluation scores are intersection over union (IOU) ratio of the bounding boxes and 2D Euclidean distance of their center points. IOU is computed as following:

$$IOU = \frac{R \cap R_{GT}}{R \cup R_{GT}}, \tag{2.61}$$

where R_{GT} is the ground truth bounding box. In this experiment, the ground truth position is determined by the maker position in the image. However, to accommodate the latter evaluation measure (IOU), a square bounding box centered at the marker location in the image designated as ground truth bounding box. Then, we run all algorithms on a same sequence with the exact preprocessing parameters. The computed precision and IOU values are considered separately for each of left and right camera images. Figure 2.12 shows few tracking results along with ground truth (GT) tracks. The proposed method in this chapter referred to as `SF` for the rest of this chapter.

In addition to precision and IOU values, tracking quality is also measured by the ratio frames having IOU or precision against some threshold values. These are illustrated as precision and success plots as shown in Figure 2.13 and Figure 2.14 for left and right camera images respectively. Average precision and IOU values are list in Table 2.1.

As seen in the table, the proposed method performed relatively better while, MIL was better in term of being within the set threshold for larger ratios. As stated earlier, to produce 3D trajectories from both left and right images should be valid. This is where other

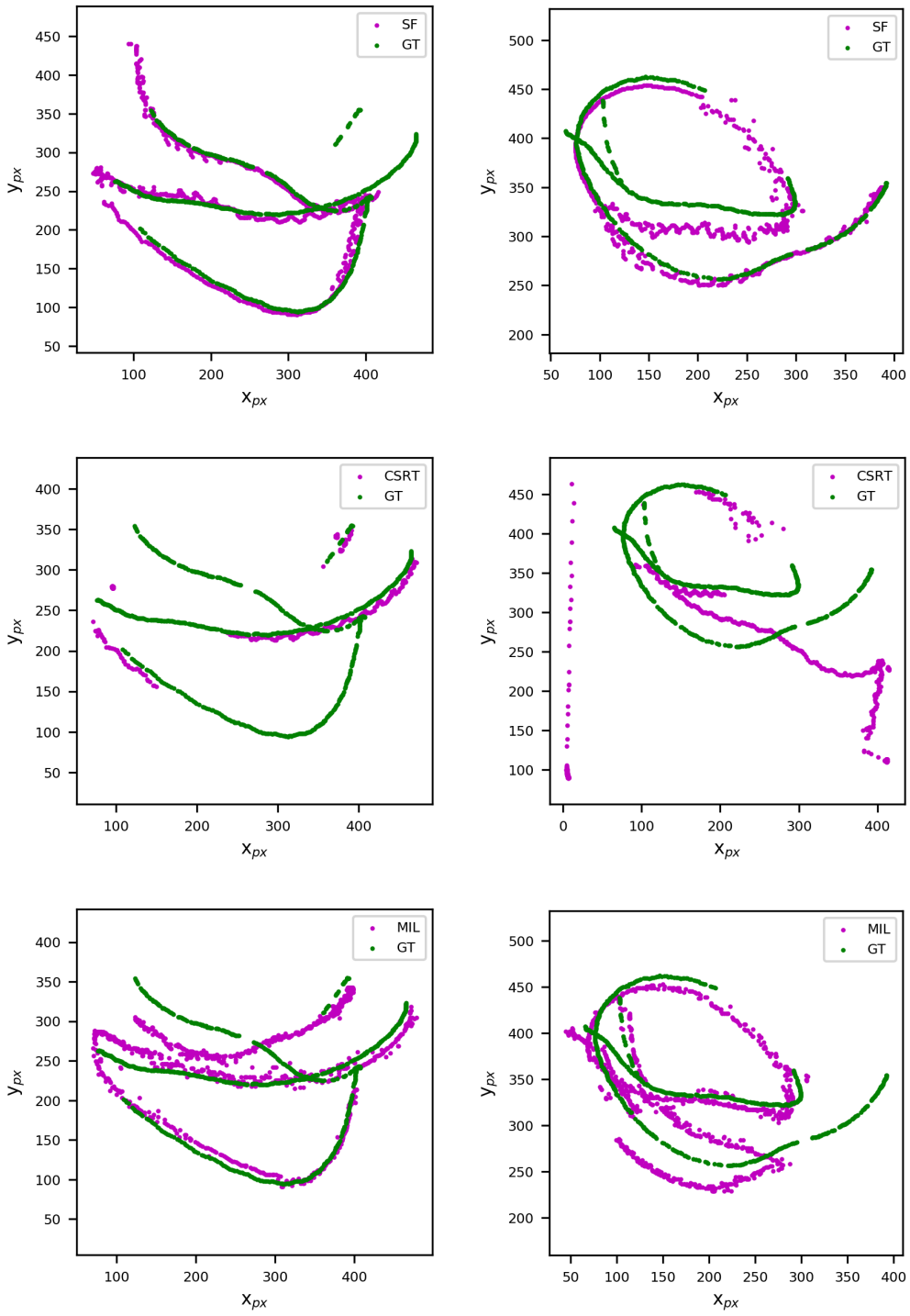
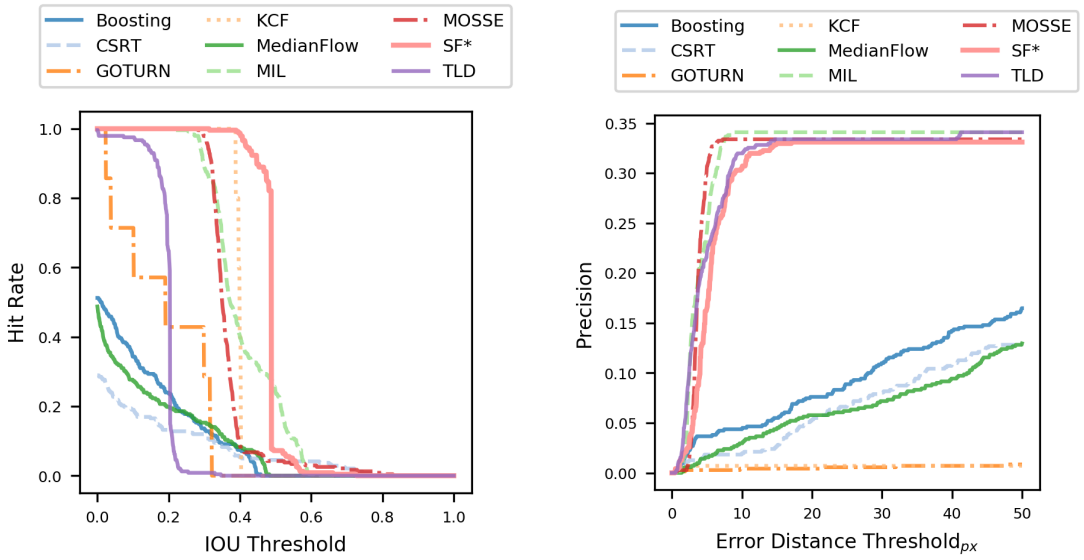
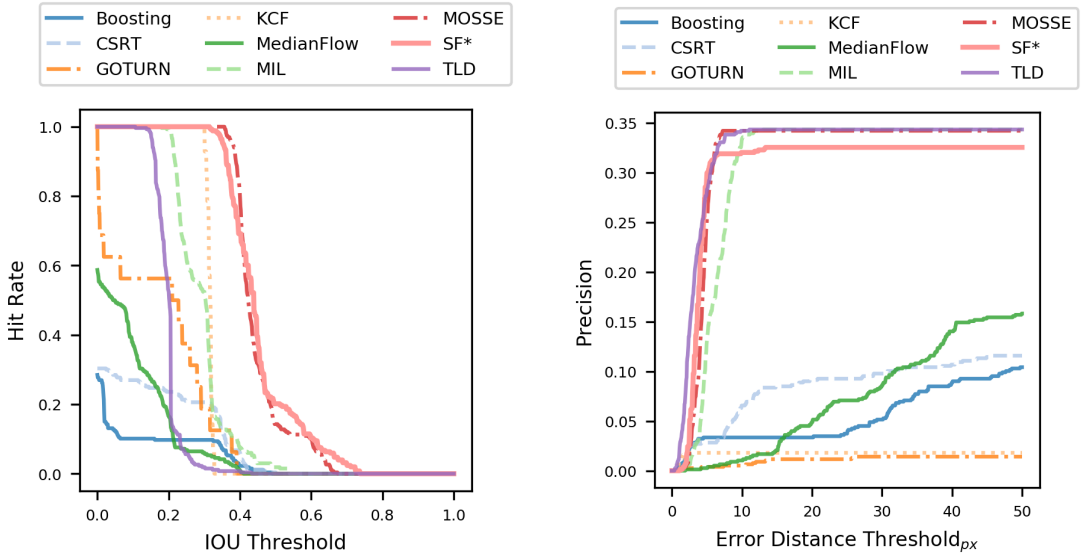


Figure 2.12: Tracked locations of two objects using the proposed algorithm, correlation based CSR-DCF and MIL trackers displayed along with the marker positions (GT).



(a) (b)

Figure 2.13: Benchmark plots for left camera image sequence with length of 400 frames. (a) Success plot. (b) Precision plot.



(a) (b)

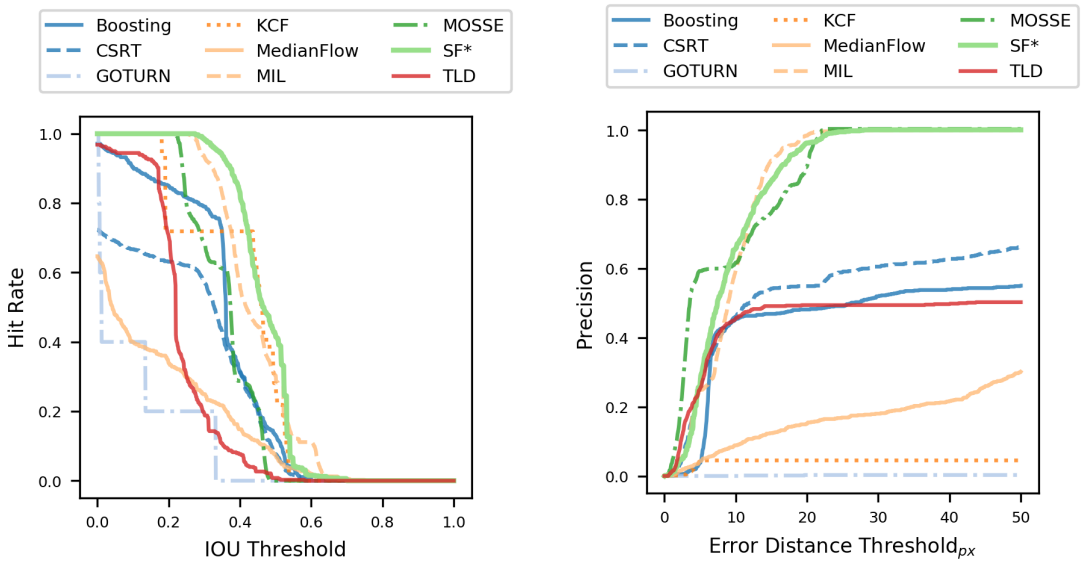
Figure 2.14: Benchmark plots for right camera image sequence with length of 400 frames. (a) Success plot. (b) Precision plot.

Table 2.1: Average results of precision and IOU in both camera images. In addition, the ratio of the frames that performance numbers were within the thresholds are listed.

Method	IOU	$IOU_{thr>0}$	$Precision_{(px)}$	$Precision_{thr<50}$
Boosting	0.0725	0.136	67.5	0.135
CSRT	0.0818	0.101	75.0	0.122
GOTURN	0.176	0.0155	239	0.0119
KCF	0.357	0.0128	**1.06**	0.0128
MIL	0.35	**0.342**	5.0	**0.342**
MOSSE	0.408	0.338	4.04	0.338
MedianFlow	0.0897	0.183	58.9	0.144
SF*	**0.47**	0.328	4.63	0.328
TLD	0.196	**0.342**	4.25	**0.342**

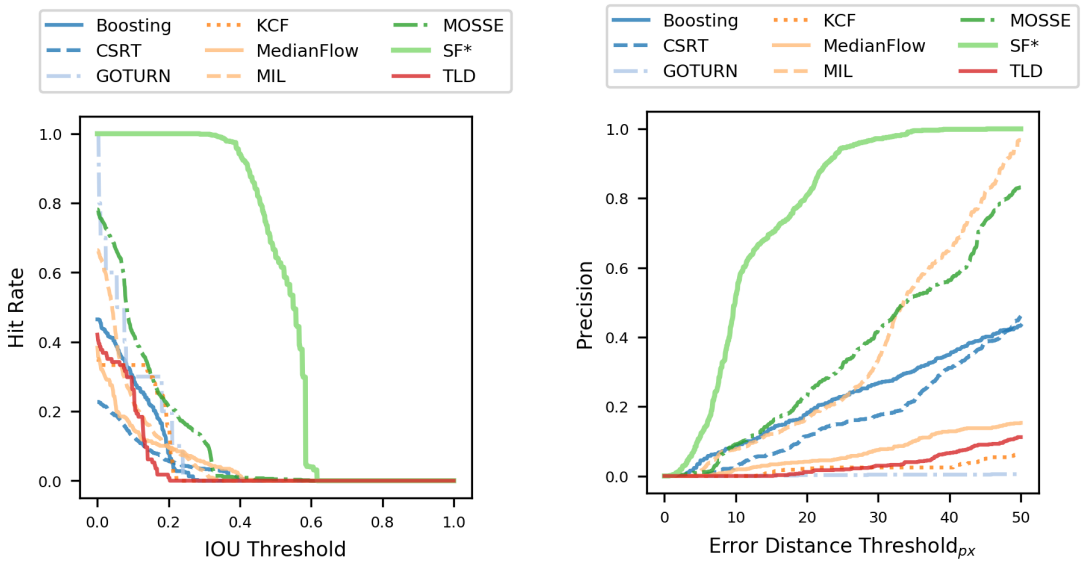
algorithms fall short. As illustrated in Figure 2.14, there is a discrepancy between the performance in the left and right images. Figure 2.15 and Figure 2.16 show the precision and success plots for longer sequence of 1000 frames. It is seen that the discrepancy issue persisted.

This discrepancy in performance between left and right frames produces invalid depth data and consequently, failing 3D reconstruction. This is while, in the proposed algorithm here, depth information is already available and it is also utilized for object identification and occlusion resolution. In simpler words, if trackers miss object position in either of images, the depth information would not be available for 3D trajectory reconstruction. Yet, another aspect, to which the high failure in 2D tracking could be attributed is highly nonlinear dynamics of flight of bats, specifically while approaching boundaries of the flight chamber. In the proposed method, 3D state predictors are employed which may experience less degree of nonlinearity or loss of information. This was clearly visible in the perfor-



(a) (b)

Figure 2.15: Benchmark plots for longer left camera image sequence with length of 1000 frames. (a) Success plot. (b) Precision plot.



(a) (b)

Figure 2.16: Benchmark plots for longer right camera image sequence with length of 1000 frames. (a) Success plot. (b) Precision plot.

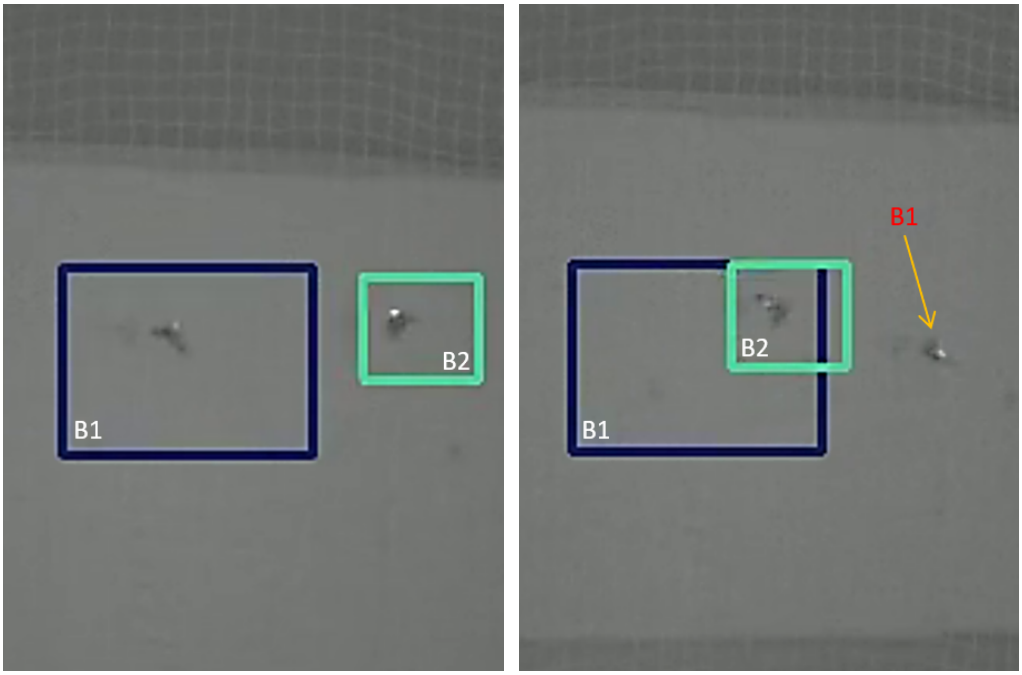
mances the 2D trackers in the experiments performed. They could perform fairly well while the targets flew across the screen where the change in depth was minimal. Ultimately, there are instances that all techniques fail altogether. These incidences occurred majorly at locations where appearance profiles of the objects diffused into the background and in case of the proposed algorithm, previous stages fail to estimate the foreground segments and their corresponding depth estimates. This situation could be remedied using more advanced state predictors. Given the dynamics of the target in 3D, it would be more plausible to track a highly maneuverable targets like bats.

Most of the trackers failed the correct registration of objects which their projected trajectories on the screen crossed. Since these trackers do not have access to depth information, they need to account for other indicators or information to identify the individual objects after the crossings. While with access to depth information this would not be the case other than instances, in which objects physically collide or closely fly past another. Figure 2.17 demonstrates few examples of such incidents.

Lastly, in the final phase, 3D coordinates tracked points was reconstructed. Since not all trackers successfully registered corresponding points in both images, best results were chosen for demonstration. Figure 2.18 illustrates instances of reconstructed tracks in 3D and Table 2.2 lists the means of Euclidean distance errors of the reconstructed trajectories in relative to the marker position in 3D for a test sequence with length of 400 frames, in 176 of which ground truth measured.

2.5 Conclusions

In this chapter, we have presented a modular multi-stage algorithm which is able to estimate and reconstruct the 3D motion of animals with low visibility in their environment.



(a) (b)

Figure 2.17: Failure of MIL tracker after path crossing. (a) Before the crossing, two objects are tracked individually as $B1$ and $B2$. (b) After the crossing, both trackers $B1$ and $B2$ track the same object failing to track both objects in the seen.

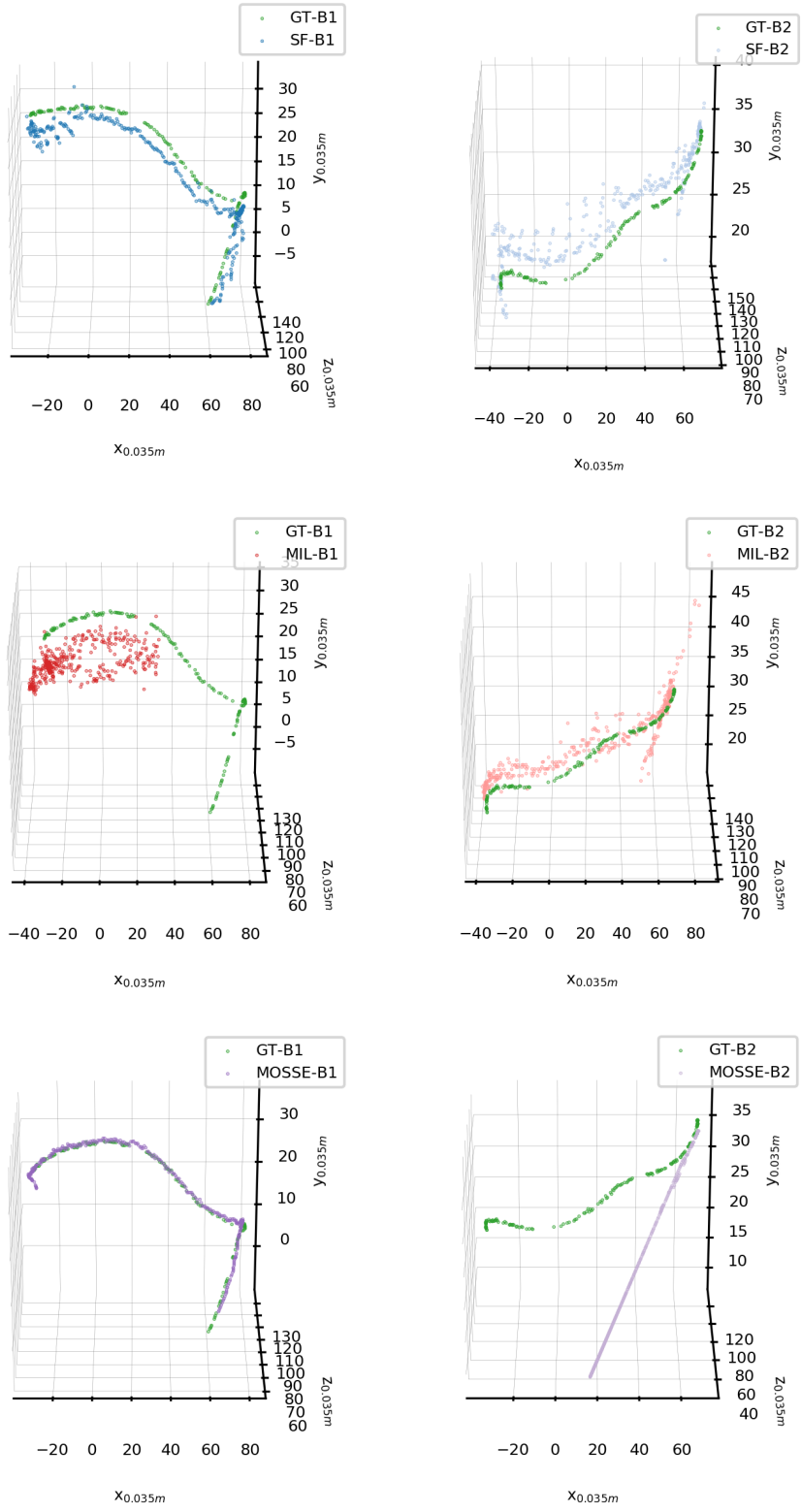


Figure 2.18: Reconstructed trajectories of two flying bats $B1$ and $B2$ using the proposed method and other top 2 performing algorithms. It is seen that the proposed method, SF, successfully kept tracking objects after the path crossing in camera images.

Table 2.2: Average, standard deviation (SD) and median of Euclidean distance error relative to the marker for a test sequence of 400 frames are listed per target and in overall.

Target	Method	Mean Err_{35mm}	SD Err_{35mm}	Median Err_{35mm}
B1	MIL	52.322	41.987	30.780
	MOSSE	10.391	26.173	**1.941**
	SF*	**7.720**	**17.673**	3.715
B2	MIL	13.745	27.972	**3.898**
	MOSSE	57.132	31.937	69.690
	SF*	**7.909**	**17.615**	4.027
All	MIL	33.034	40.466	9.960
	MOSSE	33.762	37.332	8.432
	SF*	**7.815**	**17.590**	**3.910**

This method possesses farther more potentials for identifying and distinguishing targets based on their dynamical properties. Since, the movement model of the target objects estimated in 3D, and with the assumption of semi-rigid motion, this method is less susceptible to the common failures in methods tracking in 2D. We shown that this method is capable of tracking the moving parts of objects that are with little or without any texture features. Another upside to this approach is that, it is possible to substitute the method in each stage according to the objectives. In other words, if the goal is tracking texture rich objects easily distinguishable in their habitat, we could choose a less computationally taxing methods for estimating the disparity and motion fields. Another advantage here is the ability to bypass the first stage of the algorithm given a known number of objects are already being tracked with a known simplified model. With this information, it is possible to propose the view hull of the object as foreground segment. However, all these come at the cost of complexity of variational methods and belief networks. They do not guarantee global optimums and

in certain problems, they are likely to fail. Computational cost could be improved with new parallel hardware capabilities and efficient implementations. This work also could be basis for the future works on pose estimation based on the motion components in semi-rigid body. One could imagine by imposing movement constraints concerning the physical or rigid properties of the target, it would be possible to estimate feasible solutions for the specific pose of the object.

Chapter 3

Modeling Latent Structures in Trajectories

3.1 Introduction

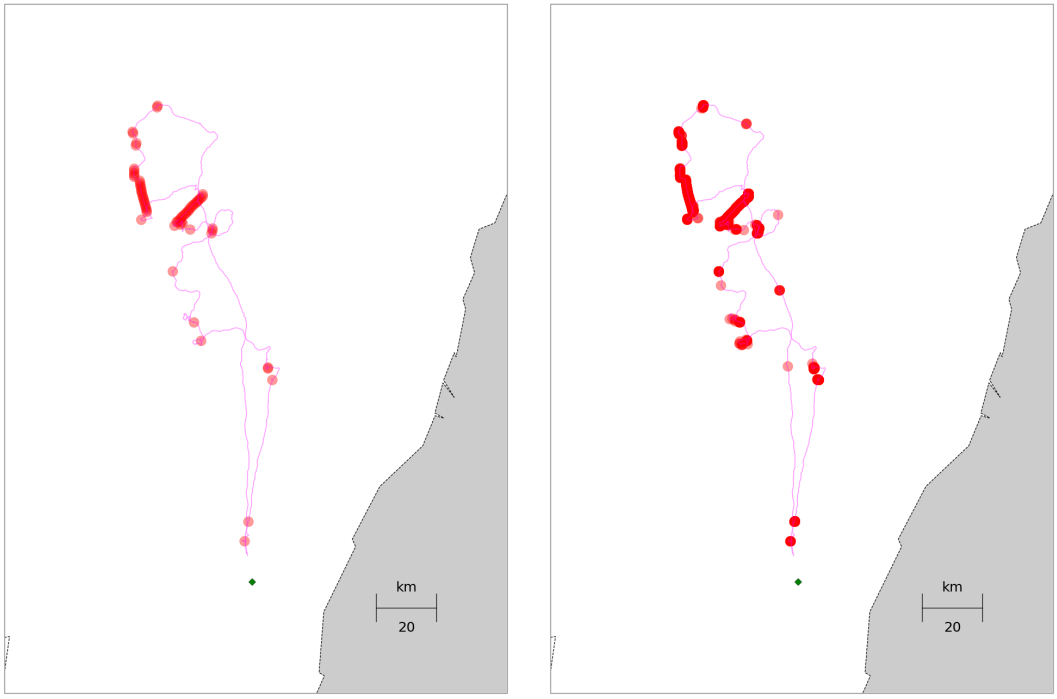
In this chapter, we look at animal movement from a novel perspective. We assume that certain trajectory points are the results of persistent behavioral attributes in focal organism's movement. Therefore, through identification of these points, trajectories containing these points could be associated to those behavioral characteristics which in the end may be linked to a specific individual or group. These points are labeled as key points in a trajectory. In this chapter we also present techniques for extracting these key points as well. It should be noted that extraction of these key points is also a consequential task since part of the modeling lies with the key point extraction method. In other words, the nature and semantics of key points are tightly intertwined with the modeling objectives of the algorithm. This is elaborated further later in this chapter. The other attribute of the algorithm presented here is that we only consider trajectory points in spatial dimensions ignoring

their chronological order. This is based on the assumption that the latent factors we are interested in are time invariant. This suggests that there is no concern about non-uniform sampling of data points or variable lengths of trajectories. Furthermore, this algorithm is greatly scalable to large data sets. In summary, the main contributions of this chapter are in three fronts. The first is feature extraction by proposing a semantical feature extraction method for movement data. The second is feature representation by adopting a technique for construction of numerical vectors of extracted features. The third is modeling by presenting techniques for discriminating or generating movement features. The main results of this chapter published in [107]. In the following section we review a few required basic technical concepts prior to delving into mechanics of the proposed algorithm.

3.2 Preliminary Concepts

3.2.1 Stay Point

As mentioned in Chapter 1, not all spatial points in trajectories carry semantics or at least not with uniform weights. In part of literature these points are termed as "Stay Points". As implied from the title, these points identify the points along trajectory that focal object remained stationary or within a threshold displacement for a certain time. There a number of algorithms for detecting stay points in trajectory mining literature [108, 109, 110]. In general given trajectory points sequence $\mathbf{P} = \{\mathbf{p}_{t_0}, \mathbf{p}_{t_1}, \ldots, \mathbf{p}_{t_n}\}$, points in contiguous segment $\mathbf{S} = \{\mathbf{p}_{t_i}, \mathbf{p}_{t_{i+1}}, \ldots, \mathbf{p}_{t_{i+l}}\}$ of P identifies a stay point if $dist(\mathbf{p}_{t_{i+l}} - \mathbf{p}_{t_i}) < \epsilon$ and $|t_{i+l} - t_i| > \delta$. $dist(.)$ determines the spatial distance between two points. Figure 3.1 shows few stay point examples.



(a) (b)

Figure 3.1: (a) Sample stay points extracted form a shearwater trajectory with time parameter of $10min$ and spatial threshold of $500m$ indicated by red $\circ$. (b) Trajectory segments with speeds below $2.5m/s$ are indicated by red $\circ$

3.2.2 Density-based Clustering

Clustering could be identified as a reduction method whose objective is to describe a data set with a defined number of homogeneous groups. The measure of pair-wise homogeneity or contrast should be defined for all members of data set. Clustering methods are classified based on their approach for modeling the data groups. The most common ones, that are based on centroid and distribution, called k-means [111] and Gaussian mixture model (GMM) [112]. With these methods, there should be a number of groups specified in prior to the process of clustering or the process is repeated for a number of times to arrive at the best fit. This feature make them inefficient for applications where the objective itself is detecting the number of underlying groups or modeling the density is part of the problem. For this group of problems, density-based clustering methods like DBSCAN [113] or mean-shift are used. However, It should be noted that in case of spatial data, DBSCAN is mostly preferred due to its well-defined cluster model, ability to classify the noise in data rather than simply partitioning it, and its linear complexity. DBSCAN has a number variants which in essence are simply extensions or generalizations of DBSCAN[114, 115, 116].

DBSCAN starts with an arbitrary point and detects all density-reachable points from it. The term `density-reachable` is defined as neighbor points within a threshold distance ϵ. If number of density-reachable points from the selected point is above a predetermined number n, this point is a `core point`, otherwise it is labelled as noise. In case of the core point, it is registered as a new cluster. Then all its density-reachable points and their density-reachable neighbors are assigned with the same label if there were tested previously. Figure 3.2 demonstrates examples of different cluster models BIRCH [117], DBSCAN, k-means and spectral clustering [118], used for a sample geospatial data. For models with required initial guess, the number set to 6. DBSCAN's minimum neighbor

set to 3, haversine distance metric and threshold distance of 5km were used. BIRCH's branching factor set to 10 with threshold of 0.001 radians.

3.2.3 Dirichlet, Categorical and Multinomial Models

Categorical distribution is the extension of Bernoulli distribution for modeling outcome of an event with more than two possible classes. In other words, categorical distribution describes a distribution over any number of outcomes. A categorical random variable is represented by a vector of K dimensions for K categories and each entry denotes the probability of the outcome belonging to the respective category. These entries are sum to unity. Probability mass function of categorical distribution is as:

$$Cat(x|\boldsymbol{\Theta}) = \prod_{k \in K} \theta_k^{\mathbb{I}_{k=x}}, \quad \sum_{k \in K} \theta_k = 1,\ \theta_k \geq 0 \tag{3.1}$$

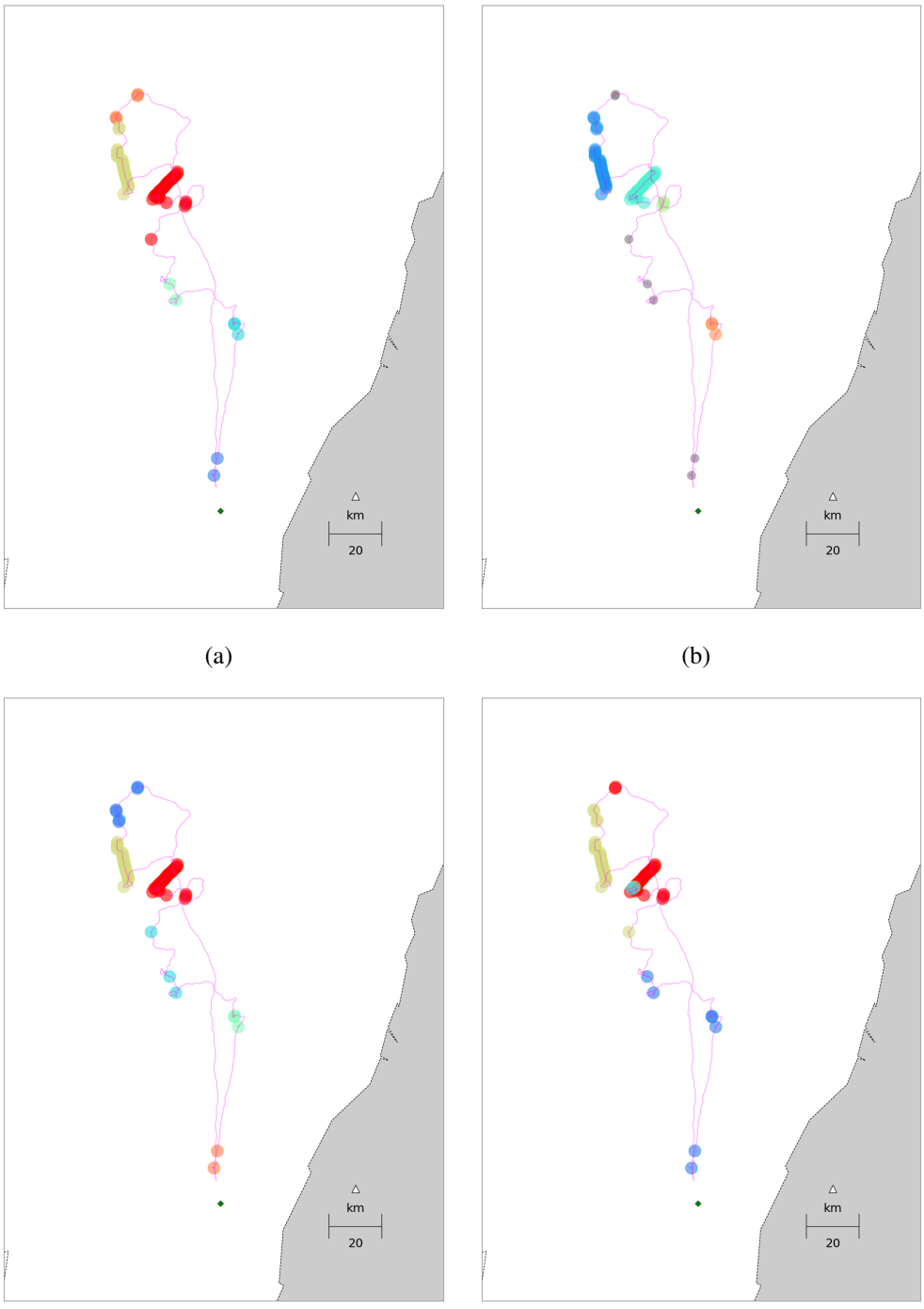
where $\boldsymbol{\Theta}$ is $K-$dimensional parameter vector of probabilities for each category and $\mathbb{I}_{\cdot}$ is indicator function which returns unity if equality condition holds true. Multinomial distribution is generalization of the categorical distribution where it is repeated for more than one trial and defined as:

$$Multi(\mathbf{x}|\boldsymbol{\Theta}) = \frac{N!}{\prod_{k \in K} x_k!} \prod_{k \in K} \theta_k^{x_k}, \quad \sum_{k \in K} x_k = N,\ \sum_{k \in K} \theta_k = 1,\ \theta_k \geq 0 \tag{3.2}$$

where N is total number of trials, $\mathbf{x}$ is a vector of length K denoting the number of outcomes being of each category and elements sum to N.

Prior distribution of categorical distribution's parameters could be modeled with Dirichlet distribution of order K as:

$$Dir(\boldsymbol{\Theta}|\boldsymbol{\alpha}) = \frac{\Gamma(\sum_{k \in K} \alpha_k)}{\prod_{k \in K} \Gamma(\alpha_k)} \prod_{k \in K} \theta_k^{\alpha_k - 1}, \quad \sum_{k \in K} \theta_k = 1,\ \theta_k \geq 0 \tag{3.3}$$



(a) (b)

(c) (d)

Figure 3.2: Identified stay point clusters using (a) BIRCH (b) DBSCAN (gray points are designated as noise.) (c) K-means (d) Spectral

where $\boldsymbol{\Theta}$ is $K-$dimensional vector of positive values sum to unity ($K-1$ simplex) and $\boldsymbol{\alpha}$ is vector of concentration parameters with the same dimension. $\Gamma(.)$ is gamma function. The posterior distribution could be estimated for a set of N data vectors $\mathbb{X}$ as following:

$$\begin{aligned} P(\boldsymbol{\Theta}|\mathbb{X},\boldsymbol{\alpha}) &\propto P(\mathbb{X}|\boldsymbol{\Theta})P(\boldsymbol{\Theta}|\boldsymbol{\alpha}) \\ &\propto \prod_{n\in N}\prod_{k\in K}\theta_k^{\mathbb{I}_{k=x_n}}\prod_{k\in K}\theta_k^{\alpha_k-1} \\ &= \prod_{k\in K}\theta_k^{n_k}\prod_{k\in K}\theta_k^{\alpha_k-1} \\ &\propto Dir(\boldsymbol{\Theta}|\mathbf{n}+\boldsymbol{\alpha}) \end{aligned} \tag{3.4}$$

where $\mathbf{n}$ is vector of observed counts of each category in the data set.

Finally, the predictive probability of $\hat{x}$ given above is:

$$\begin{aligned} P(\hat{x}|\mathbb{X}) &= \int Cat(\hat{x}|\boldsymbol{\Theta},\mathbb{X})P(\boldsymbol{\Theta}|\mathbb{X},\boldsymbol{\alpha})d\boldsymbol{\Theta} \\ &= \int \theta_{k=\hat{x}}Dir(\boldsymbol{\Theta}|\mathbf{n}+\boldsymbol{\alpha})d\boldsymbol{\Theta} \\ &= \frac{n_k+\alpha_k}{N+\sum_{k\in K}\alpha_k} \end{aligned} \tag{3.5}$$

3.3 Methodology

3.3.1 Problem Formulation

As suggested in [2] the navigation capacity and internal state of organism are viewed part of cognitive. The relationship between consecutive locations in trajectory formulated as:

$$u_{t+1} = F(\Omega,\Phi,r_t,w_t,u_t) \tag{3.6}$$

where u_t, u_{t+1} are consecutive positions at times t, $t+1$, w_t is internal state, r_t being environmental factors, Φ is navigation capacity and Ω is motion capacity which further explained

in Chapter 1. The formulation models the progression of movement in time as a dynamical process. Therefore, time is considered primary variable here and dynamical constraints govern the order of the points along trajectory. In other words, there is strong dependence between consecutive points in trajectory. These constraints could be relaxed with dropping time as the primary variable and approaching trajectories in spatial and spectral domains. Trajectories are projected on to spatial plane and represented by points based on their spectral attributes. As a result, trajectories are represented as set of K key points with semantical relevance and could be assumed conditionally independent of each other given certain movement feature. Figure 3.3 demonstrates graphical models for sequential and conditionally independent representations for trajectory points. By removing sequential dependence, exchangeability assumption [119] could be considered and bag-of-words model become applicable. The other implication of this approach is that the destination exercises a stronger semantical meaning than the path traversed to reach the destination.

Similarly, in sentiment analysis the bag-of-words model is assumed where the ordering of the words in a sentence is neglected. This may not always produce great results in more combinatory vocabularies but in general it is simplistic and practical [120, 121]. Yet another advantage of key points representation of trajectories is that it reduces the effect of transient environment factors on composition of trajectory key points. It is assumed, in general, animal compensates for such deviations caused by environmental factors and it is only apparent in generated path and time taken to the intended destination, while the location of destination remains the same [13].

Conceptually, to analyse the behavioral states using trajectories, they could be converted into series of key points which are assumed to be exchangeable in order. As though, any permutation of the key points is similarly likely. The hypothesis is that common or

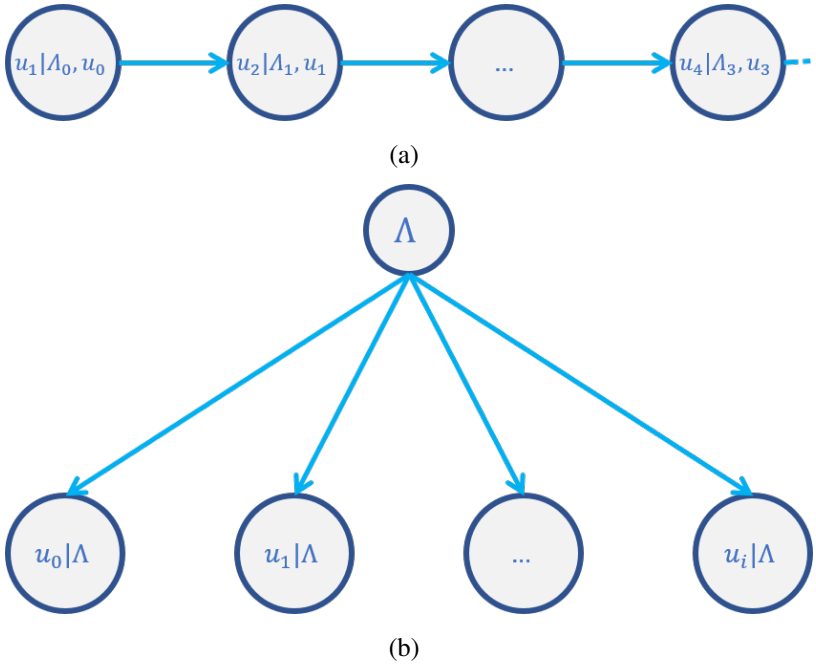


Figure 3.3: Movement graphical models (a) with sequential chain dependencies (b) Assumption of conditional independence and exchangeability. $\Lambda = [\Omega, \Phi, \mathbf{w}, \mathbf{r}]$.

similar internal states tend to generate similar key points to comprise a trajectory which could be interpreted as a particular behavior expression prevalent in trajectories. This enables us to classify these behaviors or latent states responsible for these behaviors using discriminative models or even span trajectories with such attributes using generative models. It should not be overlooked that this approach is only applicable for modeling time invariant or time independent features or characteristics. Hence, it is not expected to be able to encode the temporal dynamics of movements or behaviors or at least, within the integrated time window used for key point extraction. In other words, to model temporal behaviors using this approach, key points could be extracted over shorter time partitions and modeled independently, given their temporal partition parameters.

3.3.2 Key Point Extraction

Stay point detection algorithms are commonly used to extract points of interest from trajectories. As focal organisms remained at proximity of specific locations for a minimum time period, these points could carry semantics such as foraging, sleeping in case of animals and shopping or dining in case of people. These stay points then clustered to discover the hubs with higher significance [109]. However, in case of animals, the spatial range and time threshold for stay point detection may vary. A simpler approach to detect such key points given that logged data sampled in a relatively fixed frequency, it is safe to assume that low speed movement along trajectory creates higher spatial densities. By stacking all data on spatial plane, locations with frequent low speed crossings would automatically create high density regions that would form cluster using density-based clustering methods. As a result, trajectories could be reduced to a set of these clusters. Figure 3.4 illustrates extracted key points extracted using stay point detection method and speed threshold techniques.

3.3.3 Trajectory Representation

In order to employ numerical methods, the key points and trajectories should be represented as numbers or numerical vectors. There are different approaches to this problem, but here spectral methods inspired by text mining used in this work. K Key points in trajectories represented by tokens with integer ids. The simplest feature vectors could be just binary vector of token memberships in trajectories. The trajectory t_n is represented by a $K-$dimensional vector $\mathbf{t}_n$ as:

$$\mathbf{t}_n = [\mathbb{I}_{k \in t_n} | k \in K]^T \tag{3.7}$$

where $\mathbb{I}_{k \in t_n}$ returns one if key point k is member of trajectory t_n otherwise returns zero. A slightly more sophisticated one is key point frequency which is the number of occurrences

of these tokens in a trajectory. In this manner numerical representations of trajectories of variable lengths with fixed length vectors is constructed as:

$$\mathbf{t}_n = [\sum_{i \in t_n} \mathbb{I}_{k=i} | k \in K]^T \tag{3.8}$$

where $\sum_{i \in t_n} \mathbb{I}_{k=i}$ is key point k's frequency in trajectory t_n. So, by selection of K key points, corpus of trajectories could be represented as a matrix of $K \times N$ where N is number of trajectories in data set. In other words, corpus is of N samples of $K-$dimensional feature vectors. Using key point frequency may not also be the most efficient way of representing a trajectory. There are key points that are universally shared between all trajectories frequently. These point carry very little discriminative semantical information as a results. To encounter this, inverse trajectory frequency factor inspired by [122] is also considered. The trajectory representation vector using a smoothed variant [123] is described as:

$$\mathbf{t}_n = [\alpha_k \log \frac{1+N}{1+\beta_k} + \alpha_k | k \in K]^T \tag{3.9}$$

$$\beta_k = \sum_{n' \in N} \mathbb{I}_{k \in t_{n'}} \tag{3.10}$$

where α_k is key point k's frequency and β_k determines the frequency of trajectories sharing it. The latter two representation vectors are further normalized by Euclidean norm.

3.3.4 Generative Modeling

In this study, a modeling approach inspired by sentiment analysis is used to infer, extract and quantify subjective information about internal state and attitude of focal organism. The latent structure could be modeled as generative distributions which produce the key points in a trajectory. This is referred to as "Topic Modeling". For instance, latent Dirichlet allocation (LDA) could be a probabilistic model which intuitively explains the distribution

of behaviors or document topics [124]. Given a random mixture of hidden states in animal movement, trajectory key points are distributed according to a latent state specific distribution in the expressed movement. This could be designed as two series of Dirichlet distribution draws, one for latent states and the other for state specific key points, and two series multinomial distribution draws for latent state assignment to each key point and trajectory assignment to each key point. Parameter estimation is commonly performed using inference methods.

To model trajectories using LDA [125], H particular group of latent states are assumed which is analogous to number of topics in documents. As mentioned earlier, each of the states has a multinomial distribution over the all key points as in text processing where topics define distributions over the words in vocabulary. Prior for these multinomial distributions is $\mathbf{\Pi}$ and, each $\boldsymbol{\pi}_h$ drawn from Dirichlet with hyperparameter of $\boldsymbol{\alpha}$. Hence, to generate the trajectories, given trajectory t, $\boldsymbol{\theta}_t$ is drawn from Dirichlet distribution with hyperparameter $\boldsymbol{\beta}$. Next, for each key point k_t of trajectory t, latent state $h_{t,k}$ drawn from $Multi(\boldsymbol{\theta}_t)$ and given $h_{t,k}$, k_t is drawn from $Multi(\boldsymbol{\pi}_{h_{t,k}})$. The generative process is summarized as following:

1. For each latent state h, draw $\boldsymbol{\pi}_h \sim Dir(\boldsymbol{\alpha})$, $h \in H$
2. For each trajectory t_n, draw $\boldsymbol{\theta}_t \sim Dir(\boldsymbol{\beta})$, $n \in N$
3. For each key point k in t_n:
   (a) Draw $h_{t,k} \sim Multi(\boldsymbol{\theta}_t)$
   (b) Draw $k_t \sim Multi(\boldsymbol{\pi}_{t,k})$

Referred to as "Multinomial Principle Component Analysis (PCA)" in [126], this model would provide an unsupervised approach to analyze trajectories. Hence this can discover

latent structure of trajectory collections which is beneficial for both prediction and data exploration. The posterior distribution is described as following:

$$P(h, \boldsymbol{\Theta}, \boldsymbol{\Pi} | k, \boldsymbol{\alpha}, \boldsymbol{\beta}) = \frac{P(h, \boldsymbol{\Theta}, \boldsymbol{\Pi} | \boldsymbol{\alpha}, \boldsymbol{\beta})}{P(k | \boldsymbol{\alpha}, \boldsymbol{\beta})} \tag{3.11}$$

and could be estimated using methods like variational inference or Markov Chain Monte Carlo (MCMC) sampling or Gibbs sampling [125]. In our experiments we used an online variational inference method developed by Hoffman et al. in [126]. The variational objective derived to rely only on key point frequency per trajectory which carries the intuition of document summary based on word counts. Clearly this is applicable in case of trajectories where frequency of the key points could potentially summarize the latent states of the focal organism.

3.3.5 Discriminative Modeling

Discriminative approach could be utilized to identify the polarity of the expressed behaviors in trajectories. This involves a family of discriminative models which classify the trajectories based on the extracted key points. This approach is analogous to sentiment analysis on text documents. A great example is Twitter sentiment analysis which was topic of numerous studies. [127, 120]. Most of these methods assume bag-of-words model, which as mentioned previously in probability theory referred to as exchangeability assumption [119], that ignores the ordering of the words in documents.

For discriminative modeling, the options for off-the-self classifier are abundant. Naïve Bayes model intuitively explains the probabilistic relationship between hidden states and key points. Considering their simplicity and strong prior assumptions they tend to work very well in this sort of problem settings [128]. Using Bayesian theorem, hidden category

probability could be modeled as:

$$P(h|t, \mathbf{\Theta}, \mathbf{\Pi}) = \frac{P(h|\mathbf{\Pi})P(t|h, \mathbf{\Theta})}{\sum_{h \in H} P(h, t, \mathbf{\Theta}, \mathbf{\Pi})} \tag{3.12}$$

where $\mathbf{\Theta}$ and $\mathbf{\Pi}$ are parameters of the model, H is the number of hidden state categories, t is a trajectory in set T of all trajectories and K_t is set of key points in trajectory t. A priori is modeled as a categorical distribution with parameter $\mathbf{\Pi}$ as:

$$P(h|\mathbf{\Pi}) = \prod_{h' \in H} \pi_{h'}^{\mathbb{I}_h} = \pi_h, \ \sum_{h' \in H} \pi_{h'} = 1 \tag{3.13}$$

where $\mathbf{\Pi}$ has H dimension with L_1-norm of unity. Hence π_h could be interpreted as prior probability of h. Likelihood of a trajectory given h could be modeled as multinomial distribution as:

$$P(t|h, \mathbf{\Theta}) = \prod_{k \in K_t} P(k|h, \mathbf{\Theta}) \sim Multi(\mathbf{\Theta}, K) \tag{3.14}$$

where we assume key points are conditionally independent given hidden state h. This could be interpreted as each state has its own distribution over key points. Therefore, a generic form of naïve Bayes classifier could estimate maximum likelihood of a hidden state given key points as:

$$\arg\max_{h \in H} \pi_h \prod_{k \in K} \theta_{k,h} \tag{3.15}$$

where pi_h and $theta_{k,h}$ could be determined using maximum likelihood estimation (MLE) method which results in:

$$\pi_h = \frac{n_h}{\sum_{h' \in H} n_{h'}} \tag{3.16}$$

and,

$$\theta_{k,h} = \frac{n_{h,k}}{n_h} \tag{3.17}$$

where n_h and $n_{h,k}$ are counts of trajectories with hidden state labeled as h and key point k associated with that hidden state respectively. in practice, smoothed version of these

equations are used in which, smoothing prior constant α and αK are added to numerator and denominator respectively. Maximum a posteriori (MAP) estimate is also applicable where Dirichlet priors are assigned to the parameters. Since, Dirichlet distribution is conjugate prior of the multinomial distribution, the results are similar to the smoothed version of MLE method's solutions.

Other than naïve Bayes classifier, more complex models like support vector machines (SVM), ensemble and boosted trees could be utilized to predict hidden states based on extracted feature vectors. It is also possible to further improve the performance of naïve Bayes, SVM and tree classifiers by calibrating membership probabilities [129]. Since these classifiers are not optimized on the prediction probabilities, they often produce biased class probabilities. These biases are dependent on the method. Zadrpzny et al. [129, 130] proposed remedies to calibrate the class probabilities of the naïve Bayes classifiers using histogram methods and tree classifiers using smoothing, curtailment, Kearns-Mansour splitting criterion, and isotonic regression. In this experiment, non-parametric and parametric methods, isotonic regression and Platt scaling[131, 132] respectively, are used to calibrate the class probability of all classifiers. Since, Isotonic Regression is very prone to overfitting, k-fold cross-validation is used in training procedure.

To evaluate the performance of the classifiers, in addition to accuracy, Matthews correlation coefficient (MCC)[133], precision, recall and F_1-measure are employed. MCC is defined as following:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP) \times (TP + FN) \times (TN + FP) \times (TN + FN)}}. \tag{3.18}$$

This score is regarded as balanced measure [134] and weighs performance in true and false positives and negatives (TP,FP,TN,FN). Precision and recall are calculated as:

$$precision = \frac{TP}{TP + FP} \tag{3.19}$$

$$recall = \frac{TP}{TP + FN} \tag{3.20}$$

Finally, F_1-measure is computed as:

$$F_1 = \frac{2 \times precision \times recall}{precision + recall} \tag{3.21}$$

3.4 Experiments, Results and Discussion

Our experiment was performed on a species of seabirds off the cast of Japan called Streaked Shearwater Calonectris leucomelas. The data set was provided by Yoda-lab [135]. The trajectories recorded using logging devices attached to 271 birds belonging to 112 unique nests. These nests are located at two colonies on the east and west coasts of Japan. A quarter of all birds under study belong to the colony on the west. In addition to nest and colony information, gender of the birds is also available. This information could be used as labels in supervised modeling of trajectories based on the gender or colony of the birds. As a result, inferring the state differences in generation of trajectories is plausible. Gender of the birds identified by their gender specific vocal features [136]. The gender distribution in each colony is approximately uniform.

Here in this experiment, two types of key points extracted from trajectories. One based on stay point definition and the other based on key point extraction procedures discussed in the previous section. In the second method speed threshold used for speed was $5km/s$. For stay point detection, range threshold of $500m$ and time threshold of $10min$ was used. Then, DBSCAN is used to extract densities of points in $1km$ radius neighborhood with minimum of 30 neighbors. Densities with minimum of 10 unique bird ids were selected as key points set K. This number is essential to performance of classifiers as greatly influences the size and utility of the key points set. For instance, for increasing the spatial coverage in case

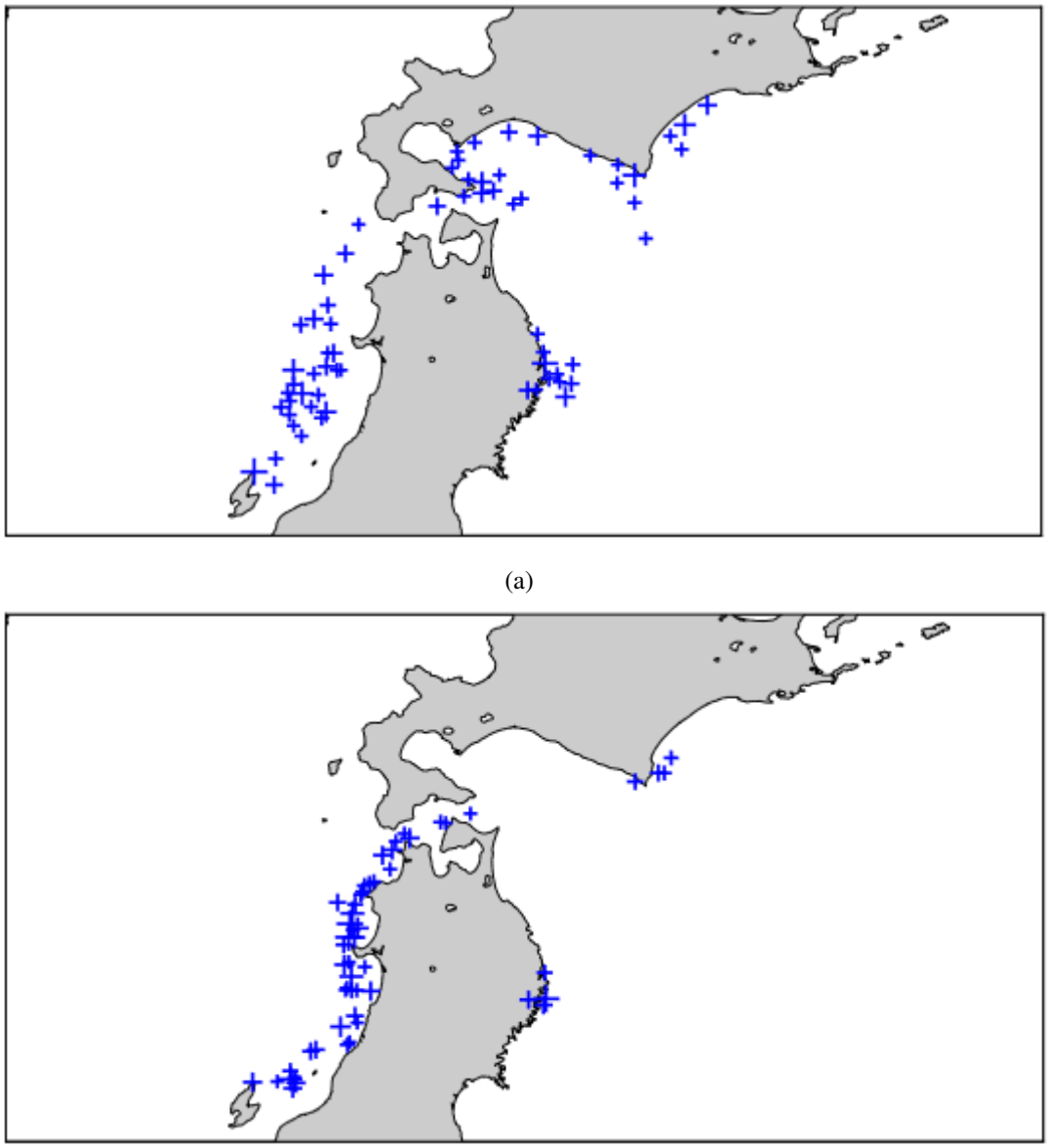
Table 3.1: Test accuracy of top performing vanilla classifiers

Key Point Type	Classifier	Test Acc (%)
Speed Threshold/ DBSCAN	Bernoulli naïve Bayes	62.75
	Gradient Boost	**70.59**
	Ada Boost	68.63
Stay Point/ DBSCAN	Bernoulli naïve Bayes	66.67
	Gradient Boost	**72.73**
	Ada Boost	66.67

of stay points, this number was reduced to 5. As a consequence, the results of classifiers improved significantly. Figure 3.4 shows the results of key point extraction procedures. Size of the extracted key points set is 65 for speed threshold extraction method and 155 for stay point detection method. This also has an effect on the number of trajectories that are possible to be processed. If a trajectory does not contain any of the key points in the key points set, it should be discarded. Therefore, in key point extraction step, these notes should be considered and appropriate hyperparameter values should be used. Figure 3.5 shows the distribution of birds over the selected key points.

Having key points set, each trajectory encoded to a sequence of key points. Subsequently, the spectral feature matrices of trajectories was created. To have a rough estimate of classification performance, vanilla classifiers trained on 80% of the data set and tested on the rest. Table 3.1 demonstrates the outcome.

It is seen that accuracy of the test predictions ranged between 60% – 70% with boosted trees resulting in top performances. These results simply show that trajectory key points carry information about gender of the birds. So, to improve the performance, we have attempted tuning of class probabilities. As described previously, Platt's sigmoid [131] and



(a)

(b)

Figure 3.4: Extracted key point which has minimum of 10 unique bird ids using (a) speed threshold and (b) stay point method. Larger cross signs show higher number of unique bird ids contained in the key point cluster

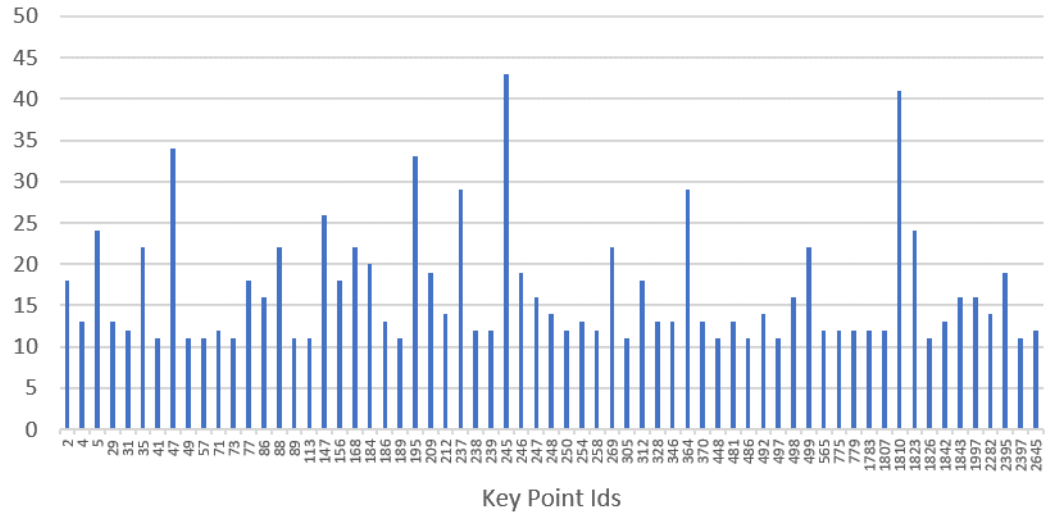
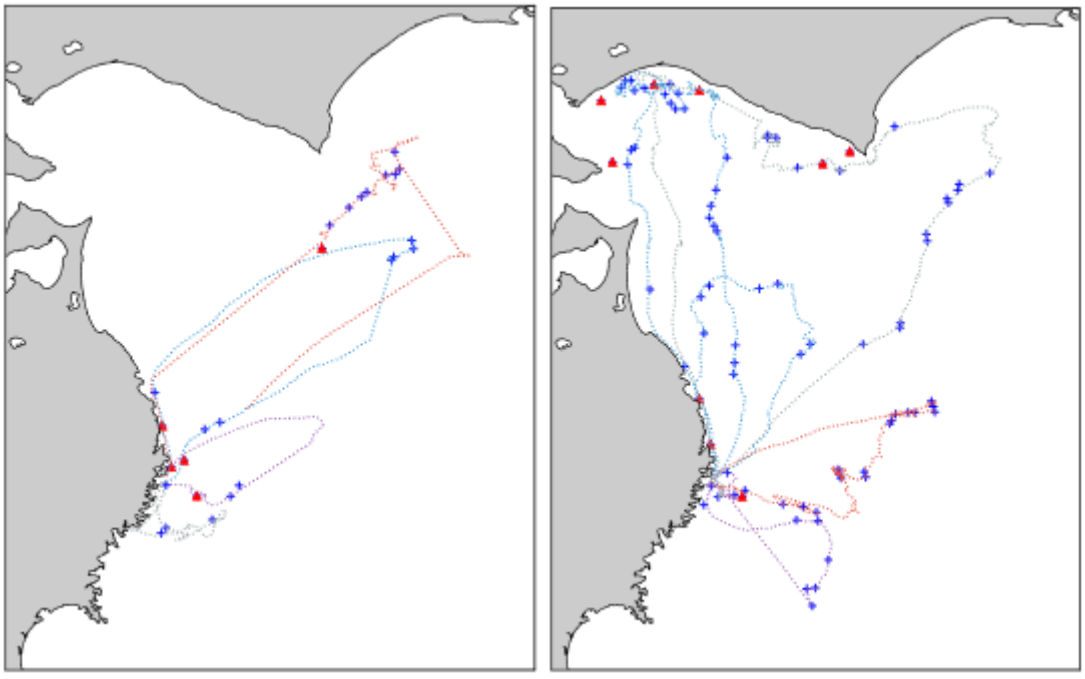


Figure 3.5: Histogram of unique bird ids at key points extracted using speed threshold method



(a) (b)

Figure 3.6: Sample of trajectories. Identified speed threshold DBSCAN clusters marked by blue + and identified vocabulary key points marked by red △. (a) Female trajectories (b) Male trajectories.

Table 3.2: Test results of top performing tuned classifiers for speed threshold/ DBSCAN key points.

Classifier	MCC	Precision	Recall	F1	Accuracy
Logistic	0.3297	0.6786	0.7037	0.6909	0.6667
Naïve Bayes	0.2542	0.6538	0.6296	0.6415	0.6275
Naïve Bayes + Isotonic	0.3297	0.6786	0.7037	0.6909	0.6667
Naïve Bayes + Sigmoid	0.2542	0.6538	0.6296	0.6415	0.6275
SVM	0.2485	0.6250	0.7407	0.678	0.6275
SVM + Isotonic	0.3723	0.6667	0.8148	0.7333	0.6863
SVM + Sigmoid	0.2173	0.64	0.5926	0.6154	0.6078
GBC	0.4923	0.7188	0.8519	0.7797	0.7451
GBC + Isotonic	**0.5000**	0.7059	**0.8889**	**0.7869**	**0.7451**
GBC + Sigmoid	0.4876	**0.7500**	0.7778	0.7636	0.7451
ABC	0.1673	0.5938	0.7037	0.6441	0.5882
ABC + Isotonic	0.3287	0.6667	0.7407	0.7018	0.6667
ABC + Sigmoid	0.0599	0.5833	0.2593	0.359	0.5098

isotonic regression [130] methods were used to tune class probabilities. Since isotonic regression is prone to overfitting, tuning used 10-fold cross-validation on 80% of data set. The tuned classifier then tested on the rest of the data. The performance scores of classifiers is shown in Table 3.2 and 3.3 for speed threshold based and key point based input features respectively.

It is observed that tuning of the class probabilities certainly improved the performance of the most of classifiers. It should be noted that, one of the significant limiting factors of decision models are lack of standard key point set. This problem would be surely less of an issue with increase in volume of data. The same classifiers also trained and tested on key points extracted using stay point detection technique. It is seen that the performance is

Table 3.3: Test results of top performing tuned classifiers for stay point/ DBSCAN key points.

Classifier	MCC	Precision	Recall	F1	Accuracy
Logistic	0.3699	0.6857	0.8276	0.75	0.6923
Naïve Bayes	0.2088	0.6364	0.7241	0.6774	0.6154
Naïve Bayes + Isotonic	0.251	0.6563	0.7241	0.6885	0.6346
Naïve Bayes + Sigmoid	0.228	0.6667	0.6207	0.6429	0.6154
SVM	0.2892	0.6667	0.7586	0.7097	0.6538
SVM + Isotonic	0.2452	0.6389	0.7931	0.7077	0.6346
SVM + Sigmoid	0.228	0.6667	0.6207	0.6429	0.6154
GBC	**0.413**	**0.6944**	**0.8621**	**0.7692**	**0.7115**
GBC + Isotonic	0.2892	0.6667	0.7586	0.7097	0.6538
GBC + Sigmoid	0.237	0.68	0.5862	0.6296	0.6154
ABC	0.0387	0.5769	0.5172	0.5455	0.5192
ABC + Isotonic	0.2048	0.6286	0.7586	0.6875	0.6154
ABC + Sigmoid	0.1659	0.6176	0.7241	0.6667	0.5962

slightly below the results achieved on the other set of key points.

Surely, to improve the objective of this experiment which is gender classification, we could also use motion capacity features. These features are also showing the physical differences between two genders. Figure 3.9 shows the recorded speed distribution over genders. The results of statistical hypothesis testings are shown in Table 3.4 as well. The disparity between the distribution of values belonging to each gender is substantial in Awashima colony (a-colony) while not as much in the other. Therefore with proper utilization of these features, the classification results could be improved further. Using speed as input feature, increases the stability of the classification results significantly.

Since the main objective of this study is to explore information capacity of the ex-

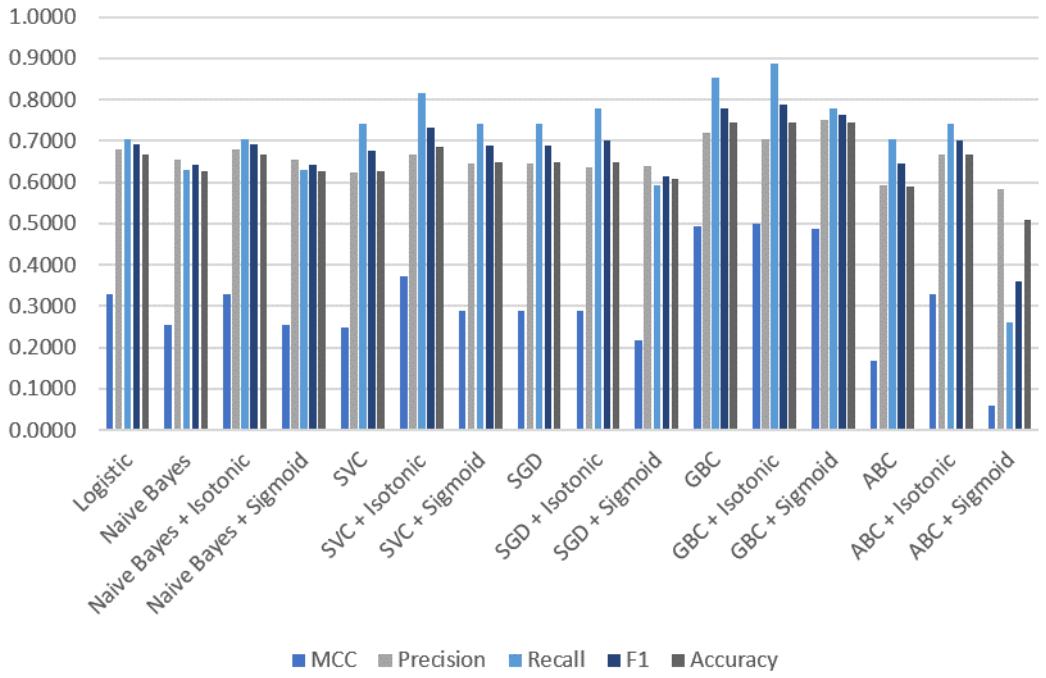


Figure 3.7: Performance results of tuned classifiers for key points extracted using speed threshold

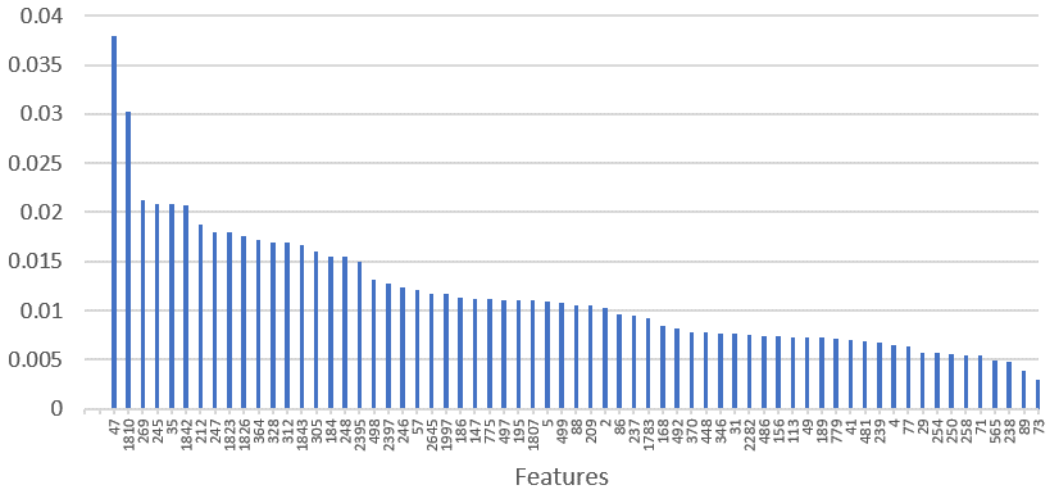


Figure 3.8: Spatial features importance for gender classification.

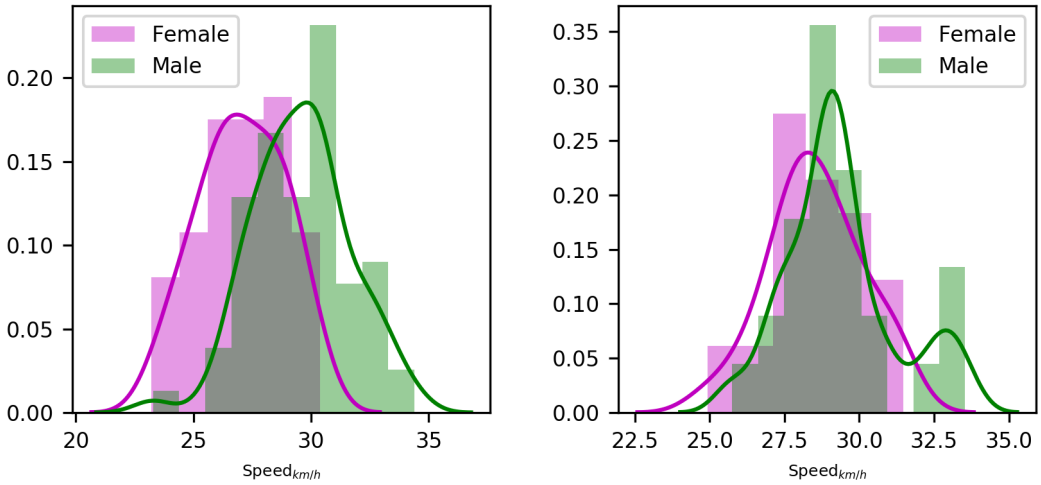


Figure 3.9: Distribution of travel speeds for birds of each gender (a) Birds of a-colony (b) Birds of t-colony.

Table 3.4: 2-sample tests for the null hypothesis of same speed distribution of male and female birds from colonies Awashima (a) and Iwate(t)

Populations	Test	p-value
$Male_a - Male_t$	Kolmogorov–Smirnov	0.9965
	t-test*	0.7026
$Female_a - Female_t$	Kolmogorov–Smirnov	$2.6_{\times 10^{-5}}$
	t-test	$1.4_{\times 10^{-5}}$
$Female_a - Male_a$	Kolmogorov–Smirnov	$2.7_{\times 10^{-8}}$
	t-test	$2.4_{\times 10^{-12}}$
$Female_t - Male_t$	Kolmogorov–Smirnov	0.0633
	t-test	0.0296

* 2 samples t-test [137].

tracted key points regarding the latent states of the focal organisms, further analysis on the features and classification results were performed. To dig more about the effectiveness of

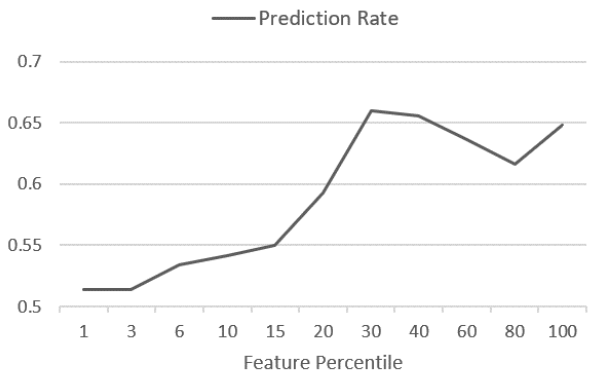
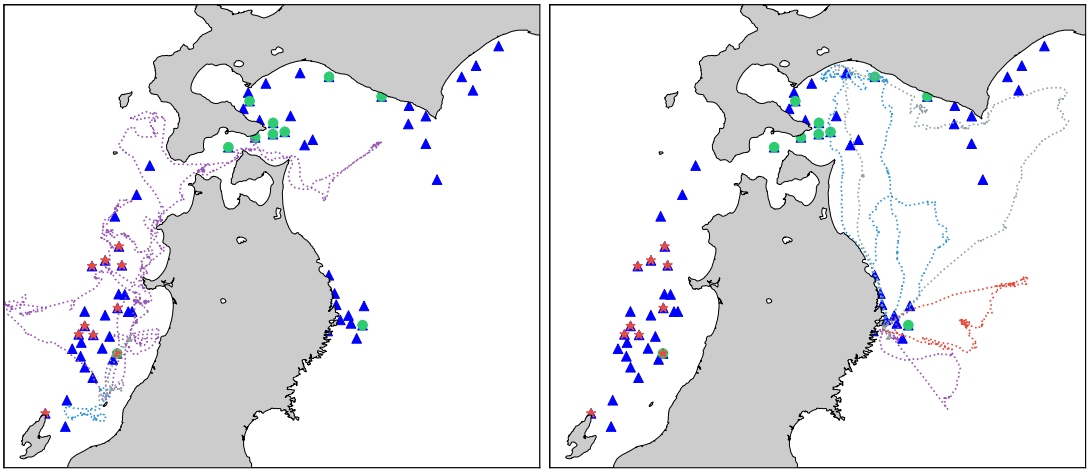


Figure 3.10: Plot of selected feature percentile versus prediction rate.

key points inclusion as input to classifiers, a feature selection method using mutual information, utilized to examine the performance of the logistic regression against the percentile of selected features. Figure 3.10 shows the results. As seen in the plot, inclusion of more features between 20 to 30 percentiles improve the classifier performance significantly while the performance peaks at 30 percentile does not improve further considerably.

As mentioned earlier, it is suspected that other than gender, there other factors that could be attributed to the latent state of the animal. This is examined by taking advantage of variational Bayesian encoder model. Here, stochastic LDA model trained using the online method purposed in [126] to identify the major components in the data set. This is used like PCA to identify the features contributing to the principle components. To create a comparable generative model to the discriminative model, number of components for LDA chosen to be 2. Then top key points contributing to the components were extracted. Figure 3.11 demonstrates the top key points of the extracted components using LDA, along with trajectories of birds of different gender and habitat and key points set. It is evident that one component has more key points shared with the sample male bird trajectory even



(a) (b)

Figure 3.11: Vocabulary key points marked by blue △. First component's top 10 key points marked by green ∘. Second component is marked by red ☆. (a) Female bird trajectories belonging to a-colony (b) Male bird trajectories belonging to t-colony .

though the birds with different genders from separate habitats have common regions in their trajectories. It can be perceived that although not with strong margins, key points are gender segregated within species trajectories. This should be reasonable as there are many more internal and external factors which also affect the path propagation process of birds.

3.5 Conclusions

In this section, we have proposed an alternative look at behavioral analysis using trajectories. The temporal constraints in the study of the trajectories were relaxed and trajectories represented by groups of independent key points. This independence assumed to be conditioned on internal states, motion capacities or navigation capacities of the organisms under study. This conditional independence allows treatment of trajectories in only spatial dimensions. Analogous to sentiment analysis and text processing, bag-of-words model could be applied, which given sufficient size of training data and ordinary language model, evidently

produces acceptable results.

One very important element of this procedure is key point identification and extraction. This means identification of major semantics. There is still room for more studies here, which could lead to introduction of new kernel methods for fast and reliable detection of the key points. One notable takeaway from our experiment is that, after key point extraction process the key points lose their spatial relations and their similarities must be measured in a different space. As though two key points spatially close to each other may not carry the similar semantics. Therefore, new points to data set must be clustered again or a projection function must be designed to transform the new points to semantic space of the key points. An instance is that, even though a foraging area and nests are spatially close, they belong to two distant points in semantical space. This encourages use of kernel-based methods for transformation of the trajectories spatial points to semantic space. Furthermore, the concept of n-gram is also applicable here to model more complex relation between consecutive points in trajectories.

Unfortunately, there are disadvantages to the approach as well. One as with most of data driven modeling methods is that, its dependency on data set and potential lack of generality. As it was seen in the experiment, size of trajectory data set and the class balance had considerable effects on the performance. However, this is likely to be handled to acceptable margins by increase in size of trajectory data set. Carefully selection of training data is also helpful. As mentioned earlier, key point extraction methods also play a very essential role in fitness of the models and key point extraction methods rely on the data set features like sampling rate, sparseness, etc. For example, directly applying a clustering method on the points as used in this experiment is highly dependent on sampling rate of the trajectories. Though, the same thresholds may not be applicable in other cases with much lower sam-

pling rates. One last potential downside to the introduced approach is lack of generality in generated models for species of different attributes. This is also held true in the most language and text processing methods. Learning models for a certain language, does not necessarily translate into information about other languages. We will propose solutions to some of the stated problems in in the following chapters.

The overall conclusion is that, this approach is efficient to semantically compare data inputs with variable length. It is also open to other semantic methods like negative matrix factorization and tensor factorization.

Chapter 4

Context-based Semantical Vectors for Modeling Latent Structures

4.1 Introduction

In this chapter we present a method for compact numerical and semantical representation of key points in animal trajectories. In previous chapter, trajectory key points simply represented by one-hot vectors with size of key points set. These vectors solely carry information about each key point independently without any hint of their contexts. Certainly, as independent random variables, they are applicable in inference for larger temporal scales where the temporal constraints are relaxed. To model trajectories in lower scales, contexts of a key point becomes influential as well. To model a sequence, it is possible to consider $n-$gram structures as independent inputs, but, this blows up the dimension of the feature space with increase in the number of key points. Even though animal trajectory data sets are dense over the span of individual trajectories, they are relatively sparse in domain of organisms' behavioral states and habitats. For instance, considering only spatiotemporal domain

of the seabirds' movement data was used in this study, the sampled data only covers small and sparse patches of it. This is while, the real-world domain also includes environmental factors and aspects of individuals' behavior. To deal with these challenges, this study offers a new approach for modeling organismal movement data motivated by skip-gram model in natural language processing (NLP) [138]. The proposed objective is mainly to create contextual semantical representation models of organisms' trajectories, specifically, relating to their behavioral trends. With the success in NLP, it is expected that skip-gram model provides a solution for modeling animal movement based on large and sparse data sets. The grounds for this approach are founded on similarities drawn between language domain and movement domain. Here, the basic assumption is that, as sentences are composed of words, trajectories are sequences of segments. Each of these segments is represented by a key point which carries semantical information and was generated given a certain context. More importantly, like words being shared among people of the same tongue, the trajectory key points are also to be shared among animals of the same habitat, or geospatial region.

In summary, in this chapter we aim to achieve the main objectives in two stages. The first stage is to extract proper key points and their features from trajectories which act analogous to words in text documents. The second and the major one is to create contextual representation vectors of these key points in feature embedding space. Hence, these representations of trajectories could be compared or analyzed at multiple levels of feature abstraction, like environmental, spatial and temporal. The method offered in this study could be utilized in various research applications like data exploration, classification or prediction. Here, two applications, a discriminatory and an exploratory one, are demonstrated analyzing data collected from a seabird species, Streaked Shearwaters (*Calonectris leucomelas*), and classifying their gender given their trajectories. Lastly, the main contribu-

tions of this chapter are in both key point extraction and representation. On the extraction side, a multilevel clustering approach presented to capture a sparse map of trajectories in different density levels. On the representation side, to represent the extracted key points with compact, efficient and semantical numerical vectors, we adopted the distributed representation concept and embedded contextual relationships in local Euclidean space using skip-gram model. This work was published in [139].

4.2 Preliminary Concepts

4.2.1 Continuous Bag-of-Words Model

Continuous Bag-of-Words (CBOW) model was introduced by Mikolov [138]. In short, it is a linear feedforward neural network which projects a one-hot vector of a word into an L-dimensional vector space and projects it back to the input space as a vector of probabilities using SoftMax function. In theory, the objective is, for a given centre word, its context words to be situated in the same neighborhood in an L-dimensional vector space. It is optimized by maximizing the sum of log probabilities of centre words given context words as

$$\frac{1}{N}\sum_{n\in N}\log p(w_n|w_c) \tag{4.1}$$

where $p(w_n|w_c)$ is the probability of centre word w_n given the context words w_c and computed as

$$p(w_n|w_c) = \frac{e^{\hat{v}_n.\bar{v}_c}}{\sum_{w\in W} e^{\hat{v}_w.\bar{v}_c}} \tag{4.2}$$

where $\hat{v}_n$ and $\bar{v}_c$ are L-dimensional representation vectors of centre word and context respectively. For each centre word w_n, a one-hot input vector, selects an L-dimensional representation vector v_n. Context consists of both history and future words without any

specific order according to Bag-of-Words (BoW) model. Context vector $\bar{v}_c$ is calculated using context words' representation vectors v_c as

$$\bar{v}_c = \frac{1}{C} \sum_{c \in C} v_c \tag{4.3}$$

4.2.2 Skip-gram Model

In skip-gram model [138], like CBOW model, the goal is to represent each word in dictionary by an L-dimensional vector v_n in an embedding space close to their context words' representation vectors v_c. The training objective for this embedding projection is to maximize likelihood of a word's context given its representation vector. This model is trained for a random number of times in the range of context size C with a word as input and its sampled context words w_c as target labels. Selection of context size and sampling of context words play important roles in quality of the embeddings. This model's objective function is

$$\frac{1}{N} \sum_{n \in N} \sum_{c \in C} \log p(w_c | w_n) \tag{4.4}$$

where N is the number of training words and $p(w_n|w_c)$ is estimated using (4.2) for each context word.

4.2.3 Candidate Sampling

Optimizing the objective function of skip-gram model turns out to be computationally costly with larger sizes of dictionary W. One solution could be incorporating candidate sampling methods. This group of methods attempts to differentiate noise from informative data. Negative sampling (NEG) [140] is one of the candidate sampling techniques which

is used for training skip-gram model. The objective function is defined as

$$\log \sigma(\hat{v}_n.v_c) + \sum_{k \in K} \mathbb{E}_{w_k \sim p_u(w)}[\log \sigma(-\hat{v}_k.v_c)] \tag{4.5}$$

where K is the number of negative samples drawn from a noise distribution $P_u(w)$ for each training sample.

Another candidate sampling method could be employed for differentiating noise in training of skip-gram is Noise Contrastive Estimation (NCE) [141]. Using this method, skip-gram objective function is approximated as

$$\mathbb{E}_{P_i(w_c|w_n)} \log \frac{p(w_c|w_n)}{p(w_c|w_n) + KP_u(w_c|w_n)} + K\mathbb{E}_{P_u(w_k|w_n)} \log \frac{KP_u(w_k|w_n)}{p(w_k|w_n) + KP_u(w_k|w_n)} \tag{4.6}$$

where $P_i(w_c|w_n)$ informative distribution from which, context is sampled. This approach differs from the NEG only in involvement of noise distribution in the loss function.

4.3 Methodology

4.3.1 Problem Formulation

In this section, the intuition behind employing Bag-of-Words and skip-gram models for analysis of organismal movement data, specifically geo-spatial trajectories, is discussed. It is intended to create abstract bridge which links concept of contextual word embeddings to geo-spatial key points embeddings in animal trajectories. For this, the collected data, which is composed of sequences of geo-spatial coordinates recorded from the focal organisms, could be thought of as sequences of sound frequencies. A segment of these coordinates constitutes trajectory segment represented by a key point as frequency segments make up spoken words. In language processing, to account for slight variances in sound, the centroids or the closest samples to the centroids are chosen. Likewise, geo-spatial centroids

used for key point selection in case of trajectories. As a result, trajectories are composed of series of key points as sentences are of words. Theoretically, it is expected that the arrangement of data points in the embedding space would represent the semantical relations between key points.

In previous chapter, the movement models were constructed based on assumption of exchangeability. It was successful to a certain level in modeling of navigation strategies ignoring the temporal order between navigated key points. This approach is applicable in cases where dynamical constraints are relaxed, since conditional independence is assumed between key points of trajectories given navigation capacity, motion capacity, internal state and environment factors. The probability of a set of navigation key points can be written as following.

$$p(u_0, \cdots, u_K|\Omega, \Phi, W, R) = \prod_{k \in K} p(u_k|\Omega, \Phi, W, R) \tag{4.7}$$

With that, it is possible to adjust the extent of the temporal window to which, these key points belong. This could range from an hour to entire a trip. The former depends on dynamical properties according to which, the sequential constraints could be loosened. For instance, effects of wind over entire trip could be ignored as animal compensates for such deviations in their trajectories [13]. In this study, this approach is improved by inclusion of contextual information in modeling of trajectories' key points. This information could be temporal, spatial or from any other semantical domain. It is described as

$$p(u_k|\Omega, \Phi, w_c, r_c, u_c) \sim p(u_k|\Lambda_k) \tag{4.8}$$

where Λ_k is contextual feature vector for u_k. Subscript c identifies the contextual information for the corresponding variables. Theoretically, over the course of a trajectory with

length K, joint probability of key points given their context should be maximized as

$$\arg\max_{\Lambda_k} \prod_{k \in K} p(u_k|\Lambda_k) \tag{4.9}$$

The main motivation driving this approach is based on [11] where contemporary environmental contexts were found to be also influential in navigation strategies of Streaked Shearwaters. For example, for foraging behavior, temporal context which implicates similar environment factors could effectively provide clues about gender of Shearwaters based on set of trajectory key points with their contextual features.

At this point, the remaining part is modeling of $p(u_k|\Lambda_k)$ and Λ_k which is addressed later in this section. Prior to that, the key point extraction method is described.

4.3.2 Key point Extraction

As discussed earlier, one of the challenges in movement ecology is sampling frequency and segmentation of trajectories. Quality of the information, which is carried by representative key points or segments, has substantial role in downstream results. Objective of analysis is also a major factor in selection of the compression or segmentation method. For instance, for analyzing navigation capacity, start and destination of trip segments are generally extracted. In the case of analyzing movement paths, shape of segments is the feature to be preserved. Here, target of analysis is navigation behavior rather than shape of local path segments. Therefore, representation points of trajectory segments are extracted using DBSCAN method. DBSCAN, requires no initial guess for the number of clusters with ability to identify noise. Besides, its relatively higher performance in identifying densities [142] makes it the method of choice here. It is worth noting that, given a fixed sampling rate, destinations of navigation, activity or resting locations, and path intersections could be extracted setting appropriate clustering parameters. These segments represent a trajectory

sequence. Generally, unlike key points for human trajectories like school, cinema, hotel, restaurant, etc. which carry predefined semantical or functional information, key points for animal trajectories, specifically in the case of seabirds are not having fixed locations nor functionally predetermined. An option to approach this problem is to create hierarchical key point clusters. This method identifies spatial super key points that are shared between trajectories and each of the member key points could have a semantical feature assigned to it. For instance, common path ways, foraging spots, or prey patches at sea could be identified using this method.

4.3.3 Model Construction and Optimization

As mentioned earlier, a way to create contextual representation vectors of key points is to maximize (4.9). The probability function $p(u_k|\Lambda_k)$ can be modelled with a SoftMax function as

$$p(u_k|\Lambda_k) = \frac{e^{\Lambda_k . v_k}}{\sum_{k' \in K} e^{\Lambda_k . v_{k'}}} \tag{4.10}$$

where K is set of all key points in habitat. Λ_k is a D-dimensional contextual feature vector and defined as

$$\Lambda_k = \sum_{c \in C} v_c \tag{4.11}$$

where C is size of the context key points set. The loss function can be written as

$$-\sum_{n \in N} \log p(u_n|\Lambda_n) \tag{4.12}$$

where, N is size of training set. This approach is analogous to continuous Bag-of-Words (CBOW) model in NLP.

In [138], it was suggested that CBOW is faster and suitable for larger data sets. It is while, skip-gram produces better representations for smaller data sets. Therefore, the model

proposed here, we took an approach in line with skip-gram model. Here, the objective is set to generate context key points given a key point. It is composed of a simple feedforward neural network with a single hidden layer. The input layer is one-hot vector of a key point and the output layer is a SoftMax function which estimates probabilities of context points given that key point. Like CBOW, this network projects each discrete key point to a point on D-Dimensional continuous vector space. Then, the D-dimensional representation vector is projected back to a K-dimensional continuous vector. Conceptually, the optimized network should project key points to vicinity of their context key points. Algebraically, this network is described as following equations

$$\hat{p}(k_c|h_n) = \frac{e^{\hat{k}_c}}{\sum_{k' \in K} e^{\hat{k}_{k'}}} \tag{4.13}$$

$$\hat{k}_c = \Theta^T h_n, \quad \Theta \in \mathbb{R}^{D \times K} \tag{4.14}$$

$$h_n = k_n^T \Phi, \quad \Phi \in \mathbb{R}^{K \times D} \tag{4.15}$$

in which, $\hat{p}(k_c|k_n)$ is the estimated SoftMax probability of sample context key point $\hat{k}_c$ given the key point k_n in data set and h_n is its representation vector in the embedding space. A schematic diagram of this model is shown in Figure 4.1.

For training this network, NCE function [141] is selected as loss function. For optimization, stochastic gradient descent is used. This combination provides both scalability and reliability to the training process. This setup is optimized as though the output probabilities are significantly higher for the context key points rather than the rest. Regarding the selection of context size C and the dimension of representation space D, they are manually adjusted to achieve the optimum results.

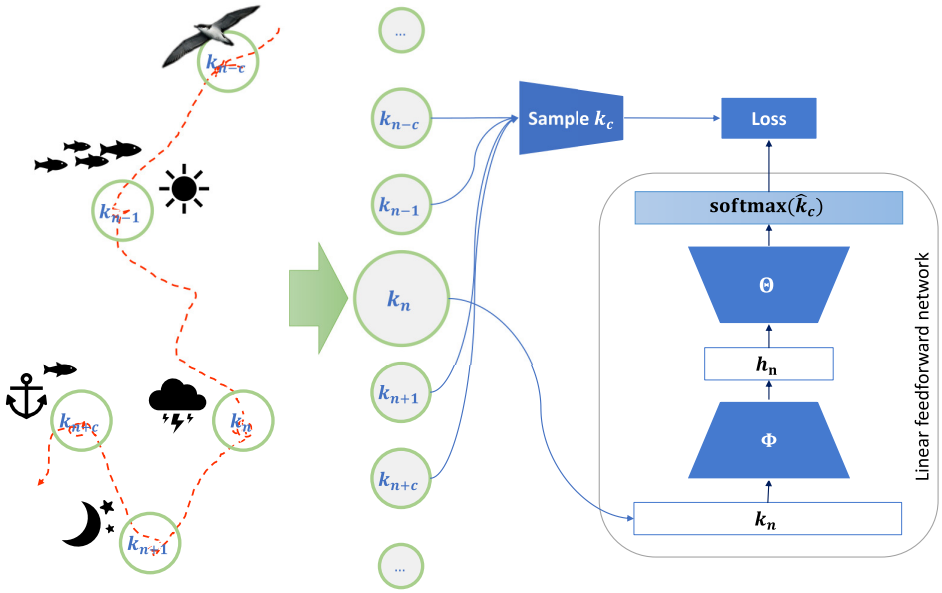


Figure 4.1: Each key point and its semantical features is encoded to a one-hot vector k_n which is then projected on embedding space h_n and projected back to original space. After applying SoftMax, the loss is computed against sampled context key points k_c. This is interpreted as the probability of $\hat{k}_c$ appears in the sampled context key points of k_n.

4.4 Experiments, Results and Discussion

In order to thoroughly address the utility of contextual vector embeddings in animal movement ecology, this section is divided into two subsections. The first one demonstrates the experimental results regarding the applicability of contextual vector embeddings in trajectory data exploration and the effects of various model configurations on the distribution of vector embeddings. In the second subsection, an application in gender classification is experimented. As mentioned previously, the experiment was performed on Streaked Shearwaters off the west coast of Japan. This data set was offered as part of CodaLab's ABC 2018 competition [143]. This data set contains 906 trajectories recorded over the years in fall, 326 of them belong to male birds, 305 to females and 275 assigned for benchmarking.

4.4.1 Trajectory Data Exploration

Trajectories are sequences of spatial data points. The length of these sequences can vary significantly while their spatial footprints only slightly change. Therefore, comparing trajectories with each other was not always a straightforward task to perform. There are methods like dynamic time warping, sampling, etc. available to deal with the existing challenges, but, most of these methods do not create an efficient metric numerical representation for the components of trajectories. Here we show how these metric representations would help researchers to explore and analyze relationships between trajectories.

The first stage involves extracting key points from trajectories. Since, we are interested in navigation trends, beginning and destination of travel segments in trajectories are ideal key points. Assuming that these end points are clusters of points with low or stationary speeds, only points with speeds lower than 2 m/s were considered for clustering. As for the DBSCAN, the neighborhood radius and the minimum number of neighbors set to 1.5 km and 10 respectively. It resulted in 667 clusters. Out of these, 500 were selected as key points based on their ubiquity and frequency in trajectories. Then one-hot encoded vectors of these key points were chosen as input to skip-gram model. Initially, the embedding vector size was set to 128, number of negative samples for the NCE was set to 8 and the context window length was set to 10 in both past and future direction (later in this section, we would discuss role of parameters like embedding vector size in the obtained results). Then the network was trained in batches of 32 until average loss did not change significantly which was at about 10^6 steps.

To visualize the resulted vector embeddings, dimensionality reduction methods PCA [144] and t-SNE [145] were used to create 2d visualizations. Results are shown in Figure 4.2. It is observable that the embedding vectors are pulled closed to each other in some

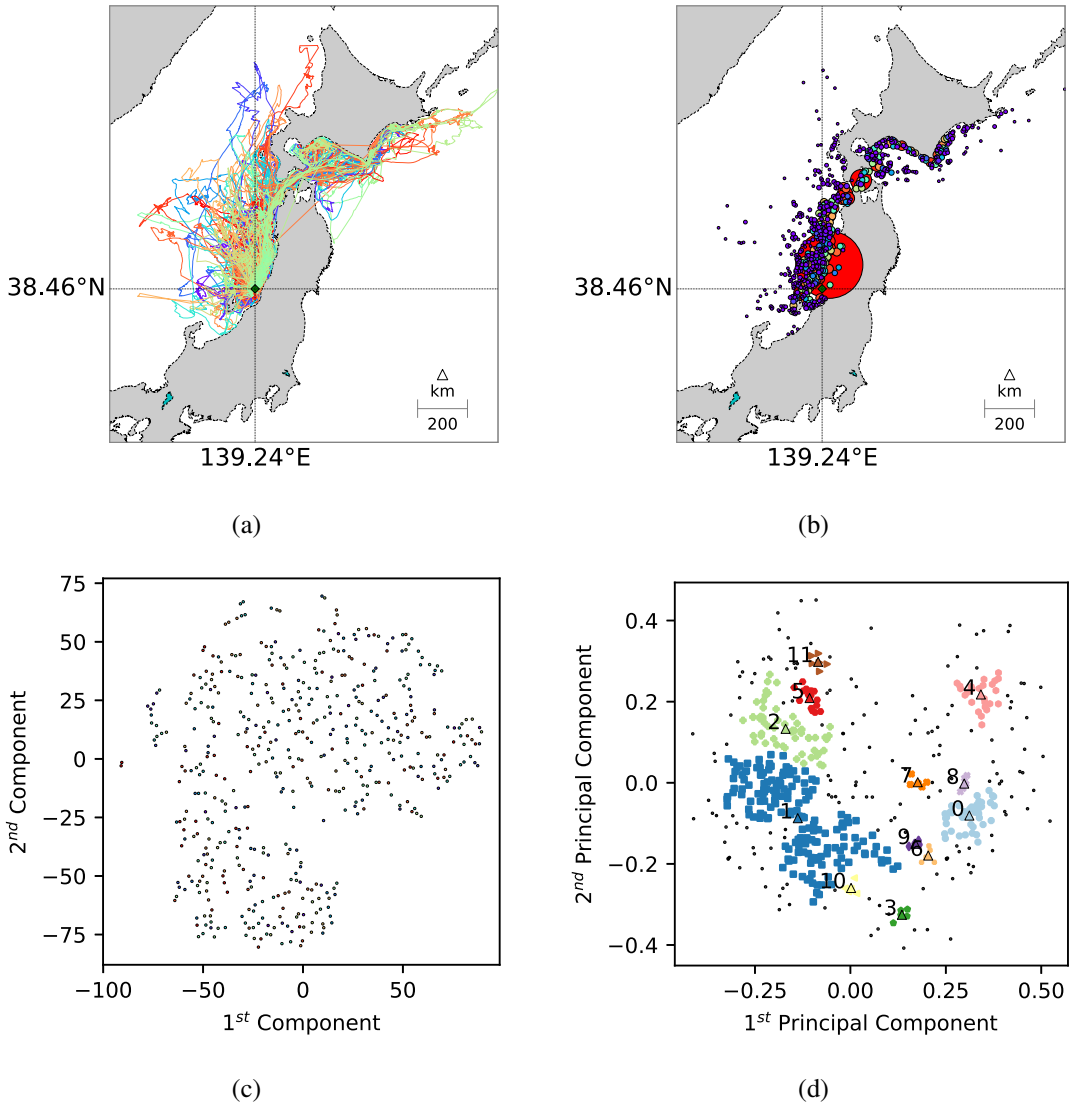


Figure 4.2: Trajectories, selected key points and visualizations of their embedding. (a) Trajectories in geo-spatial space. The colony is designated by ◇. (b) Extracted key points in geo-spatial space. Marker size conveys information about the count of individual trajectories sharing the key point. (c) 2d visualization of key point embeddings using t-SNE. (d) Identified densities of key point embeddings using the first 2 principal components. Densities with minimum of 5 members within the distance of 0.03 are highlighted. Centroids are designated by △.

regions while some points are positioned farther from the rest. To analyze the underlying structure, vectors sampled from local neighborhoods in embedding space and are projected back to geo-spatial space. For instance, the designated key points associated with clusters 11, 5, 10 and 3 in Figure 4.2(d) were projected back to geo-spatial space and plotted along with their corresponding trajectories in Figure 4.3. It is apparent that they are not situated at the same relative distances to each other both intra-cluster and inter-cluster wise. For example, key points in clusters 11 and 5 share major geo-spatial bounding regions while in the embedding space, they have no shared bounding regions. These points are not from the immediate neighborhood in embedding space, but geo-spatially they are close. In fact, even though these points seem to be geo-spatially close, they belong to trajectories with different geometries. It is worth noting that there are distances in the embedding space that differ from of those in the corresponding geo-spatial space. The embedding space in this specific instance is optimized to represent key points' sequential patterns and semantics. Therefore, key points which are traversed consequently may not be in a close neighborhood in geo-spatial space while in the respective embedding space, they appear to be closer to each other. Analogous to text mining, for example, the optimized semantical embedding space pulls "king" closer to "father" rather than "wing", even though king and wing have closer distance in character-based measurements. Furthermore, regarding the embedding space created based on sequential semantics, it is noted that repetitive or cyclic trajectory segments produce concentrated densities in both geo-spatial and the embedding space as shown in Figure 4.4(c) and Figure 4.4(d). This is due to the fact that subsequent key points are located in close neighborhood geo-spatial space.

In Figure 4.4, the corresponding trajectories with key points in clusters 8, 9, 2 and 4 are illustrated. It is also apparent that clusters 8, and 9 have visually closer trajectories than 2,

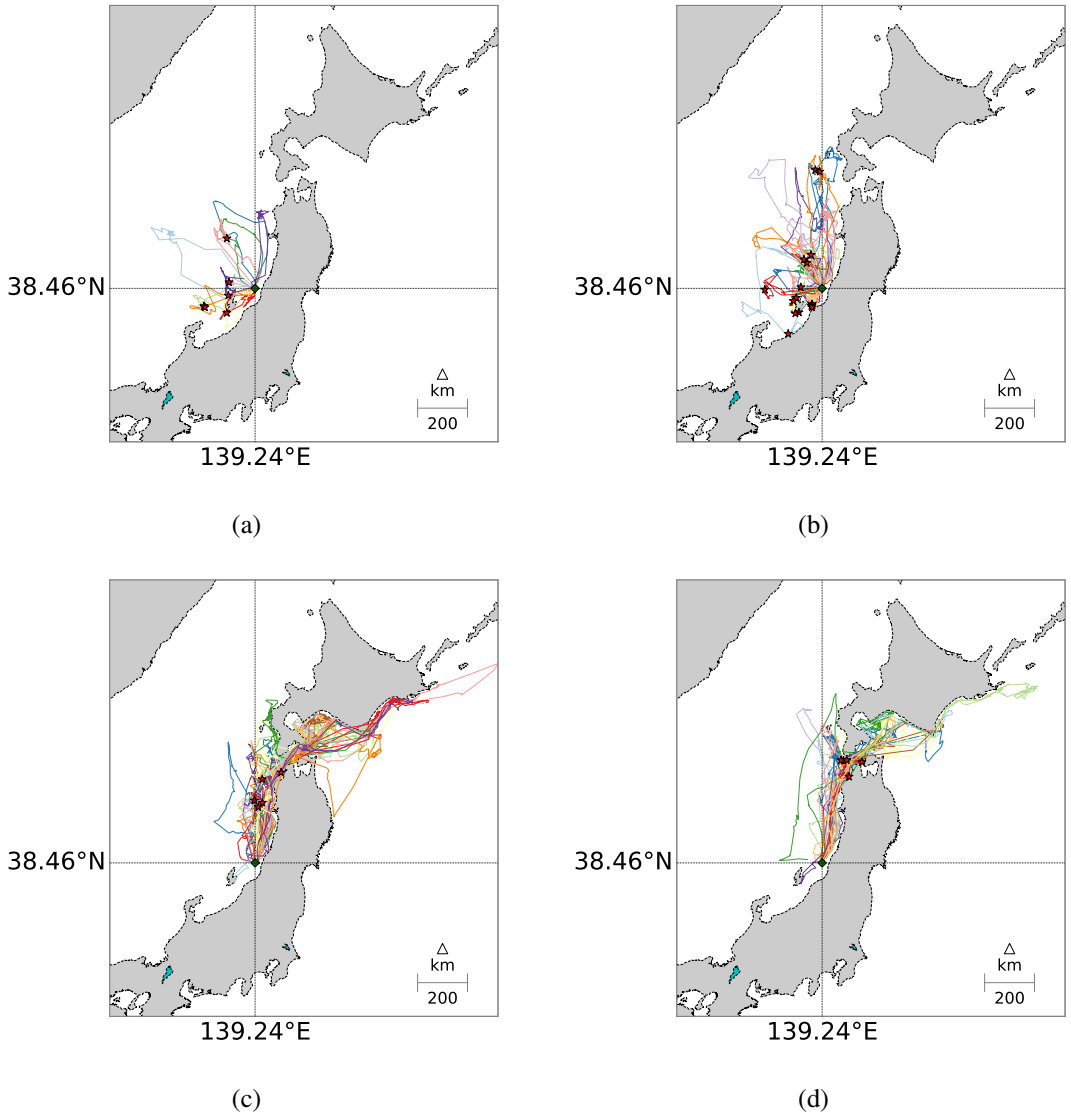


Figure 4.3: Corresponding trajectories of the key point clusters in Figure 4.2(d). key points are identified by ☆. The colony is designated by ◇. (a) Cluster 11 (b) Cluster 5 (c) Cluster 10 (d) Cluster 3

and 4. It is while cluster 8 could be imagined as a transition point between 9 and 4. The same could be applied to 8 between 2 and 4. This underlying structure could be very useful in identifying navigational behaviors between species or discovering relationship between trajectories. For instance, considering clusters 4, 8, 9 and 2 the proportions of connected trajectories being of male gender are about 95%, 89%, 77% and 52% respectively. Though, it should be noted that based on the construction of this data set, there is an unlikely chance of these trajectories belong to an individual bird.

These could very well be advantageous in gender classification of trajectories. As mentioned previously, environment is also an influential factor in generation of navigation strategies. As a result, the clusters of key points in the embedding space could be attributed to a certain weather condition or even habitat features like rivers, coasts, etc. However, due to the absence of calendar information for this data set, it was not possible to test this case for contemporary weather conditions. Up to this point, only sequential semantics in spatial domain were used in construction of embedding vectors. In the next section, utilization of different contextual information in other domains like time and activity is discussed. But, before proceeding to the next experiment, it is worth to discuss tuning of the key model parameters like the representation space's dimension and the context window size, and their effects on the captured information. In regards to the dimension of embedding vectors, certainly, information capacity of embedding vectors is directly related to their dimensionality, or dimension of the representation space. To examine this, the size of hidden layer and context window in the embedding network was modified individually, and their corresponding training results were compared. Part of these results are shown in Figure 4.5. In Figure 4.5(a) and Figure 4.5(b) moving average of the validation error for the last $5e5$ steps and the mean and standard deviation of the last $1e5$ steps in training of different model con-

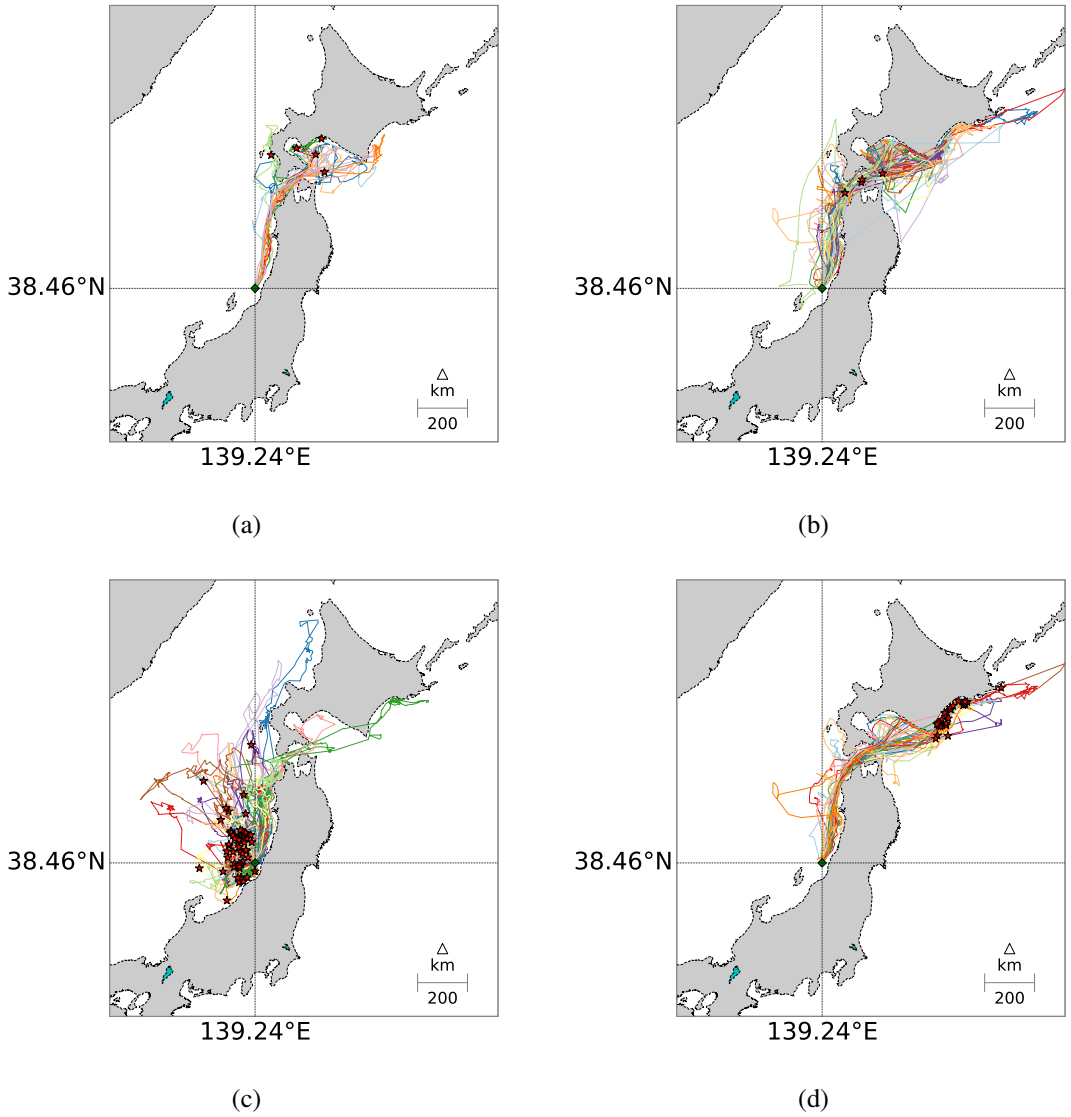
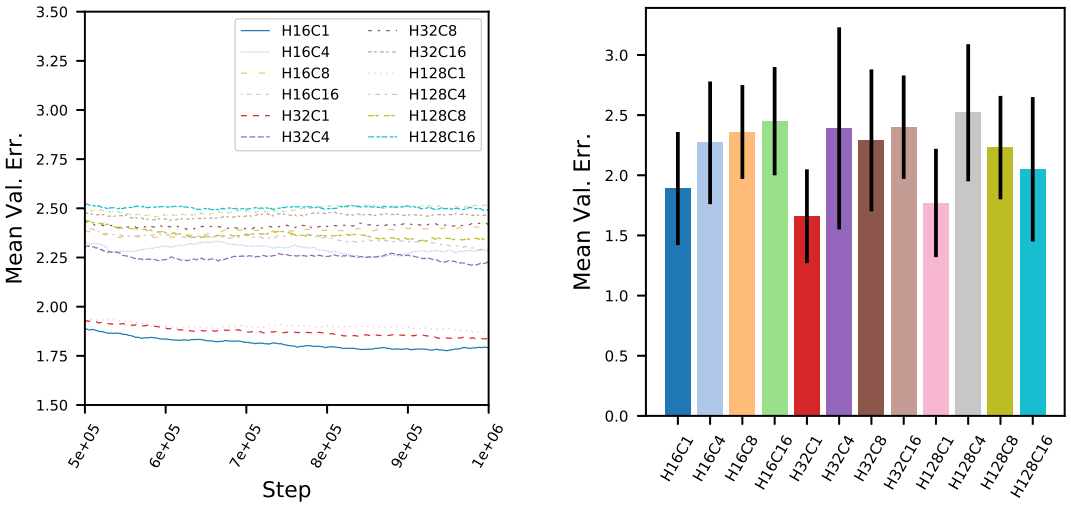


Figure 4.4: Corresponding trajectories of the key point clusters in Figure 4.2(d). key points are identified by ☆. The colony is designated by ◇. (a) Cluster 8 (b) Cluster 9 (c) Cluster 2 (d) Cluster 4



(a) (b)

Figure 4.5: Average validation errors for training different model configurations. The numbers proceeding the letters `H` and `C` represent the dimensionality of the embedding space and the context window size. The number of negative samples and skip samples are set to 4 and 8 respectively and context window spanned bidirectionally. (a) Average of validation errors for different models in last $5e5$ steps of training. It is apparent that $1e6$ steps is sufficient for the training as no further improvement is noticed. (b) Average and standard deviation of the validation error for the last $1e5$ steps of training. This demonstrates that larger context window size requires greater size of embedding vector while it increases the standard deviation.

figurations are shown respectively. The trend of error curves in Figure 4.5(a) shows that there was no significant improvement expected beyond $1e6$ training steps. In Figure 4.5(b), it is seen the that networks with smaller hidden layer dimension have higher NCE loss with larger size of context window, while, posted less standard deviation.

A large context window size attempts to map points farther down in the sequence onto close vicinity of the sampled point. These points may not be geo-spatially close to the sampled point. Therefore, if the context window size is set to 1 the embedding may seem evenly distributed. This is due to the fact that few points in trajectories share the same immediate neighbor key points. On the other hand, if the size of context window is in-

creased, the possibility of sharing contextual key points becomes higher which results in emergence of larger clusters in the embedding space. This effect could be seen using 2d t-SNE embeddings of the trained representation vectors with different context window sizes in Figure 4.6. The context window size is set to 1 for the embeddings shown in Figure 4.6(a) and 8 for the ones shown in Figure 4.6(b). It is apparent that when the window size is small, representation vectors are spread evenly in small concentrations in contrast to the greater context window size with larger pronounced concentrations. In the end, the best choices for the size of hidden layer and context window are dependent on the application, where there are trade-offs to be made.

4.4.2 Gender-based Classification

This experiment aims to illustrate utility of using semantical embedding vectors in classification and prediction. Here, Streaked Shearwaters' trajectories were used to predict their gender. As mentioned previously, recent studies [11] have concluded the existence of gender segregation in trajectories of Streaked Shearwaters. In this experiment, to benchmark advantages of using embedding vectors, we have compared the classification results of a recurrent neural network that was fed in raw spatial coordinates of key points with the results obtained from the same network except that it was used the embedding vectors of key points as input. We also set them side by side with the results achieved using techniques in [37], where LCSS and entropy were employed, and in previous chapter, where trajectories were represented with fixed length vectors constructed based on key points and trajectories frequencies.

To extract key points from different regions with different densities, DBSCAN clustering was performed in 5 levels. In the first level, the minimum distance was set to 1.5 km

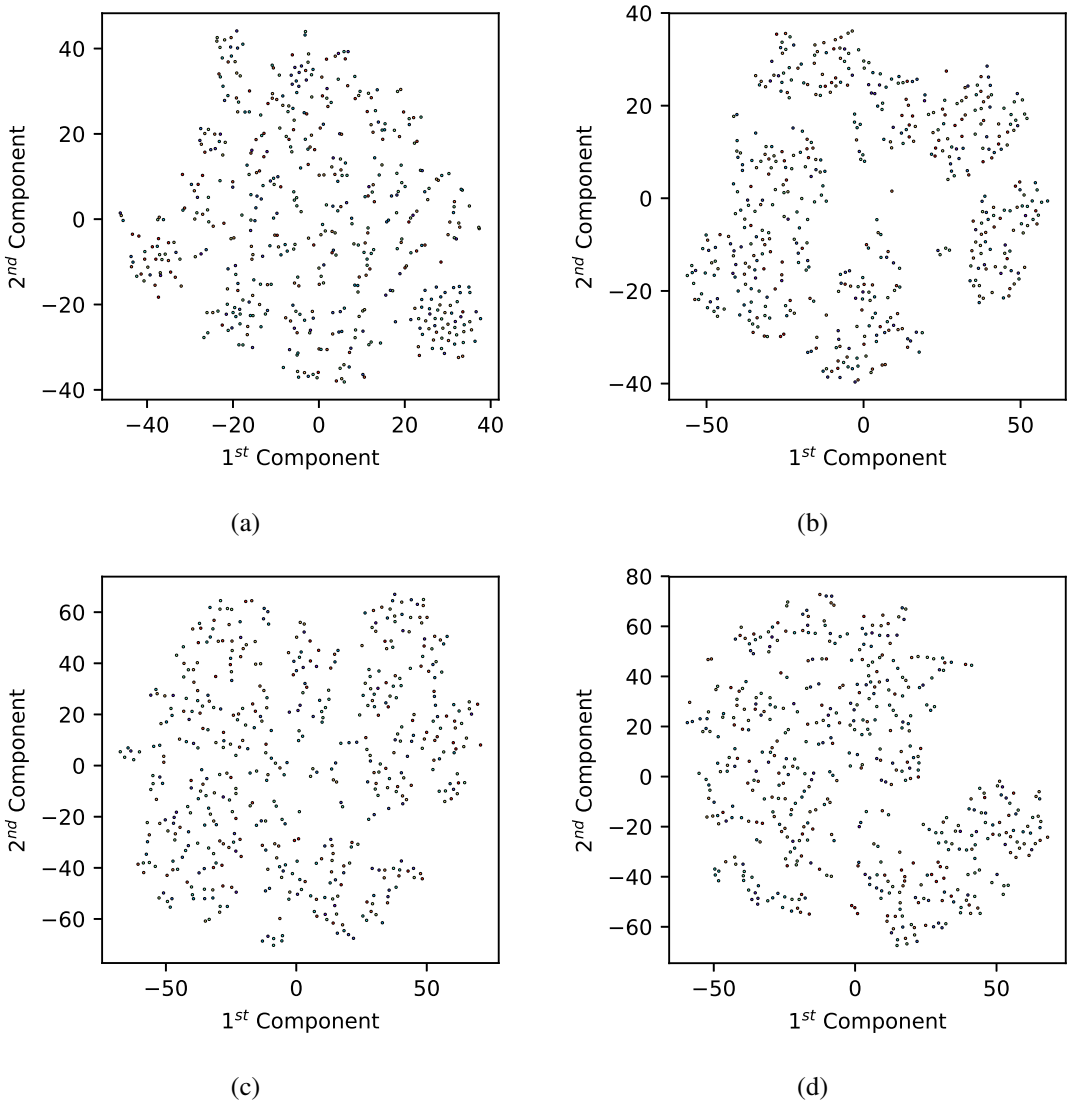


Figure 4.6: 2d t-SNE visualizations of the resulted embedding vectors for different model configurations. The numbers proceeding the letters `H` and `C` represent the dimensionality of the embedding space and the context window size bidirectionally. Model configurations: (a) H16C1 (b) H16C8 (c) H32C4 (d) H32C16

and the minimum number of neighbors was set to 10. At each level, the minimum distance was doubled, and clustering was performed on the noise from the previous level. Then each of these clusters was assigned a unique label. The results are shown in Figure 4.7.

As seen in the result, the lower levels capture topographical features. For instance, in level 1 and level 2 key points, the colony's location and persistent foraging locations are identifiable. It is while level 4 and 5 generally captured transient locations which may have caused by contemporary environmental conditions. Since the method introduced in previous chapter used solely spatial information, here as well, the features were strictly extracted from spatial information for comparison purposes. Based on the distribution of average speed densities shown in Figure 4.8(a), for each trajectory segment containing a key point, mean direction of flight associated with speeds equal or over 2.5 m/s and mean direction of drift associated with speeds less than 2.5 m/s were extracted. These were quantized in 4 directions and a neutral label. Histograms of this feature for female and male birds are shown in Figure 4.8(c) and Figure 4.8(d).

These features designed to encapsulate information about local activities and environment factors in the corresponding key points. For the average sequence length of about 71 extracted from trajectories, the embedding vectors dimension was set to 64. Dictionary size was set to top 500 frequent key points. In the case of recurrent neural network used here, a long-short term memory (LSTM) network [146], with a single layer of 128 cells, was used. Gender classification result achieved from embedding vectors with LSTM (LSTM-EMBD-SP) was compared with the ones obtained from raw spatial coordinates of key points with LSTM (LSTM-SP). These results are listed in Table 4.1.

There are two sets of results that are posted for embedding vectors. One is LSTM-EMBD-SP-1 which has used vector representations created by skip-gram model and the

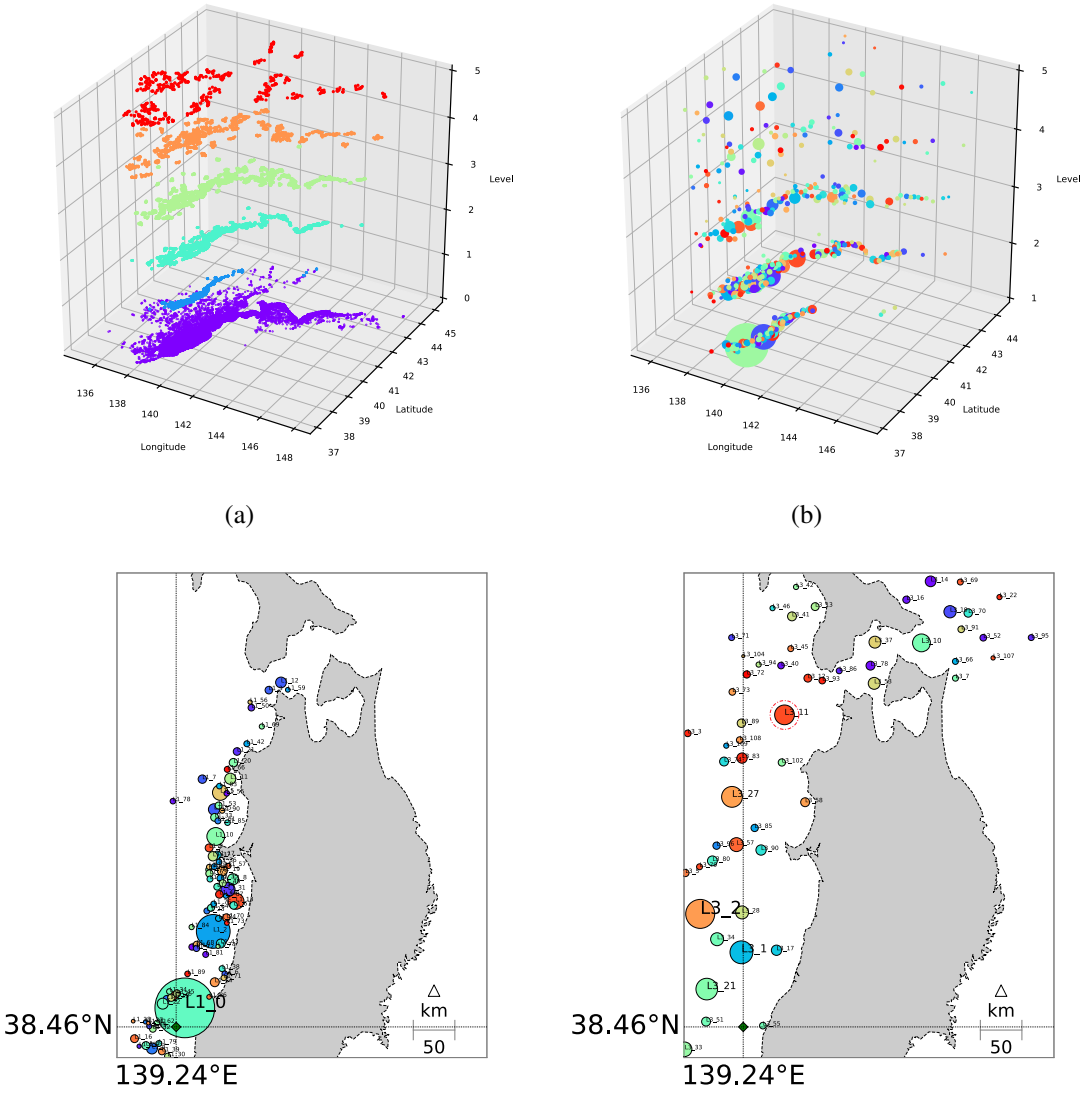


Figure 4.7: Multi-level DBSCAN clustering of Streaked Shearwater trajectory points. Level 0 is trajectory points. Level 1, 2, 3, 4 and 5 are detected clusters with neighborhood radii set to 1.5, 3, 6, 12, 24 km respectively. Minimum neighbors number set to 10. Each centroid's marker size is proportional to the number of trajectories sharing the corresponding key point. (a) Trajectory points assigned to the detected clusters for each level. (b) Centroid points of detected clusters for each level. (c) Sample centroid points of Level 1 clusters. L1_0 is located at colony. (d) Sample centroid points of Level 3 clusters.

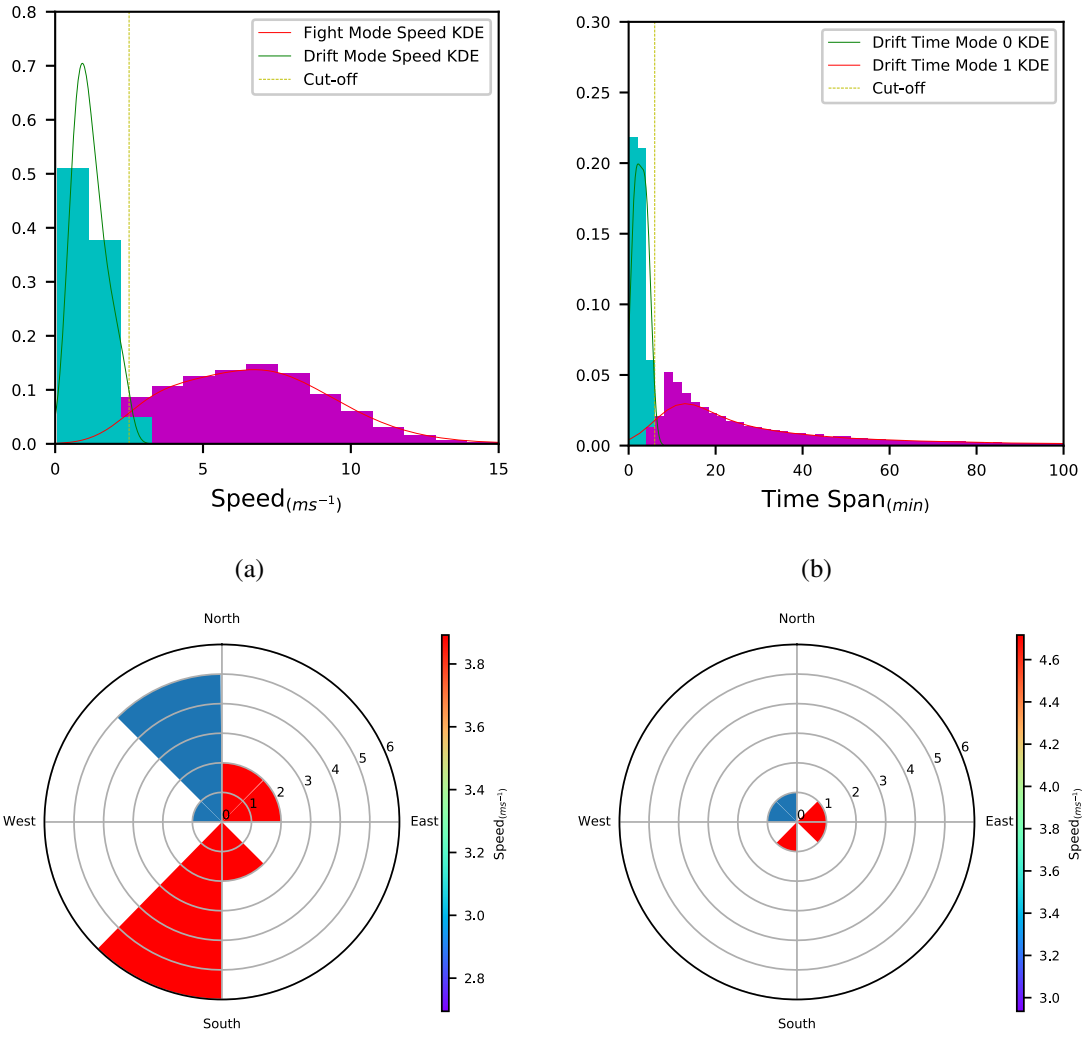


Figure 4.8: Semantical features extracted for key points are based on speed, time and direction. Each one of these features is discrete and semantical. (a) Dominant mode of activity as either flight mode or floating on water designated as drift mode based on 2.5 m/s speed threshold. (b) Time spans for which birds remained at key points. (c) Discretized directions female birds took at "L3_11" key point in Figure 4.7(d).

Table 4.1: For each method, the mean and standard deviation of validation accuracies, and test accuracy is listed.

Method	Acc. Mean (%)	Acc. Std.(%)	Test Acc (%)*
LSTM-SP	57.11	3.88	57.09
LSTM-EMBD-SP-1	**69.85**	3.01	**68.36**
LSTM-EMBD-SP-2	68.31	2.67	-
LSTM-EMBD-SPT	**73.10**	2.22	-
SVM-ENTLCSS[37]	63.03	2.44	61.09

* Test accuracies are obtained via CodaLab's ABC2018 [143] submissions.

other is LSTM-EMBD-SP-2 which has used the ones created by CBOW model. It is seen that there is not a wide performance gap between them. In addition, the performance of classification using entropy and LCSS (SVM-ENTLCSS) [37] was also included. For this method, the highest performing parameters in the classifier were selected. It can be seen that having vector embeddings as input has improved the performance over the other method by about 7% and about 10% over the one using raw spatial coordinates as input. Again, it should be noted that, only spatial information was utilized for creating features. Closely examining variance of the validation results and validation data itself, it shows that, certainly, there are trajectories that are short or spatially uninformative. These are probably gender neutral and their key points are common among both genders. This is analogous to neutral sentiment sentences in sentiment analysis of documents. Furthermore, time, calendar and the features extracted from other domains may also be informative in gender prediction of trajectories. Here, an additional experiment performed by including a new feature constructed based on the continuous time span of a trajectory segment assigned to a key point. In other words, it measures the amount of time that a seabird remains at a key point. Figure 4.8(b) demonstrates the distribution of measured time spans in minutes

for all trajectory key points. Two main densities were identified by a cut-off point set at 5 mins. With this new feature, the network was retrained and tested. Results, labelled as LSTM-EMBD-SPT, are listed in Table 1. It is seen that it could achieve about 4% gain over the results obtained from using only spatial features. This illustrates the potential of features from other domains in further improvement of the classification results. However, this is out of the scope of this work and it is suggested as a follow-up study. The main objective here was to evaluate advantages of using embedding in trajectory classification.

4.5 Conclusions

In this chapter, we have proposed an alternative method for creating vector representations for animal trajectory key points. These representations designed to encapsulate both spatial and semantical information of trajectory segments. This makes them applicable to wide range of problems in trajectory mining such as segmentation, clustering and classification. As part of our experiment, it was illustrated that these representations offered improvements in gender-based classification for Streaked Shearwaters' trajectories. To construct these vectors, we have proposed skip-gram model to be utilized which brings efficiency and scalability over solely using recurrent auto-encoders. With the approach presented in this study, researchers would be able to produce vector embeddings of very large data sets belonging to various organisms. As a result, these embeddings could be counted as numerical representations of behavioral features and they could be used to compare different species' responses to different external and internal factors. In the case of Streaked Shearwaters, it was shown that each gender has differentiable navigation strategies. Another point worth mentioning is the use of negative sampling that offered a very fast and efficient approximation of the loss function, while avoided readjusting all network weights

at each step in a noisy training set. This helps to nudge the vectors far from unlikely neighborhoods at each step.

Like any other method, there are downsides to the proposed approach. Apparently, the very first is in key point extraction. The clustering method's hyper-parameters should be adjusted based on the data. It is not always guaranteed that the same or even similar results achieved on different data sets given keeping the hyper-parameters unchanged. Besides, in case of classification, selection of vocabulary key points is very influential in final results. It is necessary to choose the points which were shared between trajectories. This is analogous to sentiment analysis and language processing. For instance, if there are words in a text that were not seen in any other document, it would be challenging to determine the text's sentiment. The last disadvantage to mention here is that the performance and generalization of this approach as a data driven method heavily relies on the size and quality of the available training data. But, as stated before, with recent advances in data collection and storage, this is the least of concerns.

After all, the final takeaway is considering interpretation of trajectories as sequences of semantical key points which were generated given contemporary internal and external conditions. Since these conditions are shared both temporally and spatially among a set of trajectories, it would be possible to represent these trajectories in a semantical and more informative continuous feature space as animals navigate using semantics rather than numerical coordinates.

Chapter 5

Encoding Trajectory using Recurrent Neural Networks

5.1 Introduction

In previous chapters, sequential constraints in trajectories were generally relaxed. In other words, temporal dimension was not considered as the primary variable in modeling trajectories. In this chapter, we consider time as the primary variable and we attempt to model temporal dynamics of animal trajectories. It is worth mentioning that we could still consider trajectories at multiple temporal scales, but, the trajectory points are considered in temporal order. Availability of efficient and light tracking sensors provided researchers with each tracking data of individual organisms in finer resolutions. Since this data is recorded in form of time series and in variable lengths, it is harder to compare or model them in their original sequential representations. As stated previously, these coordinate sequences may describe actions, behaviors and responses of animals in or to environment. Hidden Markov models were a common tool for modeling trajectory dynamics in move-

ment ecology [52]. However, after recent advances in efficiently training deep recurrent neural networks (RNN) in both software and hardware side, they have become the top tier sequence learning and classification tools. This ranges from very successful speech recognition models [147] to playing video games [148]. They are supervised, semi-supervised and unsupervised learning models [149, 150, 151]. Long short-term memory (LSTM) networks, a sub-variant of RNNs are very successful in house keeping the contents of hidden states in a manner that they can memorize distinctive features in longer sequences [146]. A very powerful feature of RNNs is ability to map the variable length sequences into fixed size vectors. Then these vectors could be used as representation of encoded sequences [152]. Furthermore, these representation vectors can generate outputs based on their held state either conditioned on the last output or independently [151]. Here, the objective is to experiment LSTM networks in various configurations for encoding and modeling trajectory data. The problem is approached in an unsupervised setup. Similar methods were used for video and phrase representations [151, 153] where predictors and autoencoder models learn the state vectors that can produce immediate outputs functioning as a predictor or reproduce the input window functioning as a descriptor respectively. Then, we visualize the local structure and topology of these vectors in latent space by embedding them in lower dimensional mapping space using t-SNE method [145]. This would provide a richer abstract information about the dynamical patterns in the trajectory data. The main contribution of this chapter is unsupervised modeling of animal movement using deep RNNs and it is partially published in [154].

5.2 Preliminary Concepts

5.2.1 Recurrent Neural Networks

For an input sequence of $X_t, t \in \{1, ..., T\}$ where T is time window length, an RNN consists of T hidden vectors H_t and output vectors Y_t as:

$$H_t = \mathcal{F}(W_X^H X_t + W_H^H H_{t-1} + \mathbf{b}^H) \tag{5.1}$$

$$Y_t = (W_H^Y H_t + \mathbf{b}^Y) \tag{5.2}$$

where $\mathcal{F}$ is point-wise non-linearity function. H_t is the hidden state or recurrent vector. Subscripts and superscripts for weights W_{in}^{out} determines the input and output dimension alignments respectively. A multilayer version of RNNs is easily constructed by stacking hidden units on top of each other. The equation for each internal layer could be described as:

$$H_t^l = \mathcal{F}(W_H^H H_t^{l-1} + W_H^H H_{t-1}^l + \mathbf{b}^H) \tag{5.3}$$

where H_t^l identifies the layer l's hidden state vector at time step t. Three sets of weight matrices W_X^H, W_H^H, and W_H^Y and biases $\mathbf{b}^H$ and $\mathbf{b}^Y$ are learned using back propagation. Generally, in RNNs, there are two major issues with simple application of back propagation. Exploding and vanishing gradient in long sequences. Exploding gradients could be avoided using gradient clipping, where gradient values are restricted to a certain limits. Regarding vanishing gradient issue, LSTM network architecture was proposed as a solution to this problem [146]. LSTM networks are built of fundamental blocks called LSTM units. Each unit, consists of a memory cell c_t storing data at time step t which is controlled by non-linearity function gates for reading or writing. These gates are commonly sigmoid functions and namely are input gate i_t, forget or reset gate f_t and output gate o_t and described

as following:

$$\mathbf{f}_t = \sigma(W_X^f X_t + W_H^f H_{t-1} + W_c^f c_{t-1} + \mathbf{b}^f), \tag{5.4}$$

$$\mathbf{i}_t = \sigma(W_X^i X_t + W_H^i H_{t-1} + W_c^i c_{t-1} + \mathbf{b}^i). \tag{5.5}$$

The value of the c_t is determined by point-wise product of previous value of the cell c_{t-1} and output of forget gate f_t summed with point-wise product of input gate i_t and non-linearity *tanh* output of biased weighted sum of previous hidden state H_{t-1} and current input vector X_t as following:

$$\mathbf{c}_t = \mathbf{f}_t \circ \mathbf{c}_{t-1} + \mathbf{i}_t \circ \tanh(W_X^c X_t + W_H^i H_{t-1} + \mathbf{b}^c) \tag{5.6}$$

where, the symbol $\circ$ represents the point-wise or element-wise multiplication. Output of each cell c_t is non-linearity *sigmoid* output of biased weighted sum of H_{t-1}, x_t, and c_t. The updated hidden state H_t is a weighted copy of output where weights are in range $(-1, 1)$ and are controlled by c_t:

$$\mathbf{o}_t = \sigma(W_X^o X_t + W_H^o H_{t-1} + W_c^o c_{t-1} + \mathbf{b}^o) \tag{5.7}$$

$$H_t = \mathbf{o}_t \circ \tanh(\mathbf{c}_t) \tag{5.8}$$

In equations above, the inclusion of weighted c_t values in gates is called `peephole` connection. Similar to RNNs, LSTM units can be stacked in layers to construct deeper networks. A variant of RNNs is bidirectional RNNs, and in case of LSTMs, they are called bidirectional LSTM (BiLSTM) networks [155]. In bidirectional networks, hidden layers are run in both forward and backward directions. Each direction's hidden layer output contributes to the network's output as:

$$Y_t = W_H^Y \overrightarrow{H}_t + W_H^Y \overleftarrow{H}_t + \mathbf{b}^Y \tag{5.9}$$

where $\overrightarrow{H}_t$ and $\overleftarrow{H}_t$ represent the forward and backward hidden layers' outputs at time step t. These networks specially benefit from exploitation of both future and past contexts. As

for multilayer bidirectional LSTM networks, input for each layer's forward and backward LSTM block is written as:

$$X_t^l = W_H^H \overrightarrow{H}_t^{l-1} + W_H^H \overleftarrow{H}_t^{l-1} + \mathbf{b}^H \tag{5.10}$$

A simpler form of RNN units are gated recurrent units (GRU) and were introduced in [153]. The memory cell in GRUs controlled by set $\mathbf{s}$ and reset gates $\mathbf{r}$ defined as:

$$\mathbf{s}_t = \sigma(W_X^s X_t + W_H^i H_{t-1}), \tag{5.11}$$

$$\mathbf{r}_t = \sigma(W_X^r X_t + W_H^i H_{t-1}). \tag{5.12}$$

The memory cell $\mathbf{c}_t$ which is also the output of the unit is described as:

$$\mathbf{c}_t = \tanh(W_X^c X_t + r_t \circ W_H^c H_{t-1}). \tag{5.13}$$

Finally, the updated recurrent state is:

$$H_t = \mathbf{s}_t \circ H_{t-1} + (1 - s_t) \circ \mathbf{c}_t. \tag{5.14}$$

5.2.2 Backpropagation Through Time

Given loss function value $\mathcal{L}_t$ for each time step of an RNN, stochastic gradient descent and backpropagation algorithms are used to optimize the weight $W_\cdot$ parameters of the network. In order to update theses parameters, $\sum_{t \in T} \frac{\partial \mathcal{L}_t}{\partial W_\cdot}$ and $\sum_{t \in T} \frac{\partial \mathcal{L}_t}{\partial \mathbf{b}_\cdot}$ should be computed. Referring to Eq. 5.2, it is seen that $\frac{\partial \mathcal{L}_t}{\partial W_H^Y}$ and $\frac{\partial \mathcal{L}_t}{\partial \mathbf{b}^Y}$ only depend on H_t. Therefore, their gradients are straightforward and derived as:

$$\frac{\partial \mathcal{L}_t}{\partial W_H^Y} = \frac{\partial \mathcal{L}_t}{\partial Y_t} \frac{\partial Y_t}{\partial W_H^Y} \tag{5.15}$$

However, for terms involved with recurrent state vector H_{t-1}, the calculation of gradient become more complex. Since recurrent network's weight parameters are shared over all

time steps, their gradient is calculated by summing over backward procession in time. For instance, for each $\frac{\partial \mathcal{L}_t}{\partial W_H^H}$ is calculated as:

$$\frac{\partial \mathcal{L}_t}{\partial W_H^H} = \sum_{\tau=0}^{t} \frac{\partial \mathcal{L}_t}{\partial Y_t} \frac{\partial Y_t}{\partial H_t} \left(\prod_{\tau'=\tau+1}^{t} \frac{\partial H_{\tau'}}{\partial H_{\tau'-1}} \right) \frac{\partial H_\tau}{\partial W_H^H}. \tag{5.16}$$

Similarly, gradient updates for W_X^H and more complex components in LSTMs and GRUs could be derived. It is evident in Eq. 5.16 that longer time sequences will have adverse effects on the numerical stability of the gradient updates. As mentioned earlier, LSTM or GRU networks introduced as a remedy for diminishing or vanishing gradients due to repetitive multiplication of small numbers. On the other hand we could prevent the explosion of gradients in case of large numbers multiplication by clamping the gradients or limiting the number of steps such multiplications should be done.

5.2.3 Mixture Density Networks

If we suppose to provide a solution for inverse problem of a many-to-one forward problem, our model should be capable of dealing with one-to-many mappings. Most of regression tools provide solutions with the assumption that underlying data has Gaussian or Gaussian like distribution. This might not be the case for modeling physical factors which are identified by the same outcome. In order to model multimodal distributions as such, we could employ mixture density networks (MDN) proposed by Bishop in [156, 157]. These networks introduced to deal with non-Gaussian problems like inverse problems. These networks are able to approximately model an arbitrary distribution using mixture components. A standard Gaussian mixture model could be considered as:

$$P(Y|\mathbf{x}) = \sum_{c \in C} \pi_c(\mathbf{x}) \mathcal{N}(Y|\boldsymbol{\mu}_c(\mathbf{x}), \boldsymbol{\sigma}_c^2(\mathbf{x})) \tag{5.17}$$

The parameters of the distribution components are estimated by a neural network. Since $\pi_(c)$ is a prior probability, it is estimated using $softmax(.)$ function on $|C|$ outputs of the network as:

$$\pi_c(\mathbf{x}) = \frac{e^{y_c^\pi}}{\sum_{c' \in C} e^{y_{c'}^\pi}} \tag{5.18}$$

where y^π is set of $|C|$ outputs of the network assigned to estimation of prior probabilities π_c. Components' means could be directly estimated using the network outputs y^μ. But, in case of variances, network outputs y^σ should be passed through a function which has range of positive real numbers. Therefore $\boldsymbol{\sigma}(\mathbf{x})$ is computed as:

$$\sigma_c(\mathbf{x}) = e^{y_c^\sigma}. \tag{5.19}$$

It should be noted that the components could be multivariate distributions where $\boldsymbol{\mu}_c$ is a multidimensional vector mean and $\boldsymbol{\sigma}_c$ is non-zero elements of Cholesky decomposition of covariance matrix. In order to train MDN networks, log-likelihood of the data labels $\mathbf{y}$ given network parameters is chosen as the objective function for optimization.

$$\mathcal{L} = \sum_{n \in N} \log[\sum_{c \in C} \pi_c(\mathbf{x}_n, W) \mathcal{N}(\mathbf{y}_n | \boldsymbol{\mu}_c(\mathbf{x}_n, W), \boldsymbol{\sigma}_c(\mathbf{x}_n, W))] \tag{5.20}$$

In order to use back propagation for training the network's parameters, first we need to compute the derivatives of output heads for components' parameters.

$$\frac{\partial \mathcal{L}}{\partial y_k^\pi} = \pi_k \gamma_k - \gamma_k + \pi_k \sum_{c \in C/k} \gamma_c = \pi_k - \gamma_k \tag{5.21}$$

where γ is written as:

$$\gamma_k = \frac{\pi_k \mathcal{N}_k}{\sum_{c \in C} \pi_c \mathcal{N}_c}. \tag{5.22}$$

Given the equation for normal distribution as:

$$\mathcal{N}(x|\mu, \sigma) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(x-\mu)^2}{-2\sigma^2}}, \tag{5.23}$$

the derivatives with respect to mean and variance outputs are computed respectively as following:

$$\frac{\partial \mathcal{L}}{\partial y_k^{\mu}} = \gamma_k \frac{\mu_k - y}{\sigma_k^2} \tag{5.24}$$

$$\frac{\partial \mathcal{L}}{\partial y_k^{\sigma}} = \gamma_k \frac{1}{\sigma_k} - \gamma_k \frac{(\mu_k - y)^2}{\sigma_k^3}. \tag{5.25}$$

The equations above could be also computed for multivariate Gaussian distributions. In case of isotropic distribution, all equations remain almost the same other than means being computed for each dimension and in variance equation, L_2-norm is used to measure the distance between data point and each components mean.

5.2.4 RNN Autoencoder

In certain applications, the objective is to replicate or copy the input data. This process is performed through an internal or hidden layer that inscribes the information required for reconstructing or generating the given input. Mathematically, this could be described as:

$$\mathbf{h} = \mathcal{F}(\mathbf{x}) \tag{5.26}$$

$$\hat{\mathbf{x}} = \mathcal{G}(\mathbf{h}) \tag{5.27}$$

where $\mathbf{h}$ is internal state or hidden layer of an autoencoder. $\mathcal{F}$ and $\mathcal{G}$ are called `encoder` and `decoder` functions respectively. In general, autoencoders are designed in a way to store essential information about input in their internal states, which could be imagined as a surface or submanifold that input data resides in input feature space. This in other words is dimensionality reduction or manifold learning. However, in recent years, autoencoders were expanded to probabilistic mappings and learning like variational models which

makes them a state-of-art generative models. Two large groups of autoencoders are regularized and undercomplete autoencoders. In case of undercomplete autoencoders, they act as nonlinear principal component analysis (PCA). They have a mathematical description as described in Eqs. 5.26 and 5.27. Both decoder and encoder functions could be modeled using feedforward neural networks and their parameters could be optimized by minimizing following loss function:

$$\mathcal{L} = \Delta(\mathbf{x}, \hat{\mathbf{x}}) \tag{5.28}$$

where $\Delta(.,.)$ represents a desired dissimilarity measure between original input data and the reconstructed one. The main feature of this type of autoencoders is reduced dimensionality of their internal space or $\mathbf{h}$. With that, networks learn to project the input data from an input space $\mathbb{R}^N$ to a submanifold $\mathbb{R}^D$ where $D < N$ and the conserved information about the data is maximized. In case of regularized autoencoders, the objective is not limiting the capacity of transferred information, rather representation of data with different properties like sparseness, robustness to noise. To construct a regularized autoencoder, a regularization term could be added to the loss function as:

$$\mathcal{L} = \Delta(\mathbf{x}, \hat{\mathbf{x}}) + \Lambda(\mathbf{h}) \tag{5.29}$$

where $\Lambda(\mathbf{h})$ is a candidate regularization term [158, 159, 160, 161, 162]. It is also possible to use data augmentation, where the output of decoder could be compared with augmented inputs as in denoising autoencoders [163, 164, 165].

Variational autoencoders (VAE) are approximate inference approach to construction and modeling of autoencoders [166, 167]. In this approach, encoder and decoder are modeled as conditional distributions $Q(\mathbf{h}|\mathbf{x})$ and $P(\hat{\mathbf{x}}|\mathbf{h})$ respectively. To approximate these distributions, it is possible to define $P(\mathbf{h})$, a probability density function over a D dimensional latent space $\mathcal{H}$ and a neural network model which projects the sampled latent vector to

N dimensional data space $\mathcal{X}$. The network's parameters are optimized to maximize the following marginal probability:

$$P(\mathbf{x}) = \int P(\mathbf{x}|\mathbf{h}; W)P(\mathbf{h})d\mathbf{h} \tag{5.30}$$

It is seen that VAEs could act as generator models as well. We can directly sample from latent distribution and project it to data space.

Like autoencoder models in feedforward neural network models, sequential autoencoders are tasked to reconstruct the input data from latent variables. But in contrast, they receive variable length input sequences with variable lengths. These networks are optimized in a way that the reproduce the input sequence as their outputs. They consist of two main blocks of encoder and decoder as well. The encoder is fed with input sequence and then the last hidden state of encoder network is used as initial state of decoder network. Then it is trained in a way to generate the input sequence in reverse order [151]. This would guarantee that the hidden state contains distinctive features for generating input data sequence.

5.2.5 RNN Predictor

Given a sample sequence of data, an RNN in predictor configuration is able to produce a latent state representation for the stretch of length T and predict the subsequent points. This unsupervised learning setup could be reconfigured as a supervised one by using the subsequent points as target labels for training of the recurrent network. This setup was experimented in predicting subsequent video frames in [151, 168]. Hypothetically, the hidden states that are capable of generating correct predictions encapsulate essential or the most important features or information about the sequence. The prediction window could be designed with variable length while designing the appropriate loss functions are vital to

the success of these models [169]. They could be constructed in two variants. Conditional variant which produces the output based on previously generated output fed back to the network. The other variant receives no information regarding the previously generated output.

5.2.6 Conditional or Unconditional Recurrence

As discussed in [151] there is a design decision on choosing the decoder part of the network models to be conditional or unconditional. Conditional decoder operates by being fed by the previously generated output. There is an advantage to this approach where the network does model multiple mode target sequence distribution. Apparently, unconditional decoder targeting multiple mode targets would results in average of all modes. On the other hand, conditional decoder tends to exploit the immediate correlations between the input sequence. Therefore, it generates outputs based on these similarities rather that generating targets from deep feature information embedded in the hidden vector.

5.3 Methodology

5.3.1 Problem Formulation

In this chapter, we concentrate on modeling path procession process in animal movement. As stated in Chapter 1, this process is influenced by other internal and external processes. Here, we assume all of these processes are represented by a state vector $\Lambda \in \mathbb{R}^D$. The movement process for time step t could be written as:

$$u_{t+1} = \mathcal{F}(\Lambda_t, u_t) \tag{5.31}$$

We are going to model $\mathcal{F}$ with an LSTM autoencoder and optimize it to be able to reproduce the trajectories. But, we employ encoders in various configurations. First, we attempt to model the trajectories utilizing undercomplete autoencoder model. In this part we use standard LSTMs in four different configurations of unconditional and conditional predictors and autoencoders. The Euclidean distance is simply chosen as the loss function. The objective is to explore the capabilities of LSTMs in encode and decoding animal movements.

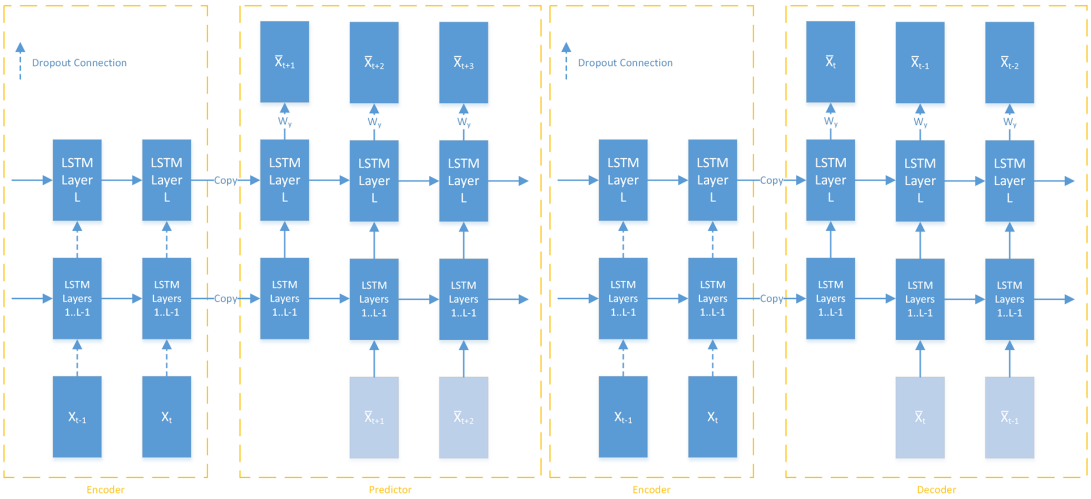
As one suspects. animal movements in environment could be a multimodal process. In other words, it is many-to-one mapping, where different circumstances may lead to a same path. Therefore, instead of simply using Euclidean loss as cost function, we propose generating Gaussian mixture components as output and using likelihood of the sequence $\mathbf{u}$ with length T as:

$$P(\mathbf{u}) = \prod_{t \in T} [\sum_{c \in C} \pi_t^c P(u_{t+1} | \boldsymbol{\mu}_t^c, \boldsymbol{\sigma}_t^c)] \tag{5.32}$$

where C is the set of mixture components. Consequently, to estimate the loss of decoder, negative log-likelihood is computed.

5.3.2 Undercomplete Autoencoder Model

To compress trajectory sequence data in a lower dimensional vector, we use vanilla autoencoder model with Euclidean loss in both descriptor and predictor configurations. Both network models employed here consist of multiple LSTM layers. LSTM cells feature peephole connections with forget bias. As a regularizer, in the encoder component, two dropout layers are placed, one after input vectors and one before the top most layer. In addition, a bidirectional LSTM encoder is also constructed with the same number of layers. The decoder network receives a copy of the encoder network's last state as initial state. Outputs



(a) (b)

Figure 5.1: (a) Multilayer LSTM predictor network diagram. X_t is input data to the encoder part is the estimated input vector at time $t+1$. Predictor inputs are previous step's output. (b) Multilayer LSTM autoencoder network diagram. X_t is input data to the encoder part. $\bar{X}_t$ is decoded input from hidden state at time t.

of decoder block are weighted to produce the immediate data points in the sequence conditioned on the previously generated points. The cost function chosen to be sum of square Euclidean loss with L_2 weight regularizer as:

$$\mathcal{L} = \sum_{t \in T} (\mathbf{u}_t - \hat{\mathbf{u}}_t)^2 + \lambda \|\mathbf{W}\|_2, \tag{5.33}$$

where $\hat{\mathbf{u}}_t$ is the estimated trajectory point at time step t by decoder and $\mathbf{W}$ is network parameters. Both descriptor and predictor networks have similar structures other than the way their decoder components estimates the points. In descriptor model, decoder component produces the points that were fed to encoder. This can be trained to be in forward or reverse order. But, in case of predictor model, decoder component, generates the subsequent points in the trajectory. The length of predicted sequence could be shorter or longer than the input sequence to encoder.

5.3.3 Mixture Density Encoder Model

The internal structure of this model is similar to a standard LSTM network. The only point it differs is its observation model. This network models the inputs as a mixture of Gaussian components [170]. In case of geospatial trajectories, these components are bivariate with 5 parameters and a prior weight. So given the number of components C, the output dimension of our network at each step should be $6 \times C$. The loss function for a sequence is written as:

$$\mathcal{L}(\mathbf{x}) = \sum_{t \in T} -\log\Big[\sum_{c \in C} \pi_t^c \mathcal{N}(\mathbf{x}|\boldsymbol{\mu}_t^c, \boldsymbol{\sigma}_t^c, \rho_t^c)\Big] \tag{5.34}$$

where bivariate Gaussian $\mathcal{N}(.)$ is calculated as:

$$\mathcal{N}(\mathbf{x}|\boldsymbol{\mu}, \boldsymbol{\Sigma}) = \frac{1}{2\pi|\boldsymbol{\Sigma}|} e^{-\frac{1}{2}(\mathbf{x}-\boldsymbol{\mu})^T \boldsymbol{\Sigma}^{-1}(\mathbf{x}-\boldsymbol{\mu})} \tag{5.35}$$

where $\boldsymbol{\Sigma}$ is covariance matrix and $\boldsymbol{\mu}$ is mean vector. They are defined as:

$$\boldsymbol{\mu} = \begin{bmatrix} \mu_x \\ \mu_y \end{bmatrix}, \quad \boldsymbol{\Sigma} = \begin{bmatrix} \sigma_x^2 & \rho\sigma_x\sigma_y \\ \rho\sigma_x\sigma_y & \sigma_y^2 \end{bmatrix} \tag{5.36}$$

To produce the parameter set $\{\mu_x^k \in \mathbb{R}, \mu_y^k \in \mathbb{R}, \sigma_x^k > 0, \sigma_y^k > 0, \rho^k \in (-1, 1), \pi^k \in (0, 1)\}_{k \in C}$ from the outputs of the network, following functions are applied on top:

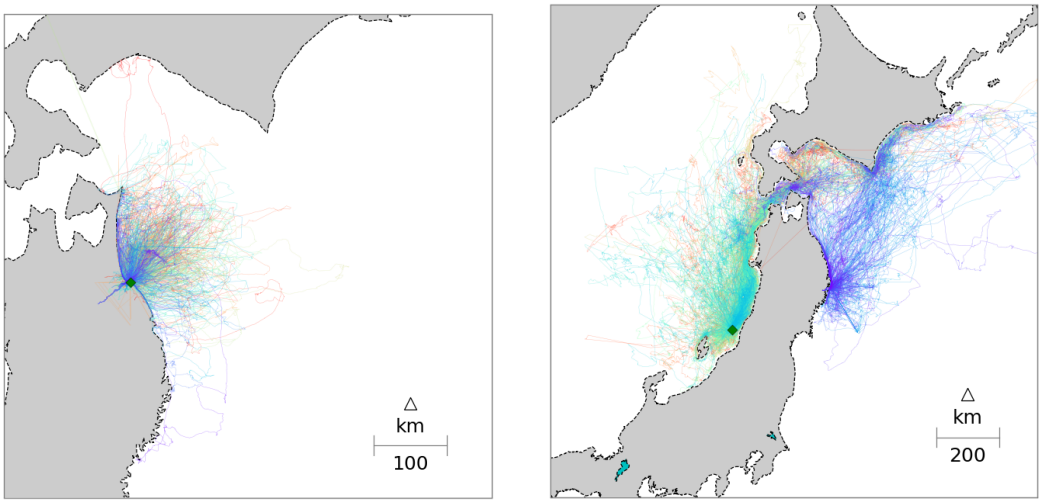
$$\rho^k = \tanh(\hat{y}^k) \tag{5.37}$$

$$\sigma_\cdot^k = e^{\hat{y}^k} \tag{5.38}$$

$$\pi^k = \frac{e^{\hat{y}^k}}{\sum_{c \in C} e^{\hat{y}_c}} \tag{5.39}$$

$$\mu_\cdot^k = \hat{y}^k \tag{5.40}$$

Given equations above, the computation graph of the loss function could be constructed and applying backpropagation, network weight gradients are computed.



(a) (b)

Figure 5.2: (a) Trajectories of seagulls and (b) trajectories of shearwaters.

5.4 Experiments, Results and Discussion

5.4.1 Undercomplete Autoencoder Model

In this experiment, GPS data recorded by loggers attached to Shearwaters and Gulls provided by [135]. Trajectory points are resampled in 1 minute periods and divided into travel segments for training. In addition, trajectory points were mapped to Cartesian coordinates, northings and eastings, using Universal Mercator projection. Figure 5.2 demonstrates the plotted trajectories of both species on the map.

To experiment with the network capacity in encoding trajectories, number of layers set to values 4 and 8. Besides LSTM's cells size was set to 10, 20, 30 up to 100 based on time steps length. To avoid overfitting the dropout probability set to 0.4. Prediction window and number of time steps were varied to observe the network memory capabilities.

Training parameters in this experiment varied to achieve the optimal results. Learning

Table 5.1: Experiment results.

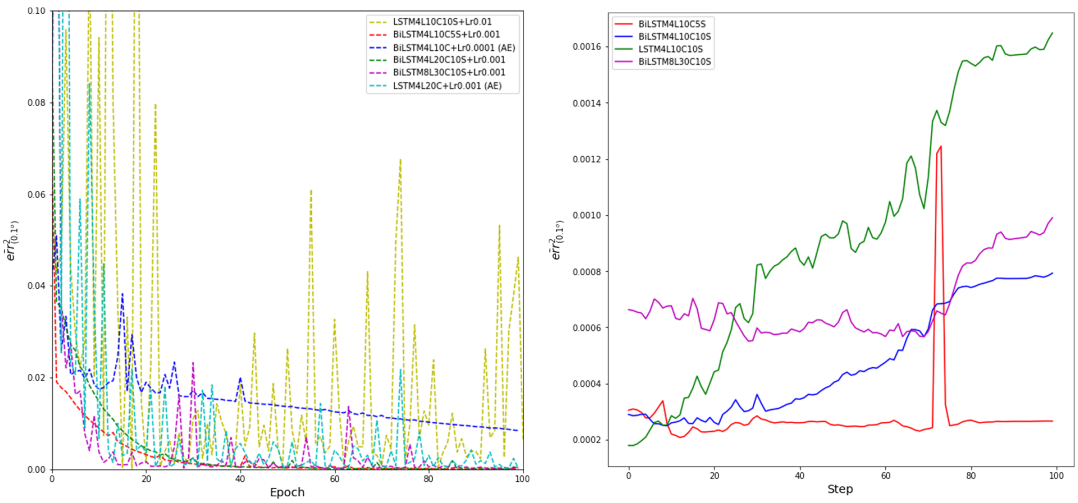
Model	**Train** $\bar{err}_{(0.1^o)}$	**Test** $\bar{err}_{(0.1^o)}$
LSTM4L10C10S+Lr0.01 (P)	0.120870	0.030668
BiLSTM4L10C5S+Lr0.001 (P)	0.011750	0.016677
BiLSTM4L10C+Lr0.0001 (D)	0.019439	0.021809
BiLSTM4L20C10S+Lr0.001 (P)	0.011811	0.069301
BiLSTM8L30C10S+Lr0.001 (P)	0.017418	0.026124
LSTM4L20C+Lr0.001 (D)	0.035561	0.054156

rate was initialized with range from 0.01 to 0.0001 and its decay rate set to 0.5 at each 100 iteration. Batch sizes varied based on the trajectory lengths along with number of iterations. For testing results, depending on the training set size about 10% of it left for testing.

A set of notable results of the experiment are shown in Table.1. Numbers after letter *C* is cell size, *L* is layer size and *Lr* represents the learning rate. Networks with more cell and layers are tend to have smoother training. The bidirectional LSTMs has fast steady decrease rate in loss while standard unidirectional LSTM has less steep descent. Increase in learning rate help the rate of descent while it causes oscillations. Starting with very low learning rates decreases the slope of descent and subsequently slow convergence.

Training trajectories with constraints such as nest number achieve better training accuracy while surprisingly has lower test success. It shows that there overlapping parts of space which are explored by multiple birds in similar ways. If training set restricted to only a bird, the held-out test regions achieve higher losses. Increase in number of cells and layer achieve better results in training while tests show signs of overfitting and shown in Figure 5.3.

Overall performance of autoencoders is lower than predictor as the length of generated

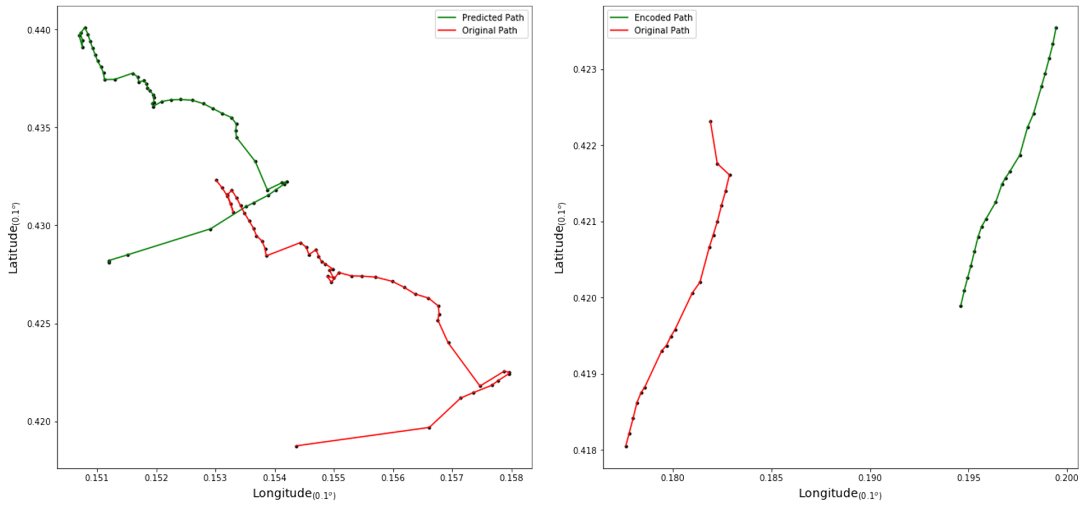


(a) (b)

Figure 5.3: Optimization cost plots. Network parameters are indicated with numbers proceeding letters `L`, `C`, `S`, and `Lr` for number of layers, cell size, prediction steps and learning rate respectively. All sequence lengths are set to 20 time steps. (a) Training cost plots. (b) Test cost plots of prediction networks.

sequences were longer. For instance, an autoencoder with 20 time step units needs to generate 20 data points while in predictor models with time steps predicting 10 or 5 steps ahead had higher performance. In general, the average lowest cost was in neighborhood of 0.15 degrees which is not significantly accurate as in Figure 5.4 . However, considering noisy nature of the measurements and non-deterministic behavior of the birds on top of environmental factors, it is apparent that only spatial context information would not provide highly accurate predictions.

Besides predictions, hidden states of these network would provide information about the traversed trajectory. This can be used to segment and compare the bird trajectories with variable lengths with each other. As shown in Figure 5.5 an embedding of the hidden states of the trajectory should provide abstract level information about hidden states of the birds. Trajectory features such as twists, tangles and spatial contexts would map the states on



(a) (b)

Figure 5.4: (a) Sample of predicted trajectory points from trained hidden states on previous 20 time steps, projection window of 5 time steps, with sampling rate of 1 minute and conditioned on previous output. (b) Sample of generated trajectory points from hidden states encoded with input vectors of 20 time steps at sampling ratio of 1 minutes and conditioned on previous output.

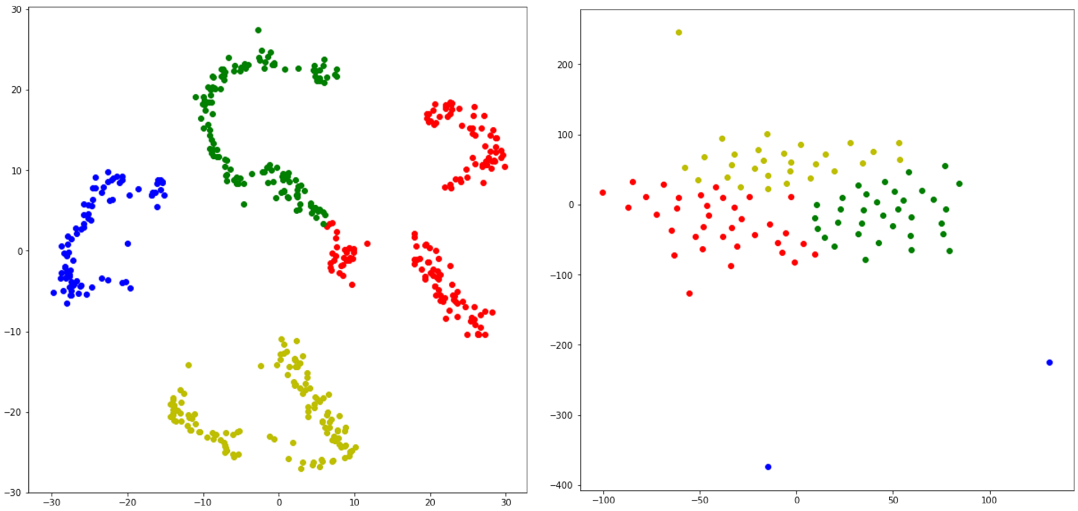
different parts of the embedding space. Moreover, vector embedding demonstrated degree of distinct features learned by the network. For instance, Figure 5.5(a), and Figure 5.5(b) show two different embedding of latent states. States which contain more information are embedded in segments rather than being spread uniformly.

5.4.2 Mixture Density Encoder Model

In this part, we use relative northing and easting to the previous point as input to the network as well.But, here, we segmented trajectories at very long gaps and computed relative northing δn and eastings δe for consequent trajectory points as following [171, 172]:

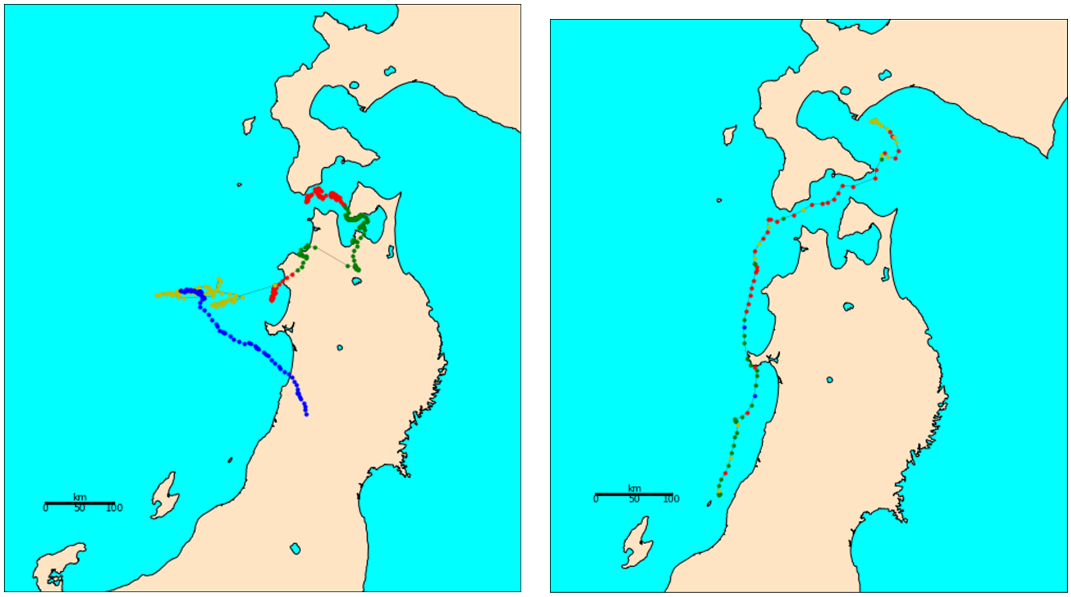
$$\delta e = (\frac{a}{\chi} + h)\cos\phi\delta\lambda - (\frac{a(1-e^2)}{\chi^3} + h)\sin\phi\delta\phi\delta\lambda\cos\phi\delta\lambda\delta h, \tag{5.41}$$

$$\delta n = (\frac{a(1-e^2)}{\chi^3} + h)\delta\phi + \frac{3}{2}a\cos\phi\sin\phi e^2\delta\phi^2 + \delta h\delta\phi, \tag{5.42}$$



(a) (b)

Figure 5.5: Sample embedding maps of trained hidden state vectors using t-SNE. (a) Embeddings of the hidden state vectors form clusters. (b) Embeddings are spread uniformly which mey convey they contain arbitrary information.



(a) (b)

Figure 5.6: Trajectory segments labeled according to the hidden states' embeddings in (a) Figure 5.5(a). (b) Figure 5.5(b).

where

$$\chi^2 = 1 - e^2 \sin^2 \phi,$$

$\delta\phi$, $\delta\lambda$, and δh are changes in latitude, longitude and height respectively. a and e^2 are semi-major axis and first eccentricity squared for `WGS84` datum as:

$$e^2 = 2f - f^2, \tag{5.43}$$

where f is flattening for WGS84.

As for network architectures, we increased the number of cells and reduce the number of layers. Networks with 1, 2 and 3 layers with 64, 128 and 256 LSTM cell size where configured and tested. With increase in cell size, the sequence length is set to 100, 200 and 300 respectively. Finally, there are two options considered for number of mixture components of the output, 10 and 20. With regards to optimization, `RMSProp` [173] and `Adam` [174] were used with gradients clipped at 10. In case of RMSProp, learning rate was set to 0.01, decay to 0.95, momentum to 0.9, and epsilon to 10^{-4}. For Adam, learning rate and decay rate set to the same values as RMSProp, decay rates of the first and second moments set to 0.9, 0.999 respectively, and epsilon set to 10^{-8}. Training and validation result for a selection of network configurations are shown in Figure 5.7 and Figure 5.8 for gulls and shearwaters respectively.

It is observed that with increase in number of layers the network loss drops but this decrease is not significant between networks with single and two layers. It is also seen that with increase in sequence length loss increases. The overfitting u-turn is also clearly visible in validation results of single layer network with sequence length of 300. It is also possible to see differences between outcomes of two species. Shearwaters settle at higher NLL values comparing to seagulls. This is expected as shearwater trajectories are generally of longer ranges. This is also observable in Figure 5.2. Shearwater trajectories cover

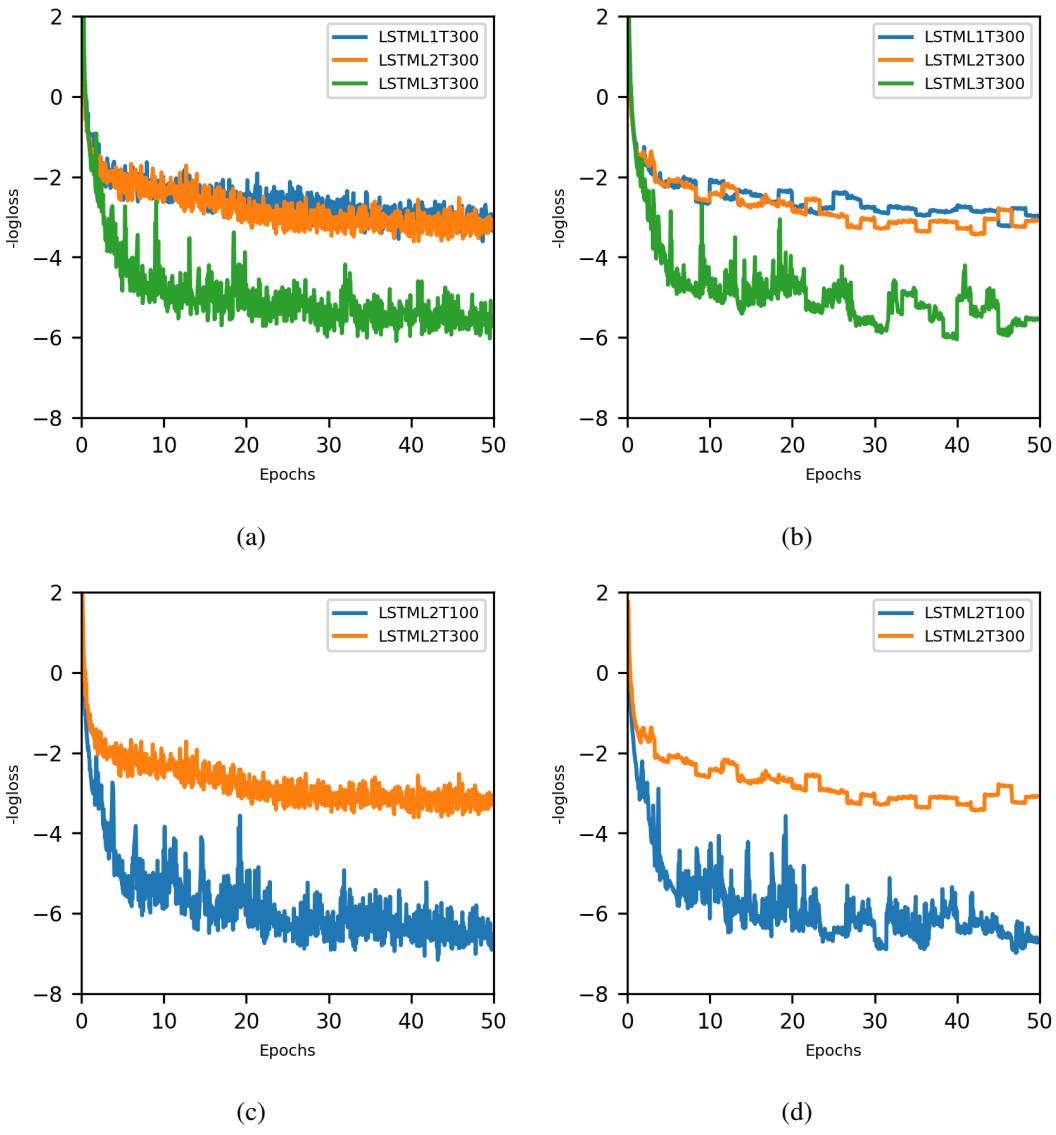
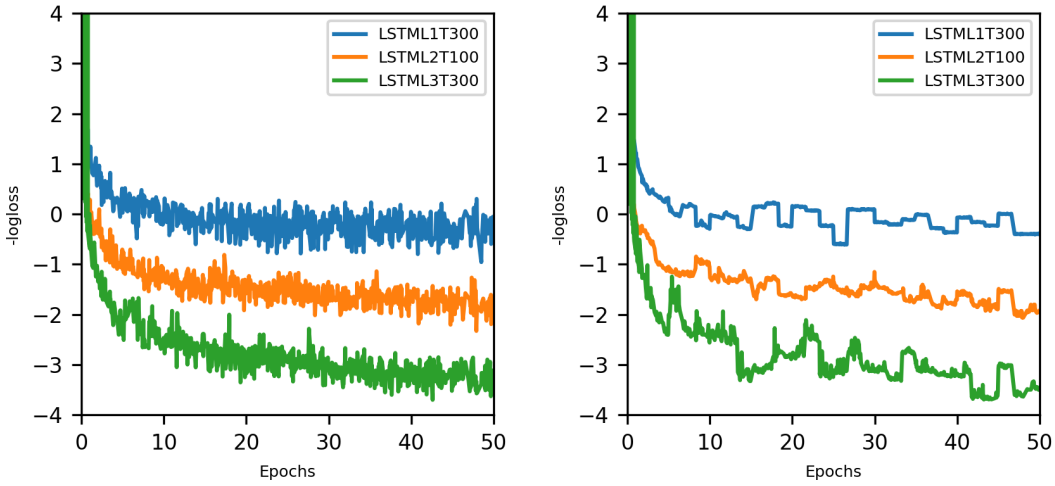


Figure 5.7: Training and validation negative log loss (NLL) plots of the first 50 epochs for seagulls on the left and right respectively. (a), (b) Networks with different number of layers. (c), (d) Networks with different sequence lengths

.

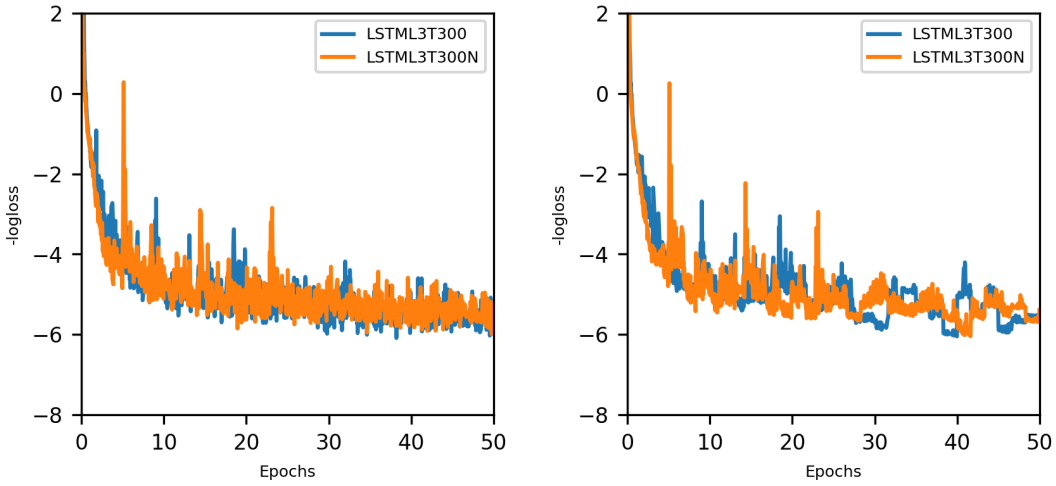


(a) (b)

Figure 5.8: NLL plots of the first 50 epochs for shearwaters with networks with different number of layers. (a) Training results. (b) Validation results.

.

vaster span of geospatial canvas in contrast to seagulls. It is seen that the validation results demonstrate some degree of instability. This, even though not very pronounced in training results, might be originated from the nature of the input data not being normal. A remedy for this issue could be scaling, clipping and normalization of the input data. As a result, experiments performed to compare the results obtained using scaled, clamped and normalized input data. In case of seagulls, northing eastings are normalized to have standard deviation of unity and training and validation results are shown in Figure 5.9. It is seen that the validation result is slightly improved but not considerably. In case of shearwaters, both scaling and normalization of northings and eastings were experimented and results are shown in Figure 5.10. It is seen that normalization alone did not improve the performance significantly, however, scaling with an appropriate number would improve both training and validation results considerably. This should be noted that based on dynamical constraints of the species the scale and clamp factor could be chosen. For instance,



(a) (b)

Figure 5.9: NLL plots for normalized and raw input gulls data. Suffix *N* denotes that the northing and easting steps are normalized in contrast to being only scaled. (a) Training results. (b) Validation results.

a maximum displacement could be considered and divide all northing and easting could be divided by such value and larger entries could be clamped to one. Another choice is to discard the segments with larger displacements. The latter may be preferable in larger data sets.

The other issue to be noted is ensuring flow of gradient in backpropagation. With help PCA [144] and t-SNE [145] it is possible to examine that weights of network layers are trained to being optimized by backpropagation. Instances of the first and second components for PCA and t-SNE projections of the network layers' weights are shown in Figure 5.11 for networks trained on trajectories of shearwaters and gulls.

Here, the gradients are clamped between $[-10, 10]$ in both Adam and RMSProp optimizers. Figure 5.12 shows the training and validation plots for both optimizers. It is seen that both optimizers converge to close values while Adam takes the faster path.

The training could be stopped given the training and validation losses converge to a

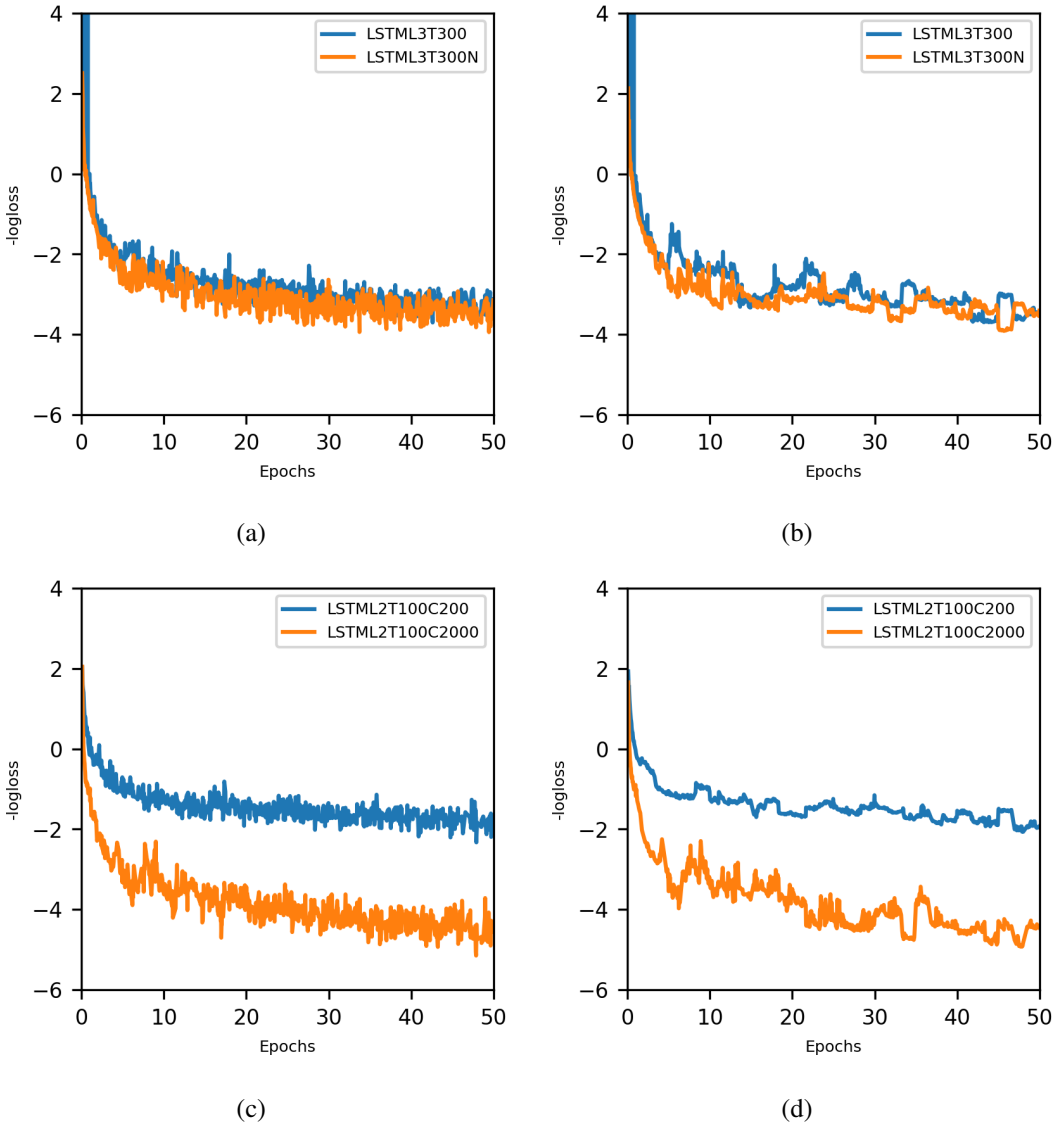


Figure 5.10: NLL plots for normalized and scaled input shearwaters data. Training plots on the right and validation plots are on the left. (a), (b) Normalized inputs versus raw northings and eastings. Suffix *N* denotes that the northing and easting steps are normalized. (c), (d) Scaled and clamped inputs. The number proceeding *C* denotes the value at which the northing eastings are scaled.

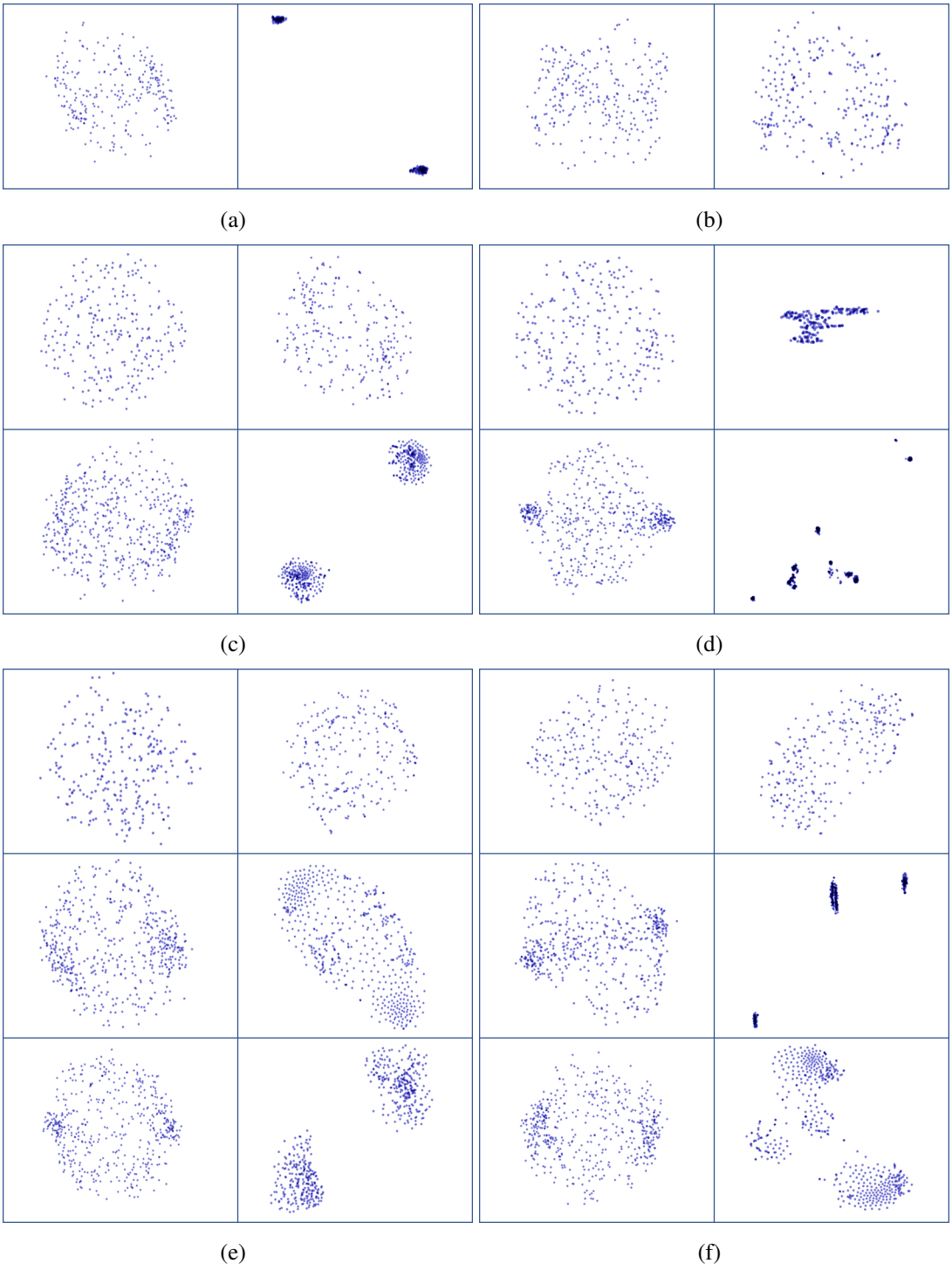
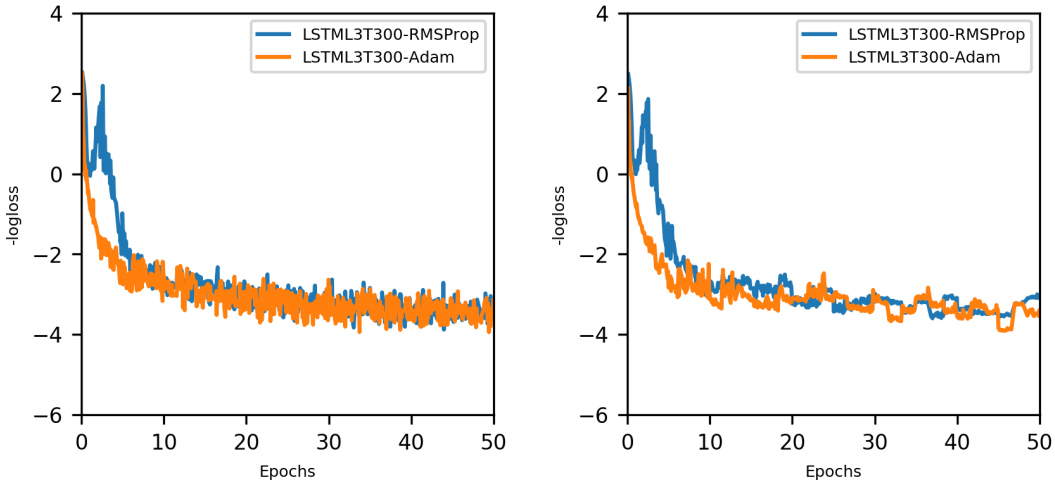


Figure 5.11: 1st and 2nd components of PCA and t-SNE embeddings of network layers' weights for networks with single, two and three layers. Columns on the left are PCA and on the right are t-SNE. (a), (c), (e) Trained on gulls trajectories. (b), (d), (f) Trained on shearwaters trajectories.



(a) (b)

Figure 5.12: NLL plots for Adam and RMSProp optimizers. (a) Training results. (b) Validation results.

lower bound and do not change significantly. This may be variable based on the number of layers in the network and the sequence length in addition to the data set. Models trained on shearwaters trajectories converged later. After stopping the training, we are able to sample trajectories from the networks by feeding zero states and zero coordinates for the first input. Figure 5.13 and Figure 5.14 show few unbiased sampled trajectories with lengths of 200 for both gulls and shearwaters. It is apparent that the sampled trajectories have distinguishable features particular to each species. For instance, gulls trajectories contains more stops and shorter ranges in contrast to the sampled trajectories from shearwater model. Later on to confirm these particularities, we draw sample populations from each model and compare their statistics.

To compare the generated results with the trajectories, a population of 100 trajectory sets of 2000 sampled points drawn from gull and shearwater models. We compare the main kinematic properties of trajectories, mean and standard deviation of speeds, and the

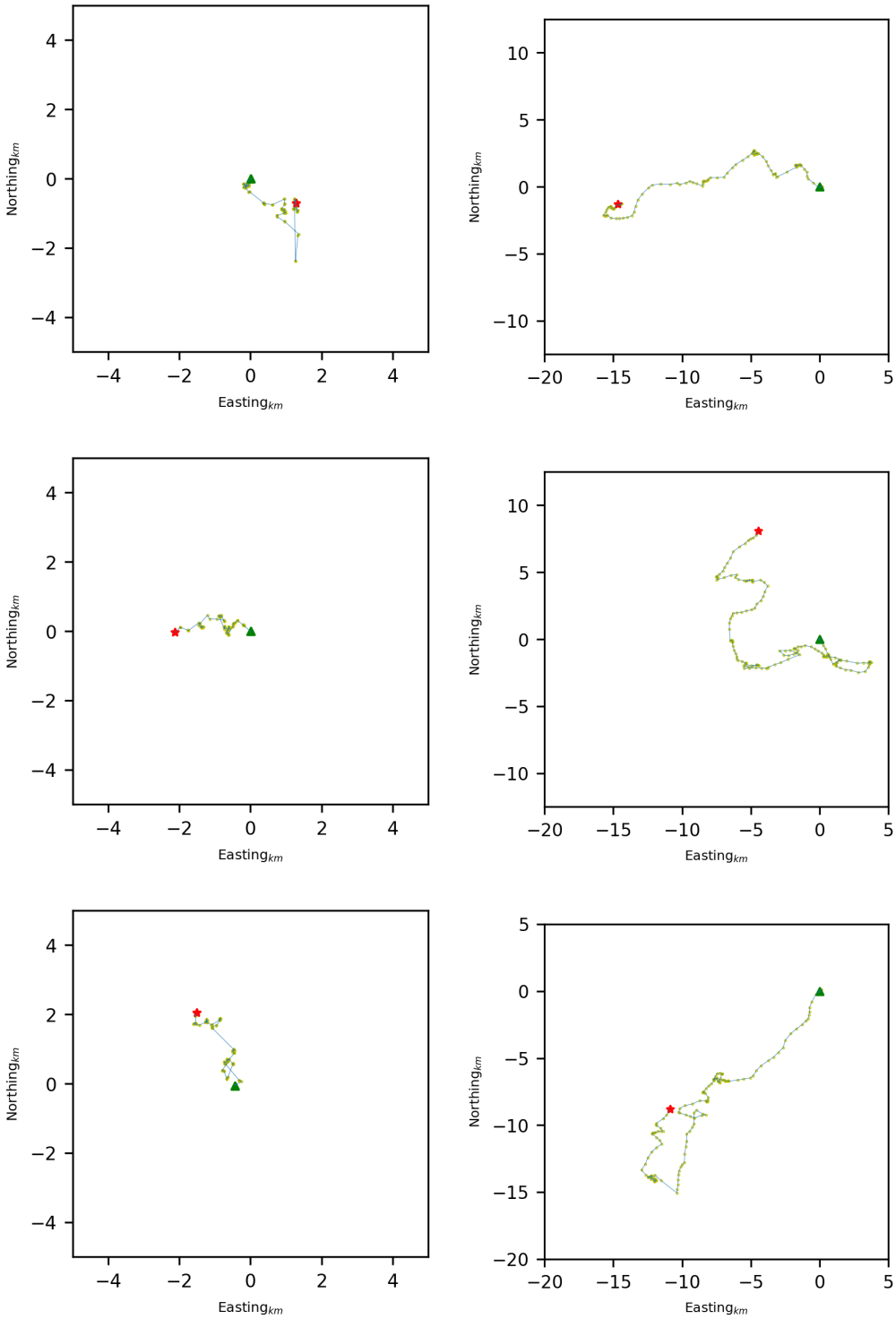


Figure 5.13: Unbiased samples of trajectories drawn from gull and shearwater models shown on the left and right columns respectively. Green △ denotes the start and red ☆ determines the end of the segment.

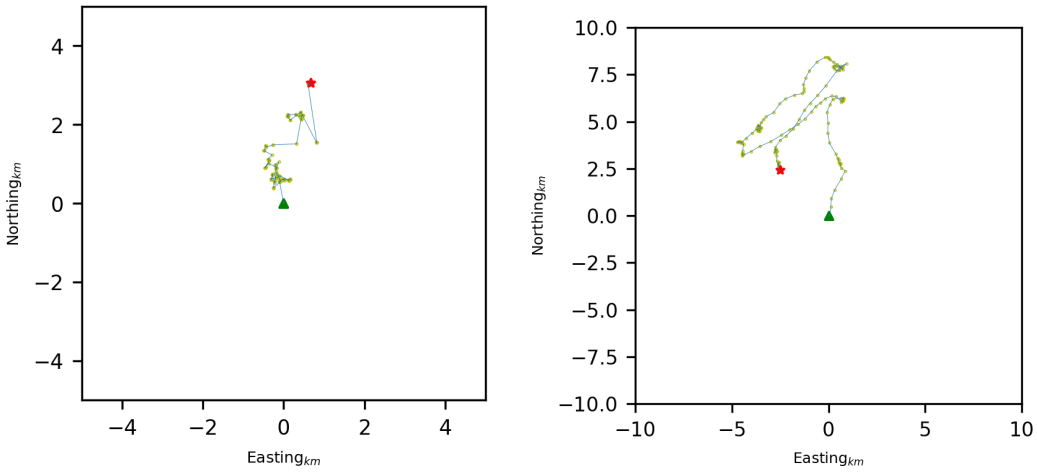


Figure 5.14: Unbiased samples of trajectories drawn from gull and shearwater models shown on the left and right columns respectively. Green △ denotes the start and red ☆ determines the end of the segment.

fraction of points that are considered stationary. This is determined by speed threshold of $2.5m/s$, where segments with speeds below the threshold considered stationary. Figure 5.16 and Figure 5.17 compare the kinematic statistics of the generated trajectories and the data. It is seen that in case of seagulls, the speed means are centered about $22km/h$ ($21.9km/h$) while mean of the data set is about $32km/h$ ($31.6km/h$). It is also should be considered that the data is not distributed normally. It also clearly observed that there are two main components in distribution of stationary points and standard deviation of speeds. The generated samples fall in the major components neighborhood. Regarding shearwaters, generated and data components are closer while standard deviation of speeds are centered much higher in generated samples. It is also possible to examine the performance of the trained networks by reconstruction of sections of trajectories. The length of section was selected to be 60 ($1hr$) and we draw population of 100 segments from test data. In this manner, we can measure root mean squared errors (RMSE) as listed in Table 5.2, Figure 5.18 and Figure 5.19 for gulls and shearwaters respectively. It is seen that errors are smaller

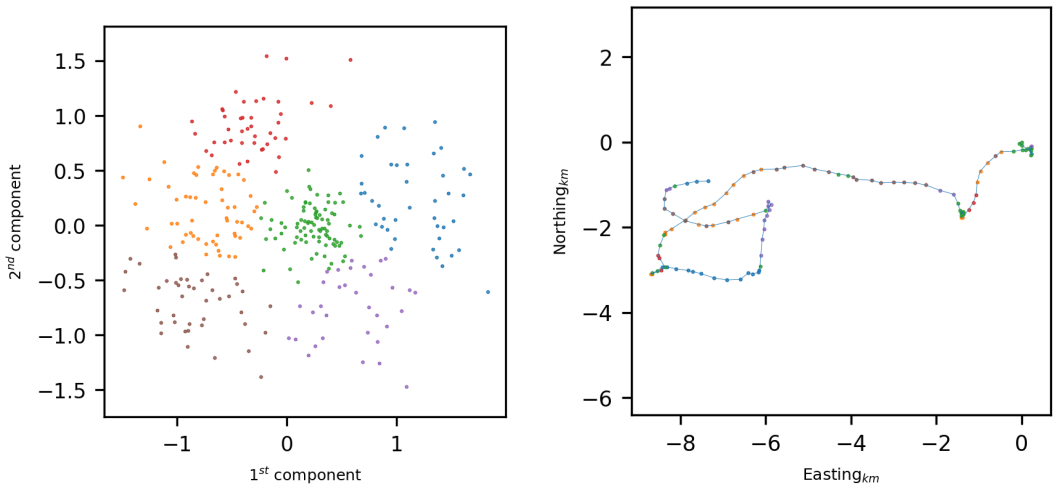
Table 5.2: Expected RMSE statistical measures of 100 reconstructed segments with length 60 ($1hr$) sampled trajectories from gulls and shearwaters test sets.

Measure	Sample Mean (km)	Sample Std.(km)
Euclidean Distance (Gulls)	0.24	1.40
Eastings (Gulls)	0.51	2.67
Northings (Gulls)	0.53	2.78
Euclidean Distance (Shearwaters)	1.33	5.91
Eastings (Shearwaters)	2.32	13.18
Northings (Shearwaters)	2.18	12.26

in case of gulls as expected, however, there is close standard deviation. This might be due to the sea trajectories of gulls which are inherently hard to predict without being provided with information about dynamics of environment such as wind speeds. Moreover, with increase in the length of sections to be reconstructed the errors increase as well. It should be noted that test sequences should be lengthier than 500 points to allow clamped input of 300 points prior to the reconstruction.

It is also to generate trajectories based on specific information available in prior about trajectories. This is also referred to as `primed` sampling in [170]. Instead of simply training trajectories on only a group of trajectories, a segment of a trajectory is fed to the network and the rest is generated. In addition, this could be preconditioned on the geospatial region of the trajectory. This is done by slightly modifying the network and providing input information about the zone to the hidden layers of the network as in [170, 175]. To demonstrate the first layer's information with regards to the trajectory points, its first and second PCA components plotted in 5.15.

Using trainable weights, it is possible to control the outputs of the network given its belief as interpreted in [170]. Likewise, here a weight window was designed with size of



(a) (b)

Figure 5.15: (a) The 1st and 2nd principal components of first layer states of the network plotted and its KMean clusters. (b) The corresponding generated trajectory and associated points to each cluster.

spatial grid of 24×28 half degree squares. Every grid cell assigned a one-hot vector as well. The discrete convolution of the window weight $\boldsymbol{\psi}$ and grid vectors $\mathbf{q}$ were supplied to the network as following:

$$H_t^1 = \mathcal{F}(W_{X_t}^H X_t + W_H^H H_{t-1}^1 + W_{\Lambda_{t-1}}^H \Lambda_{t-1} + b^H), \tag{5.44}$$

$$\Lambda_t = \sum_{s \in S} \boldsymbol{\psi}(t, s) \mathbf{q}_s, \tag{5.45}$$

$$\boldsymbol{\psi}(t, s) = \sum_{g \in G} \alpha_t^g e^{-\beta_t^g (\kappa_t^g - s)^2}, \tag{5.46}$$

where S is the set of trajectory segments grid cells, s is current segment at time t, and G is the number of Gaussian functions. α_t^g, β_t^g and κ_t^g are optimizable parameters obtained from the first layer of the network, as in [170], as following:

$$h_t = W_{H^1}^h H_t^1 + b^h, \tag{5.47}$$

$$\alpha_t = e^{h_{\alpha_t}}, \tag{5.48}$$

$$\beta_t = e^{h_{\beta_t}}, \tag{5.49}$$

$$\kappa_t = \kappa_{t-1} + e^{h_{\kappa_t}}, \tag{5.50}$$

where h_t is the tapped output from first layer, α_t is interpreted as importance factor, β_t as spread and κ_t as the offset of the window from start of the segment. These parameters are also optimized using backpropagation from the network gradients as well. Details are elaborated in [170]. Figure 5.20 illustrates few examples of the generated shearwater trajectory samples preconditioned on grid locations at Awashima colony and the Tsugaru strait cells. The latter primed sampled on the male gender. It is seen that they demonstrate different navigation strategies given the geospatial information. In addition, as seen in previous chapters, at the strait, it was more likely that the male shearwaters took the path through the strait.

5.5 Conclusions

In this chapter we demonstrated an attempt to explore potential capabilities of RNNs and in particular LSTM networks for learning trajectory features of marine birds and predicting track points. As the main objective, it was shown that these networks could learn a contextual feature vector representation of the trajectory segments. Given these vectors it would be possible to compare different trajectory features and in outlook, they present encoded behavior of the focal species.

The first notable point regarding prediction using undercomplete encoders was, if conditional decoding on previous outputs performed on very lengthy ranges, there would be drift introduced in the predicted trajectories. It strictly limits the prediction window length and to avoid it, the input data should be updated with original data points. This is expected

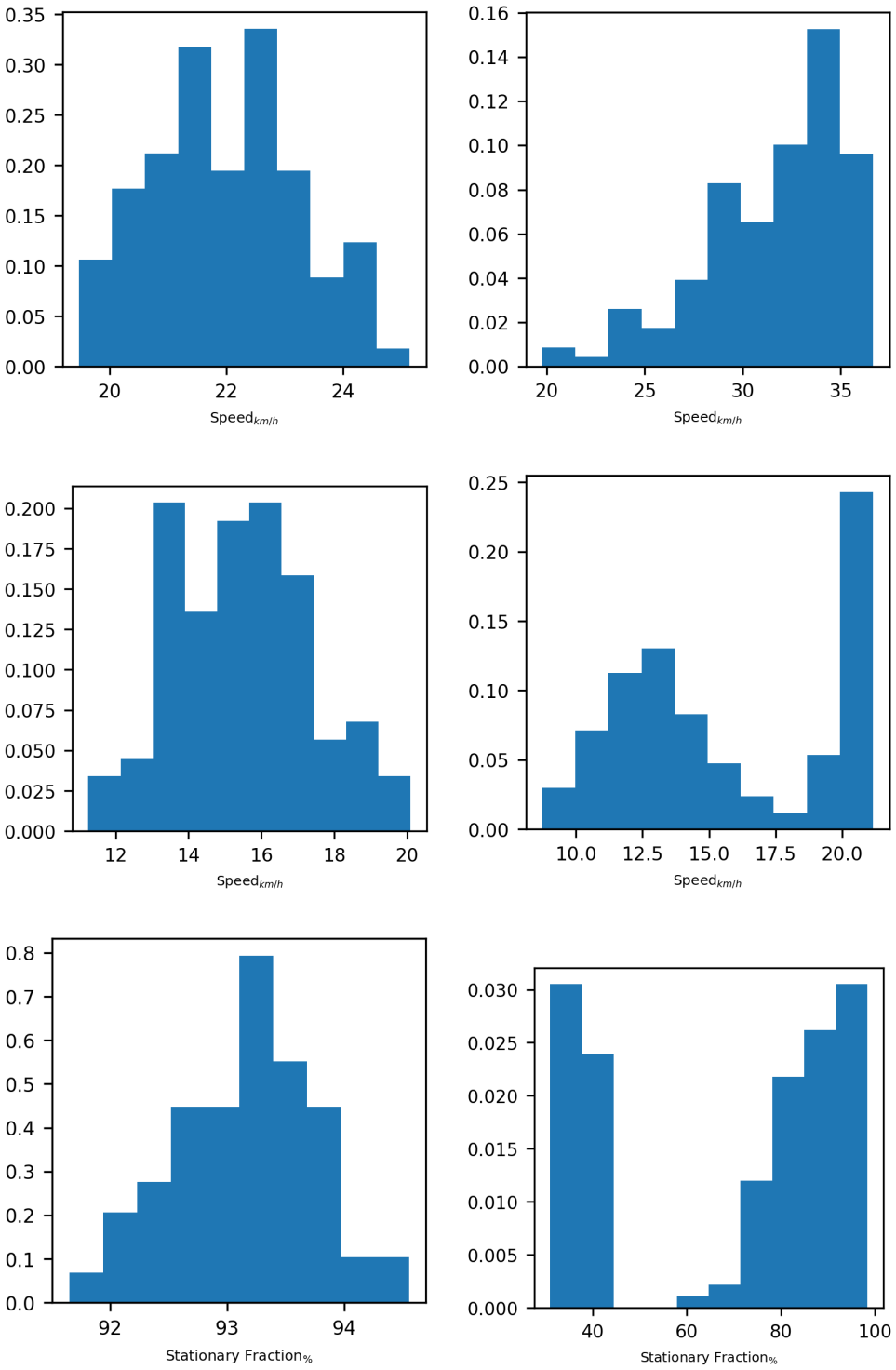


Figure 5.16: Mean (top) and standard deviation (middle) of speeds, and fraction of stationary points (bottom) in generated trajectories and trajectory data set of seagulls on the left and right columns respectively.

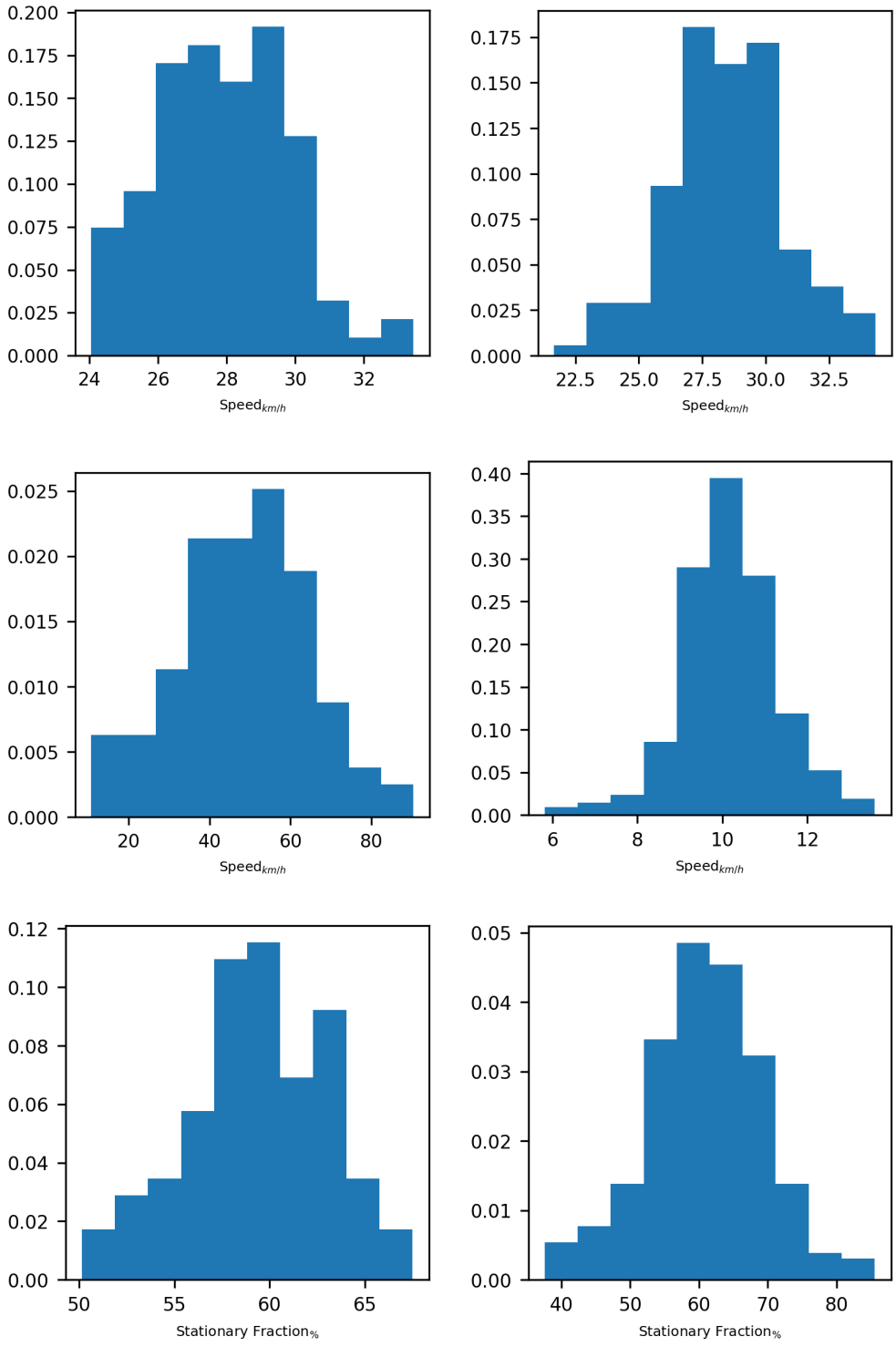


Figure 5.17: Mean (top) and standard deviation (middle) of speeds, and fraction of stationary points (bottom) in generated trajectories and trajectory data set of shearwaters on the left and right columns respectively.

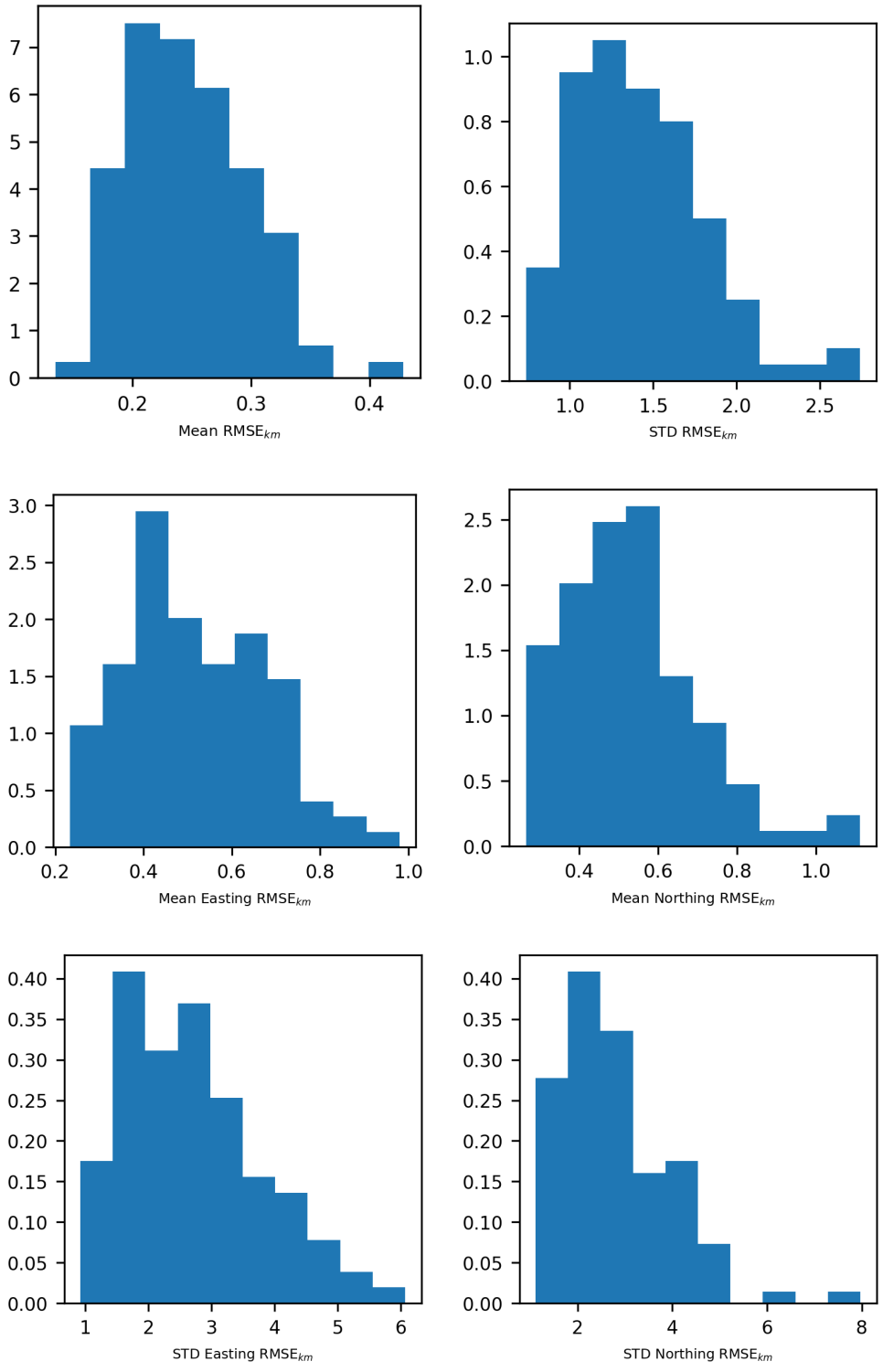


Figure 5.18: Distribution of trajectory segment reconstruction mean and std of RMSE measured in Euclidean distance, northing and easting dimensions in gulls trajectories.

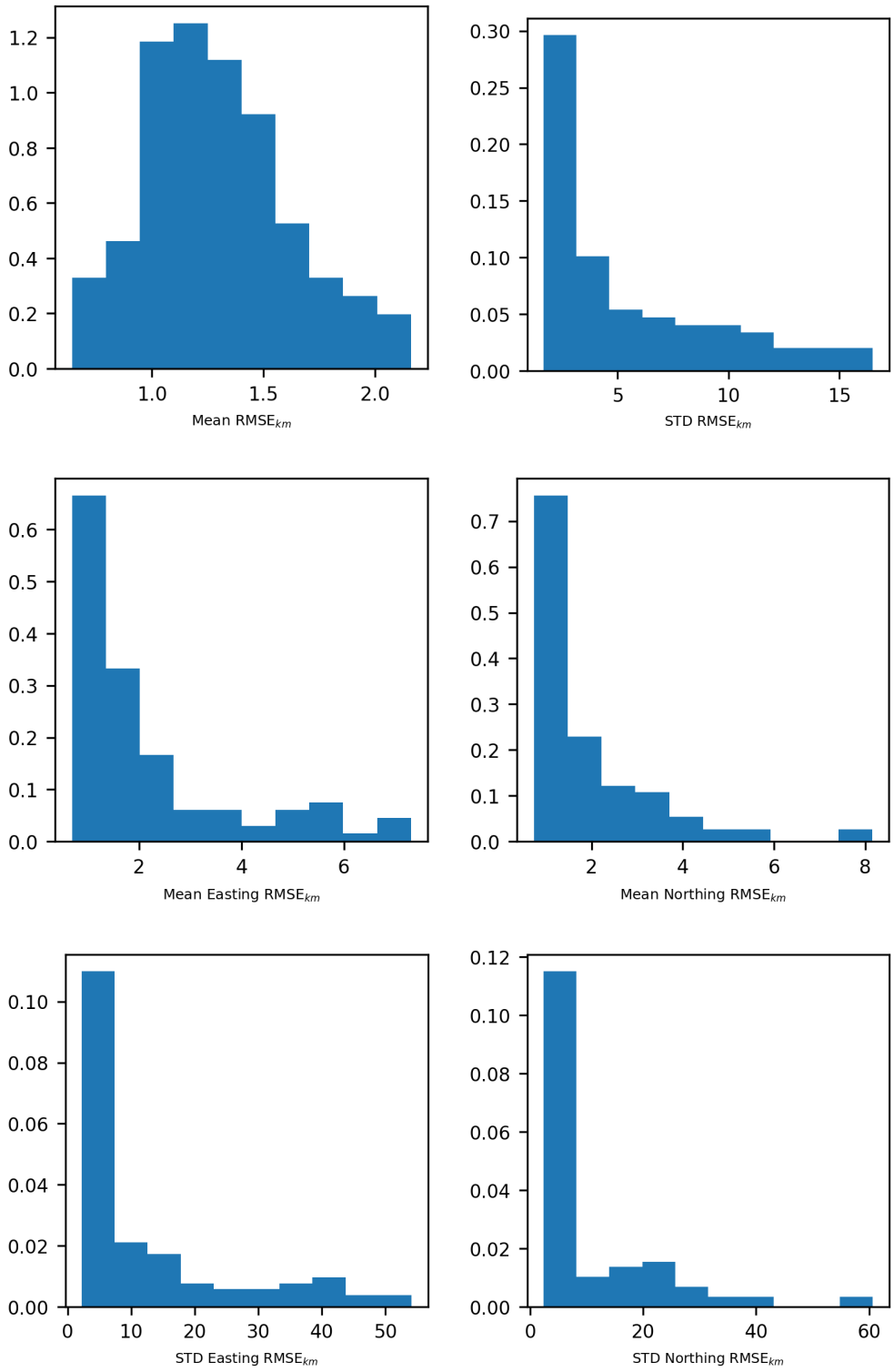
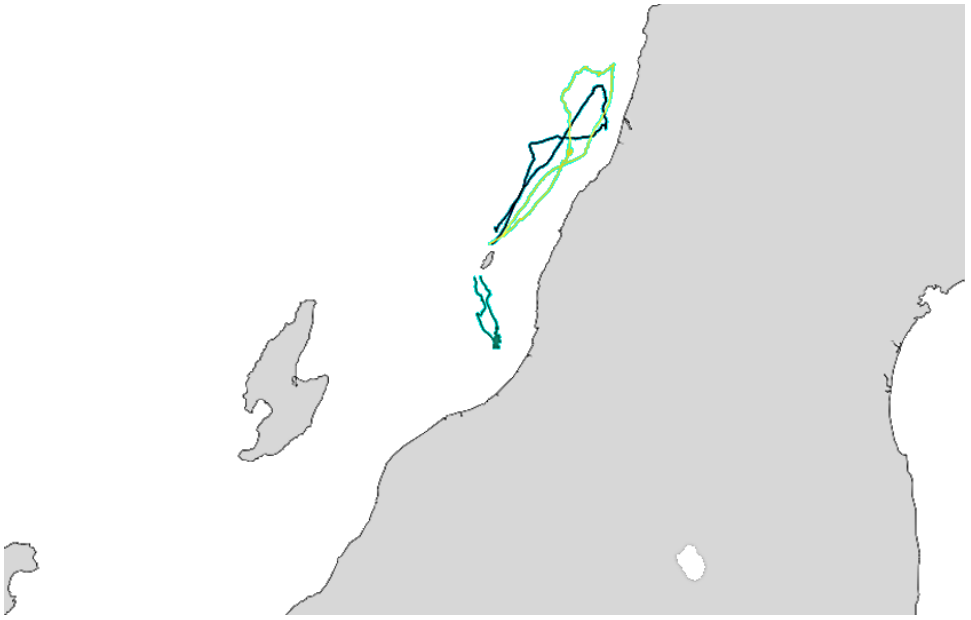
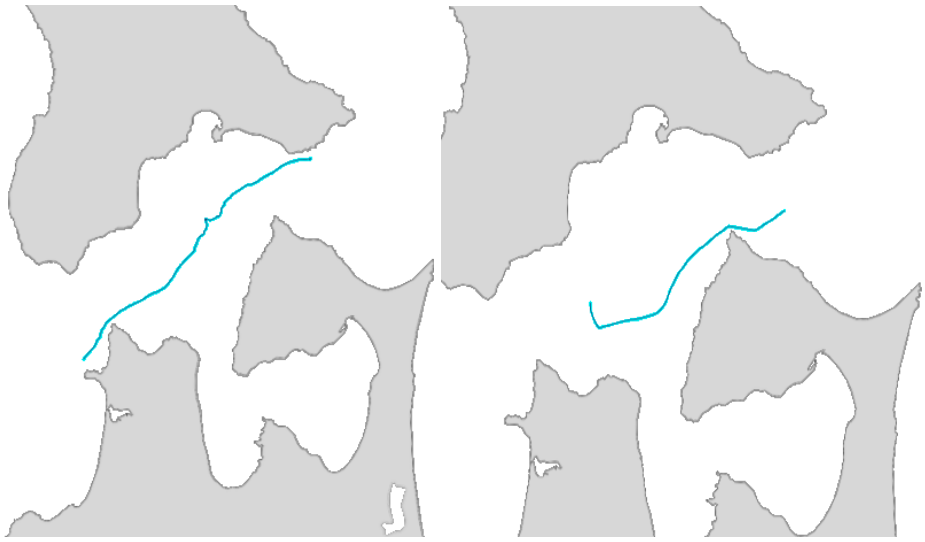


Figure 5.19: Distribution of trajectory segment reconstruction mean and std of RMSE measured in Euclidean distance, northing and easting dimensions in shearwaters trajectories.



(a)



(b) (c)

Figure 5.20: Sampled trajectory segments conditioned on geospatial and gender states. (a) Conditioned on the grid location of colony (b) Conditioned on the grid location at the straight and male gender

as the training data points are multi-modal with high variance.

More regrading the prediction and modeling, there was major considerations to put into account. Given the total number of trajectory points belonging to all birds available for training, there was the problem of multi-target or multi-modal modeling arising. This produced mixed and erratic test results. Therefore, in this work, we demonstrated using mixture density networks which has a probabilistic multi-component output structure. With these powerful networks, we were able to generate random trajectories conditioned on the geospatial grid data points which emulated the birds trajectories. Specifically, with addition of gender specific trajectory data, conditioning outputs on gender states, it demonstrates a specific bias towards certain directions accordingly. This reconfirms the results obtained in previous states regarding gender specific trajectory paths. Besides, training these networks on two different bird species, seagulls and shearwaters, we could generate trajectories particular to each species in terms of general navigational behavior.

The internal states of these networks could be very well used to classify the expressed behaviors along trajectories segments. For instance, one could use these vectors to identify random-walk behavior along trajectories which could be helpful to identify the stochastic movements in search of prey or fleeing predators along trajectories.

Unfortunately, while powerful in emulation or generating parallel paths, these networks provide very little insight into the mechanics of the animal movement. Therefore, their use should be with certain considerations about their viabilities in the intended applications.

Chapter 6

Visualization of Movement in Dynamic Environments

6.1 Introduction

In the recent decades, wildlife tracking technologies benefited significantly from advancements in silicon based processing and storage technologies. More efficient and higher capacity embedded systems with tiny footprints made it possible to attach smart tracking hardware to animals and record their trajectory information for significantly longer periods. Specifically in marine biology, equipped with such tracking devices, researchers gained unprecedented access into the daily life of marine birds. On the other hand, this increased the volume and scale of collected data significantly which required researchers to utilize new tools to visualize and analyze the recorded data. With regards to behaviors like migration which demonstrate a substantial shift in location of species, solely putting the spatial data on the map in chronological order might be satisfactory. However, for the analysis of behaviors which involve direct interaction with environment such as course correction

in windy environments, visualization rather requires inclusion of environment information such as wind direction and speed. Since these animals interact with their environments, it is essential to put the sensory data in their context for a more informative simulating visualization. With that in mind, this chapter introduces a software model that delivers chronological combination of species kinematics and environmental data visualizations. The overall objective is to present a more accurate illustration of organismal behaviors in their surrounding environment dynamics. Contents of this chapter were published in [176]

6.2 Design Methodology

6.2.1 Software Structure Model

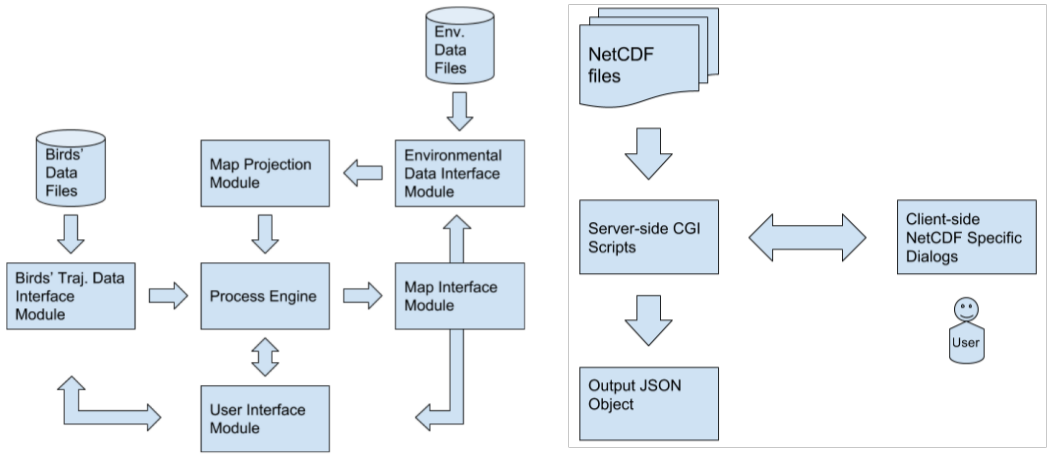
This software was designed around web application models like ERDDAP [177], OPeNDAP [178], etc. This makes it portable and accessible across various platforms from users' perspective. In terms of data storage and maintenance, this centralized approach puts the data in one place and identical set of data is accessed by researchers across an organization or communities. It also makes it possible to share computing resources and offload burden of performing computationally heavy analytic or preprocessing procedures locally on researchers' machines. For instance, interpolation of environmental maps, or coordinate projection of large data sets are being performed on server side. In addition, choice of mapping software could be customizable. This feature gives researchers flexibility to utilize their desired web-based mapping software. There are few downsides to this approach which are requirements of reliable connectivity and sufficient bandwidth. However, currently they are less of concern.

The overall structure of the proposed software is as follows. It consists of 6 essential

functioning blocks as shown in Figure 6.1(a). First is trajectory data management interface which handles data files and provides the kinematic data to the application. This encapsulates task of parsing and preprocessing data file for mapping interface. The second block is responsible for loading and parsing environmental data and presenting them either in form of spatial or spatiotemporal measurement arrays to map projection block. The Third block handles the projection of the environmental data arrays. This gives the software the capability of using maps with custom projection standards. The Fourth block handles the main processes for data synchronization, analysis and visualization animations. The Fifth block interfaces with map visualization software. This block provides the data overlays and trajectories to the mapping application. The last block handles user inputs and settings for data selection and visualization. This modular structure makes the application both customizable and scalable. For instance, to handle a new set of data file types, it simply requires only to create a new data file parser and if necessary compatible user interface dialogs without hindering the work of other modules as illustrated in Figure 6.1(b). Moreover, the server-side software can be programmed in any server supported Common Gateway Interface (CGI) scripting language, since every frontend data set handler object encapsulates its own CGI target. The server-client model enables system to be highly scalable on the backend, while on the client side, the hardware performance would only determine the amount of data to be displayed or analyzed locally.

6.2.2 Trajectory Data Interface Module

The trajectory data is provided to the mapping interface in at least 3 arrays of latitudes, longitudes, timestamps. The main function of this module is generally retrieving, parsing and preprocessing data from database or directly from sensors. Standard output of this



(a) (b)

Figure 6.1: (a) The Essential functioning blocks of the software (b) Candidate data file handler model

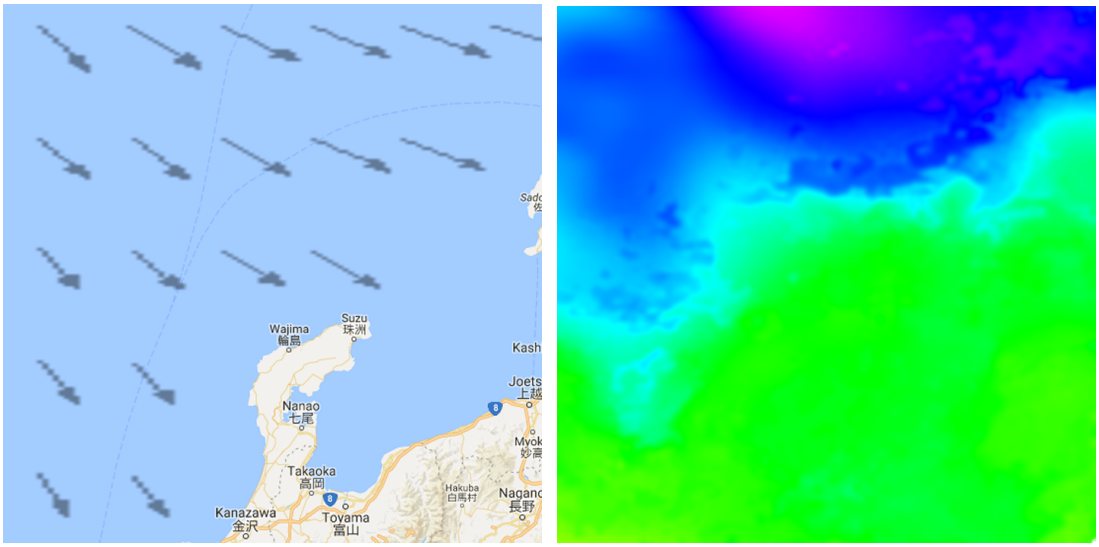
module is JSON object containing mentioned data arrays in chronological order. This module interacts with users through user interface block.

6.2.3 Environmental Data Interface Module

Environmental data, specifically oceanic and atmospheric measurements are stored in variety of formats. The essential information required for visualization is spatial coordinates and respecting measurements. Therefore, task of this block is to retrieve, parse, and preprocess environmental data set files. This module also communicates with user via the user interface block for the selection of desired measurements.

6.2.4 Map Projection Module

Generally, environmental data are visualized on map as overlay images. These overlays, depending on the dimension of their measurement variable or variables could be color



(a) (b)

Figure 6.2: (a) Map overlay image of wind speed and direction vectors. Map data is courtesy of ©2017 Google, ZENRIN and SK telecom. (b) Rainbow color map image of sea surface temperature values.

maps, vector fields, or color coded vector fields. For instance, wind, or ocean current directions are visualized as vector overlays. Figure 6.2(a) shows an instance of wind speed overlay. In contrast, scalar measurement like sea surface temperatures are visualized as color maps where temperature values are mapped into a color along a color spectrum like Rainbow as shown in Figure 6.2(b). Based on the projection standard required by the mapping application, spatial mapping transform of the spatial measurements onto teh image layers are generated and passed to the mapping interface block. In addition, map projection module does optionally perform preprocessing and interpolation. Choices of mapping scale, color map spectrum and data aggregations are customizable and presented to the user through the user interface module.

6.2.5 Process Engine

The process engine module is the main processing block of the visualization software. It handles the task of synchronizing trajectory data and environmental maps and transferring them to the map interface module. Additionally, it controls the sampling rate and locally performed windowed analytical functions on trajectory data configured through the user interface module. This module expects standard protocol inputs from the trajectory and environment modules.

6.2.6 Map Interface Module

The map Interface module is the communication link between the visualization and mapping softwares. It handles the task of transferring trajectory data and environmental maps for rendering to mapping applications. As a result, the choice of mapping application could be diversified by adding different map interfaces to the software.

6.2.7 User Interface Module

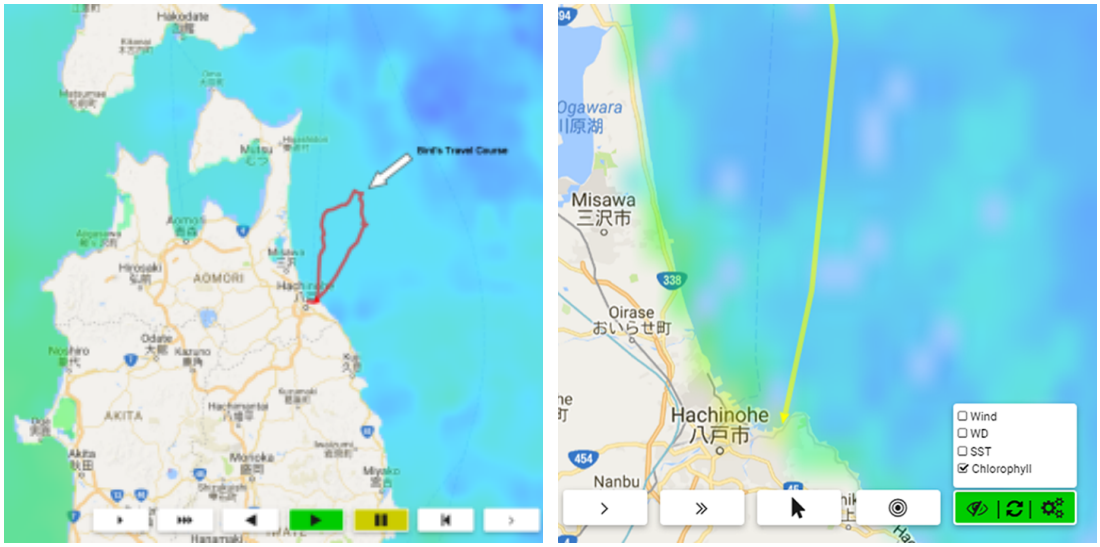
Lastly, the final essential module is user interface which communicates user configurations to other blocks. This module encapsulates the frontend dialogs, libraries and backend CGI functions interfacing with other blocks if necessary. For instance, map interface block dialogs require no backend implementations while the first three blocks require both frontend and backend implementations.

6.3 Major Functionalities

To demonstrate the major functionalities of the proposed software, a prototype is presented here. This prototype has the following setup. Its trajectory module handles CSV file types. The user selects the spatial and temporal data columns through the user interface. In addition, auxiliary data columns could be provided and visualized by dashboard plots or heatmap layers. The environmental data interface includes functionality to retrieve and parse data from NetCDF [179] and CSV file types. This data could be either solely spatial or spatiotemporal in which map interface engine optionally updates the spatial data per time stamps of trajectory data. The map application is selected to be Google Maps [60] and mapping projection uses Web Mercator projection [180] to produce environmental data image overlays. The analytic engine contains windowed operations and filters such as mean, variance, etc. and visualizes the results on dashboard plots.

6.3.1 Itinerary Reconstruction

The primary function of this prototype designed to be the reconstruction bird travel routes including the environmental parameters. Timeline and sampling rate controls allow users to skim through the itineraries and observe the trends or peculiarities of the animal behavior with regards to their ambient features. In addition, users have ability to reverse, pause or manually advance the timeline as shown in Figure 6.3(a). Overlay controls allow users to enable, disable, hide or update the environment data overlays as demonstrated in Figure 6.3(b).



(a) (b)

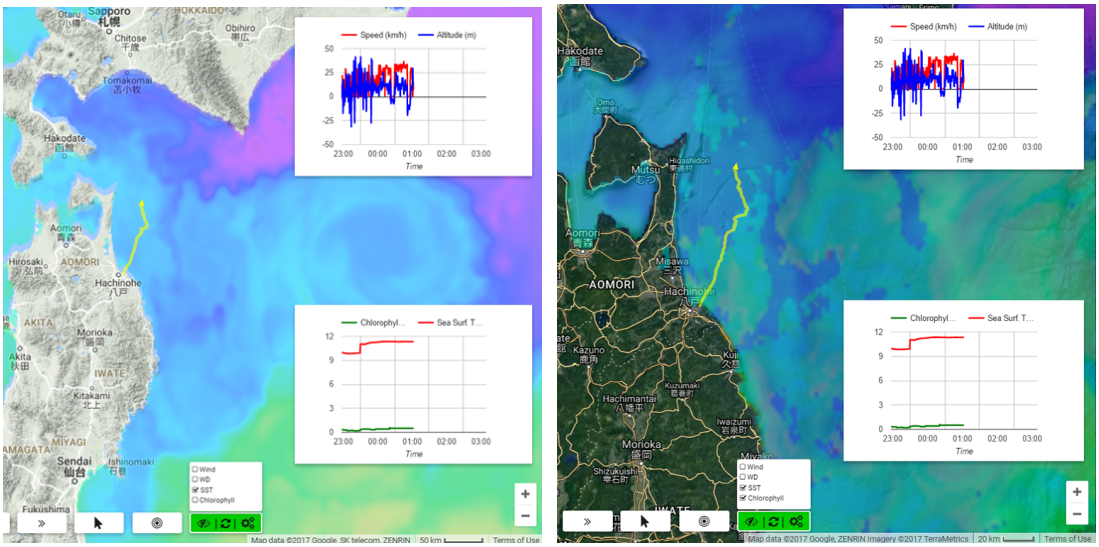
Figure 6.3: (a) Timeline controls in interactive visualization of bird trajectory. (b) Overlay controls for environment data layers. Maps data are courtesy of ©2017 Google, ZENRIN and SK telecom.

6.3.2 Windowed Analysis Visualization

The Second major tool available in the presented software does perform windowed time series analysis on the trajectory data. These could be selected from a library of functions supported by the application. The results are visualized using plots or with heat map layers directly on the map. This tool also could be utilized for filtering of the trajectory data retrieved from sensors directly. Figure 6.4 shows samples of windowed analysis plots. Figure 6.5 shows heatmaps visualizing second moment or variance of the velocity over a centralized window at each sample point.

6.3.3 Multiple Object Visualization

An another feature supported by this application is the ability to synchronize multiple trajectories' data and visualize their paths on the map. This is useful for studying flocking,



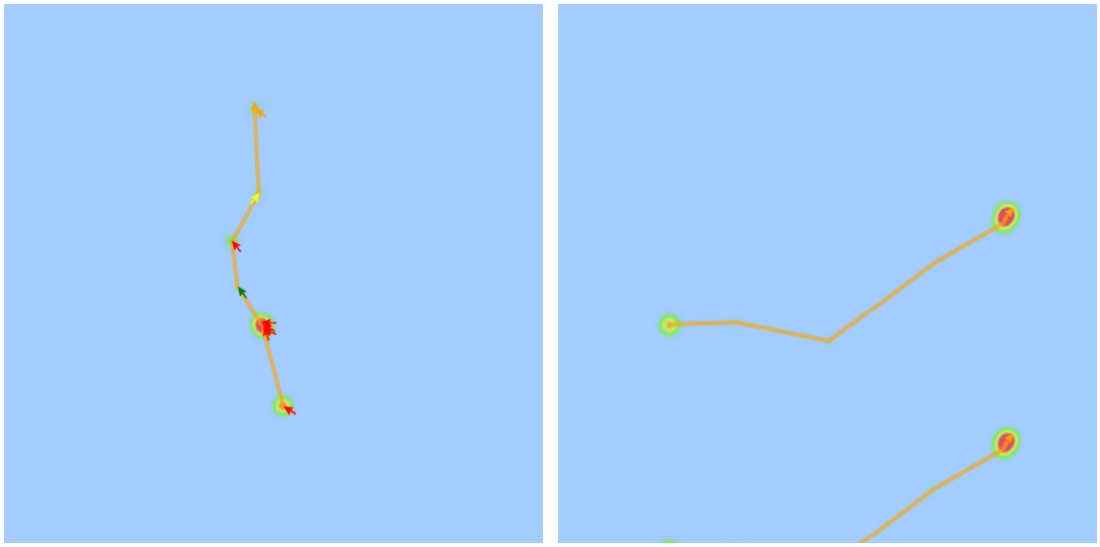
(a) (b)

Figure 6.4: Moving average plots of the selected variables. (a) Sea surface temperatures overlay. Map data is courtesy of ©2017 Google, ZENRIN and SK telecom. (b) Sea surface temperatures and chlorophyll level overlays. All map data and imagery are courtesy of ©2017 Google, ZENRIN imagery and ©2017 TerraMetrics.

foraging and group migration behavior in the seabirds. There are also aggregation functions available to be applied on multiple birds' data. An example of such function is centroid of a cluster of multiple birds' data points.

6.4 Conclusions

In summary, the presented application here, provides a comprehensive, flexible and interactive tool for visualizing animals movement paths combined with environment variables along those routes. This benefits the researchers in a sense that investigating the latent influences of environmental stimuli on movement behavior organisms in dynamical environments becomes more tangible. It gives researchers ability to apply various functions on the contemporary data and observe the results visually along the procession of trajectories.



(a) (b)

Figure 6.5: Visualizing centralized sample moving variance of velocities using weighted heat maps. (a) Single trajectory. (b) Multiple trajectories. Maps data are courtesy of ©2017 Google, ZENRIN and SK telecom.

Also from development standpoint, the modular structure of the software allows expandability and compatibility. The support for more data types or maps could be added easily without interfering with existing functionalities. An anticipating upgrade to this software is adding support for interactive labeling of trajectory points in order to be used in training of supervised machine learning models. Interactive visual interface available in this application provides foundation for developing such functionalities.

Chapter 7

Conclusions

In this chapter, we recap the topics discussed in this thesis and briefly review the results and conclusions reached at the end of each chapter with a suggested roadmap forward for follow up works on the problems approached in this thesis, and lastly, our final thoughts. An overview diagram is shown in Figure 7.1. In general, this thesis was centered around the application of data science and computer science in ecology of animal movement. It was motivated by the fact that, studying animal movements help us to understand their behavior and consequently their environment, and trajectory data mining techniques provide essential solutions for achieving this efficiently at larger scales.

In Chapter 2, we started with trajectory data acquisition from stereo images, presented as a case study of bat trajectory reconstruction. A solution to the problem of object detection, identification and correspondence in order to reconstruct 3D trajectories of flying bats was proposed in form of multi-stage motion-based 3D trajectory reconstruction algorithm. The proposed model-based technique attempts to estimate 3D motion of the moving components in the scene and identify the desired objects based on their movement model. It was demonstrated that in comparison with single view 2D tracking methods that are commonly

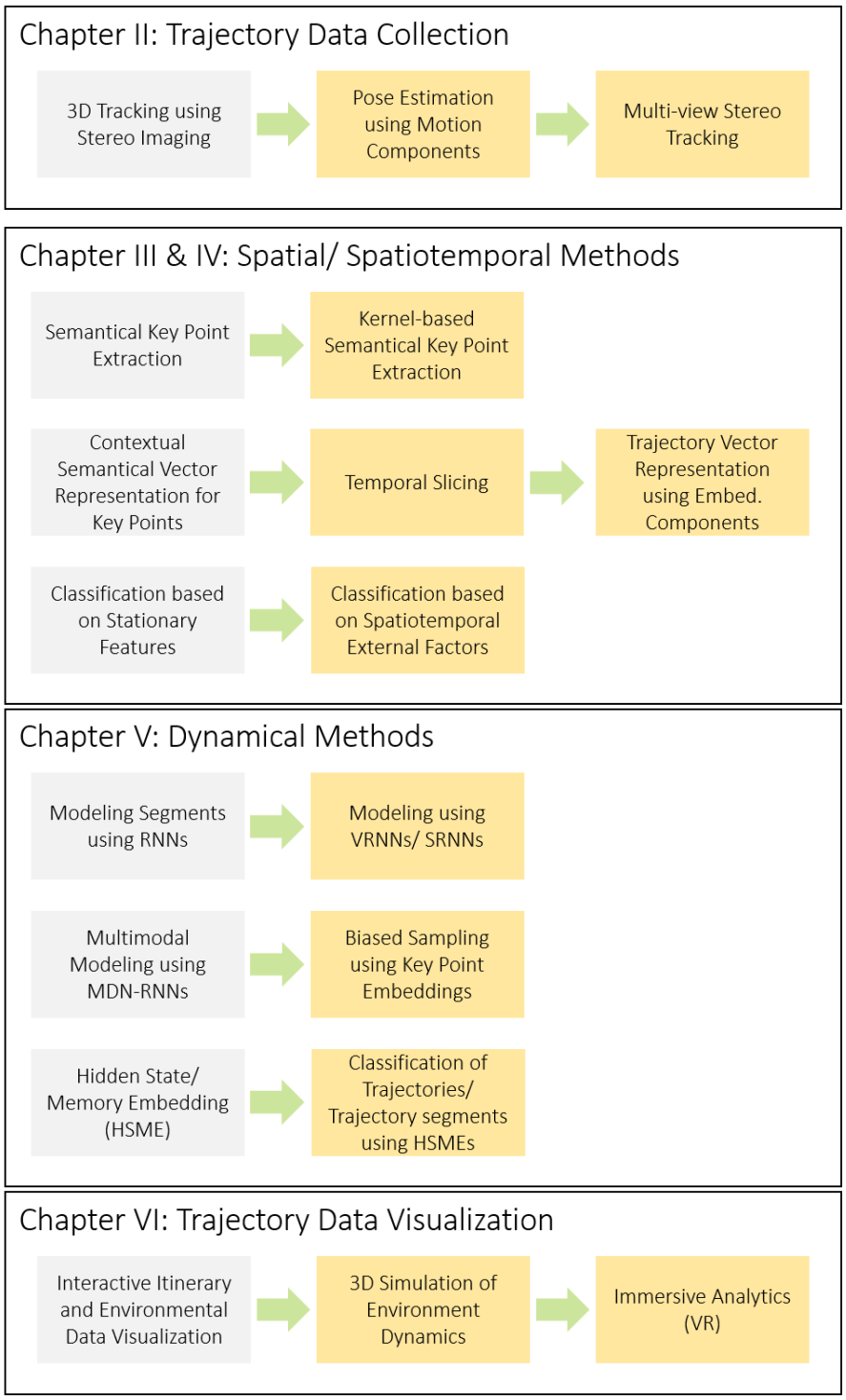


Figure 7.1: A modular overview of this thesis and suggested follow up works shaded in yellow.

used, the proposed approached performed better, specifically in scenarios with occlusions caused by multi-object path crossings. It should be noted that this performance advantage was achieved at a computational cost which makes this approach an offline method and not suitable for densely populated scenes. Therefore, improving the performance is considered a follow up objective which could be approached in both algorithm design and software implementation. Furthermore, pose estimation using motion components and multi-view stereo configurations for solving occlusion problems are also worthy of being explored . Contributions in this chapter provide a significant assistance to researchers studying bat trajectories in terms of trajectory data collection. Then, the collected trajectory data could be used in conjunction with methods presented in Chapter 5 to model the movement of flying bats.

In Chapter 3, we focused on extracting spatial features from animal movement which describe behaviors or trajectories specific to a particular group of species. We have attempted to exploit spatial features and properties forged in a grid-less form in trajectories as results of cognitive processes in focal organisms. This leads us to identification of those particular organisms associated with such features. We designated these features to key points along trajectories which could identify internal states, navigation capacities, motion capacities and even the state external factors in animal movement paths. We argued that, the sequential order between these key points turn irrelevant given such information. In other words, they become independent of previous point in temporal domain conditioned on certain information about their internal state, motion capacity, navigation capacity or external factors. To test this, this approach was employed for gender-based classification of a species of marine birds. With rigorous testing we demonstrated that in fact using key points, given a set of trajectories of an organism, the gender could be determined. With this

approach, input feature space dimension was significantly reduced which could be very advantageous in dealing with large scale data sets. However, there were also encounters with issues regarding key point extraction methods which could cause instability in the results. Hence, we suggest further work on key point extraction methods as follow up study for Chapter 3. Besides, further work could be done in modeling of key points distributions over temporal slices of the geospatial plane using probabilistic inference . This may provide clues about the effects of external factors like environment on distribution of the key points in trajectories.

Chapter 4 followed up on the issue of the efficient and informative extraction and representation of trajectory key points from the previous chapter. In this chapter, a density-based hierarchical approach towards key point extraction was taken in order to recognize contingency of the key points being along an itinerary. This opens the gate to extraction of more complex semantical information from simply geospatial coordinates. However, this would increase the input feature space dimension significantly. In order to remedy that, contextual embedding of input feature space inspired by Skip-gram model was used to project the input space onto a more locally informative embedding space. These embedding vectors provide a more informative numerical representation for key points along trajectories. It is also possible to identify the main components of the input space based on the topology of the embedding space. As a result, it becomes possible to represent trajectories with fixed length vectors as well. It was demonstrated that using embedding vectors in place of simply one-hot vectors as inputs improved the classification results significantly. It was also illustrated that with this approach, it was possible to embed wide range of semantical information into representation space which could lead to improvements in classification results. It was also noted that there was still issue of stability lingering which could be originating

from key point extraction. As stated before, follow up work on key point extraction techniques is suggested. Another takeaway from this chapter was the potential in identification of trajectories with similar sequential elements. It is a powerful tool in data exploration, feature correlation analysis and pattern discovery. This would also be beneficial in discovering contingent factors like environment events that influenced the movement paths. This as well, suggested as a follow up work. Certainly, topography and dynamical elements of the environment are huge determinants of the distribution of key points along movement paths.

In Chapter 5, we focused on sequential dynamics of animal movement. We presented two LSTM based models for encoding prominent information about structure of movement path. First we employed undercomplete recurrent autoencoder models to investigate the capability of such networks in encoding dynamics of trajectories. It was demonstrated that multimodal nature of trajectory data impeded the expected performance. It was seen that, these models tended to produce averaged results for multimodal outputs. To tackle this issue, we employed mixture density network as the output layer of the recurrent network. The outcome shown to be improved greatly in addition to gaining ability for generating movement paths. This ability in conjunction with slight modifications in the network was used to generate trajectories conditioned on certain prior information like gender or geospatial information. This suggested that the hidden and memory states of recurrent network maintain such information. This information encoded in internal states of the network could be utilized to discriminate or compare trajectory segments or trajectories in whole. This is comparable to hidden Markov models used for similar purposes. However, in contrast to HMMs, LSTMs have ability to encode trajectory information and events on a continuous manifold as distributed representations rather than discrete states. The results shown that

these networks were able to learn dynamical features particular to different species or different groups within a same species with distinguishable movement patterns or dynamics. One topic which was not addressed in this chapter was probabilistic modeling of transitions between trajectory steps. The latter in addition to exploring the application of variational recurrent models in discovering latent structures in animal movement data are suggested as a follow up endeavors. These probabilistic approaches towards state transitions may provide better descriptions of procession of internal and external states along trajectories of animals without having direct access to those states.

Through this research a strong necessity for a visualization tool which enables simulation of environmental factors and conditions along animal movement was sensed. In Chapter 6, we proposed a software model which enables researchers simple and comprehensive access to the available environmental data associated with the collected trajectory data. This was accompanied by an interactive visualization tool capable of reconstruction of trajectories in their environments with desired variables. This certainly help researchers having a better understanding of animal movement influenced by topological and environmental factors. This work could be followed up by addition of advance feature components such as virtual reality and 3D simulation of environment dynamics which accommodate immersive analytics solutions for the researchers.

Overall, from trajectory modeling perspective, this work presented three major approaches to modeling trajectories that are geospectral, geospatiotemporal, and dynamical. In the first approach, only geospatial data was used for modeling. In the second one, temporal domain was utilized but not as the primary variable and in the third approach time was the primary variable in modeling the trajectories. Utilization of each approach certainly depends on the objectives and type of data available to the researchers. This choice

of approach is critical in efficiency, fitness and viability of the results. It is worth mentioning again that, a major process involved in animal movement is cognitive process. While trajectory data consists of sequence of numerical coordinates, these numbers are encoded as semantical representations in organism cognitive process. Thence, discovering and employing such representation spaces are essential in analyzing, modeling and describing animal movement patterns.

Bibliography

[1] Heiko Hirschmuller. Stereo processing by semiglobal matching and mutual information. *IEEE Transactions on pattern analysis and machine intelligence*, 30(2):328–341, 2007.

[2] Ran Nathan, Wayne M Getz, Eloy Revilla, Marcel Holyoak, Ronen Kadmon, David Saltz, and Peter E Smouse. A movement ecology paradigm for unifying organismal movement research. *Proceedings of the National Academy of Sciences*, 105(49):19052–19059, 2008.

[3] Movebank - movebank for animal tracking data. `https://www.movebank.org/`. Last Accessed: 2019-06-30.

[4] Francesca Cagnacci, Luigi Boitani, Roger A Powell, and Mark S Boyce. Animal ecology meets gps-based radiotelemetry: a perfect storm of opportunities and challenges, 2010.

[5] Ferdinando Urbano, Francesca Cagnacci, Clément Calenge, Holger Dettki, Alison Cameron, and Markus Neteler. Wildlife tracking data management: a new vision. *Philosophical Transactions of the Royal Society B: Biological Sciences*, 365(1550):2177–2185, 2010.

[6] Urška Demšar, Kevin Buchin, Francesca Cagnacci, Kamran Safi, Bettina Speckmann, Nico Van de Weghe, Daniel Weiskopf, and Robert Weibel. Analysis and visualisation of movement: an interdisciplinary review. *Movement Ecology*, 3(1):5, 2015.

[7] Natalia Adrienko and Gennady Adrienko. Spatial generalization and aggregation of massive movement data. *IEEE Transactions on visualization and computer graphics*, 17(2):205–219, 2011.

[8] Travis J Ryan, Christopher A Conner, Brooke A Douthitt, Sean C Sterrett, and Carmen M Salsbury. Movement and habitat use of two aquatic turtles (graptemys geographica and trachemys scripta) in an urban landscape. *Urban Ecosystems*, 11(2):213–225, 2008.

[9] Collin P Jaeger and Vincent A Cobb. Comparative spatial ecologies of female painted turtles (chrysemys picta) and red-eared sliders (trachemys scripta) at reelfoot lake, tennessee. *Chelonian conservation and biology*, 11(1):59–67, 2012.

[10] Kei Matsumoto, Nariko Oka, Daisuke Ochi, Fumihito Muto, Takashi P Satoh, and Yutaka Watanuki. Foraging behavior and diet of streaked shearwaters calonectris leucomelas rearing chicks on mikura island. *Ornithological Science*, 11(1):9–19, 2012.

[11] Sakiko Matsumoto, Takashi Yamamoto, Maki Yamamoto, Carlos B Zavalaga, and Ken Yoda. Sex-related differences in the foraging movement of streaked shearwaters calonectris leucomelas breeding on awashima island in the sea of japan. *Ornithological Science*, 16(1):23–32, 2017.

[12] Ivan Tiunov, Igor Katin, Hansoo Lee, Siwan Lee, and Eunhong Im. Foraging areas of streaked shearwater calonectris leucomelas nesting on the karamzin island (peter the great bay, east sea). *Journal of Asia-Pacific Biodiversity*, 11(1):25–31, 2018.

[13] Yusuke Goto, Ken Yoda, and Katsufumi Sato. Asymmetry hidden in birds' tracks reveals wind, heading, and orientation ability over the ocean. *Science advances*, 3(9):e1700097, 2017.

[14] Yu Zheng and Xiaofang Zhou. *Computing with spatial trajectories*. Springer Science & Business Media, 2011.

[15] Jean Damascène Mazimpaka and Sabine Timpf. Trajectory data mining: A review of methods and applications. *Journal of Spatial Information Science*, 2016(13):61–99, 2016.

[16] Yu Zheng. Trajectory data mining: an overview. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 6(3):29, 2015.

[17] Siyuan Liu, Shuhui Wang, and Qiang Qu. *Trajectory Mining*, pages 2310–2313. Springer International Publishing, Cham, 2017.

[18] Renchu Song, Weiwei Sun, Baihua Zheng, and Yu Zheng. Press: A novel framework of trajectory compression in road networks. *Proceedings of the VLDB Endowment*, 7(9):661–672, 2014.

[19] Minjie Chen, Mantao Xu, and Pasi Franti. Compression of gps trajectories. In *2012 Data Compression Conference*, pages 62–71. IEEE, 2012.

[20] Wang-Chien Lee and John Krumm. Trajectory preprocessing. In *Computing with spatial trajectories*, pages 3–33. Springer, 2011.

[21] Filip Biljecki, Hugo Ledoux, and Peter van Oosterom. Transportation mode-based segmentation and classification of movement trajectories. *International Journal of Geographical Information Science*, 27(2):385–407, 2013.

[22] Jae-Gil Lee, Jiawei Han, and Kyu-Young Whang. Trajectory clustering: A partition-and-group framework. In *Proceedings of the 2007 ACM SIGMOD International Conference on Management of Data*, SIGMOD '07, pages 593–604, New York, NY, USA, 2007. ACM.

[23] Yu Zheng, Like Liu, Longhao Wang, and Xing Xie. Learning transportation mode from raw gps data for geographic applications on the web. In *Proceedings of the 17th international conference on World Wide Web*, pages 247–256. ACM, 2008.

[24] Lai Wei. Birds/bats motion tracking with infrared radiation camera for wind farm applications. 2012.

[25] Xiaodong Tian, Jose Iriarte-Diaz, Kevin Middleton, Ricardo Galvao, Emily Israeli, Abigail Roemer, Allyce Sullivan, Arnold Song, Sharon Swartz, and Kenneth Breuer. Direct measurements of the kinematics and dynamics of bat flight. *Bioinspiration & biomimetics*, 1(4):S10, 2006.

[26] Attila J Bergou, Sharon M Swartz, Hamid Vejdani, Daniel K Riskin, Lauren Reimnitz, Gabriel Taubin, and Kenneth S Breuer. Falling with style: bats perform complex aerial rotations by adjusting wing inertia. *PLoS biology*, 13(11):e1002297, 2015.

[27] William R Elliott and RL Clawson. Gray and indiana bat population trends in missouri. In *Proceedings of the National Cave and Karst Management Symposium*, volume 18, pages 46–61, 2008.

[28] José Iriarte-Díaz and Sharon M Swartz. Kinematics of slow turn maneuvering in the fruit bat cynopterus brachyotis. *Journal of Experimental Biology*, 211(21):3478–3489, 2008.

[29] William R Elliott, ST Samoray, SE Gardner, and JE Kaufmann. The mdc method: Counting bats with infrared video. In *Proceedings of the 2005 National Cave and Karst Management Symposium, Albany, New York*, pages 147–153, 2006.

[30] Peter Windes, Xiaozhou Fan, Matt Bender, Danesh K Tafti, and Rolf Müller. A computational investigation of lift generation and power expenditure of pratt's roundleaf bat (hipposideros pratti) in forward flight. *PloS one*, 13(11):e0207613, 2018.

[31] Attila J Bergou, Sharon Swartz, Kenneth Breuer, and Gabriel Taubin. 3d reconstruction of bat flight kinematics from sparse multiple views. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, pages 1618–1625. IEEE, 2011.

[32] Wu-Jung Lee, Benjamin Falk, Chen Chiu, Anand Krishnan, Jessica H Arbour, and Cynthia F Moss. Tongue-driven sonar beam steering by a lingual-echolocating fruit bat. *PLoS biology*, 15(12):e2003148, 2017.

[33] Yang Ye, Yu Zheng, Yukun Chen, Jianhua Feng, and Xing Xie. Mining individual life pattern based on location history. In *Mobile Data Management: Systems, Services and Middleware, 2009. MDM'09. Tenth International Conference on*, pages 1–10. IEEE, 2009.

[34] John M Fryxell, Megan Hazell, Luca Börger, Ben D Dalziel, Daniel T Haydon, Juan M Morales, Therese McIntosh, and Rick C Rosatte. Multiple movement modes

by large herbivores at multiple spatiotemporal scales. *Proceedings of the National Academy of Sciences*, pages pnas–0801737105, 2008.

[35] You Wan, Chenghu Zhou, and Tao Pei. Semantic-geographic trajectory pattern mining based on a new similarity measurement. *ISPRS International Journal of Geo-Information*, 6(7):212, 2017.

[36] Wayne Xin Zhao, Ningnan Zhou, Aixin Sun, Ji-Rong Wen, Jialong Han, and Edward Y Chang. A time-aware trajectory embedding model for next-location recommendation. *Knowledge and Information Systems*, pages 1–21, 2018.

[37] Peng Gao, John A Kupfer, Xi Zhu, and Diansheng Guo. Quantifying animal trajectories using spatial aggregation and sequence analysis: a case study of differentiating trajectories of multiple species. *Geographical Analysis*, 48(3):275–291, 2016.

[38] Matthias Studer, Gilbert Ritschard, Alexis Gabadinho, and Nicolas S Müller. Discrepancy analysis of state sequences. *Sociological Methods & Research*, 40(3):471–510, 2011.

[39] Alexis Gabadinho, Gilbert Ritschard, Nicolas Séverin Mueller, and Matthias Studer. Analyzing and visualizing state sequences in r with traminer. *Journal of Statistical Software*, 40(4):1–37, 2011.

[40] Greg A Breed, Daniel P Costa, Ian D Jonsen, Patrick W Robinson, and Joanna Mills-Flemming. State-space methods for more completely capturing behavioral dynamics from animal tracks. *Ecological Modelling*, 235:49–58, 2012.

[41] Wayne M Getz and David Saltz. A framework for generating and analyzing move-

ment paths on ecological landscapes. *Proceedings of the National Academy of Sciences*, 105(49):19066–19071, 2008.

[42] Martin F Breed, Maria HK Marklund, Kym M Ottewell, Michael G Gardner, J Berton C Harris, and Andrew J Lowe. Pollen diversity matters: revealing the neglected effect of pollen diversity on fitness in fragmented landscapes. *Molecular Ecology*, 21(24):5955–5968, 2012.

[43] Martin F Breed, Michael G Stead, Kym M Ottewell, Michael G Gardner, and Andrew J Lowe. Which provenance and where? seed sourcing strategies for revegetation in a changing environment. *Conservation Genetics*, 14(1):1–10, 2013.

[44] R Terry Bowyer. Sexual segregation in southern mule deer. *Journal of Mammalogy*, 65(3):410–417, 1984.

[45] Kelley M Stewart, Timothy E Fulbright, D Lynn Drawe, and R Terry Bowyer. Sexual segregation in white-tailed deer: responses to habitat manipulations. *Wildlife Society Bulletin*, pages 1210–1217, 2003.

[46] Ken Yoda, Kozue Shiomi, and Katsufumi Sato. Foraging spots of streaked shearwaters in relation to ocean surface currents as identified using their drift movements. *Progress in Oceanography*, 122:54–64, 2014.

[47] Ken Yoda, Takashi Yamamoto, Hirokazu Suzuki, Sakiko Matsumoto, Martina Müller, and Maki Yamamoto. Compass orientation drives naïve pelagic seabirds to cross mountain ranges. *Current Biology*, 27(21):R1152–R1153, 2017.

[48] David R Brillinger, Haiganoush K Preisler, Alan A Ager, and John G Kie. An

exploratory data analysis (eda) of the paths of moving animals. *Journal of statistical planning and inference*, 122(1-2):43–63, 2004.

[49] James D Forester, Anthony R Ives, Monica G Turner, Dean P Anderson, Daniel Fortin, Hawthorne L Beyer, Douglas W Smith, and Mark S Boyce. State–space models link elk movement patterns to landscape characteristics in yellowstone national park. *Ecological Monographs*, 77(2):285–299, 2007.

[50] Sara M Maxwell, Greg A Breed, Barry A Nickel, Junior Makanga-Bahouna, Edgard Pemo-Makaya, Richard J Parnell, Angela Formia, Solange Ngouessono, Brendan J Godley, Daniel P Costa, et al. Using satellite tracking to optimize protection of long-lived marine species: olive ridley sea turtle conservation in central africa. *PloS one*, 6(5):e19905, 2011.

[51] F Royer, J-M Fromentin, and P Gaspar. A state–space model to derive bluefin tuna movement and habitat from archival tags. *Oikos*, 109(3):473–484, 2005.

[52] Théo Michelot, Roland Langrock, and Toby A Patterson. movehmm: an r package for the statistical modelling of animal movement data using hidden markov models. *Methods in Ecology and Evolution*, 7(11):1308–1315, 2016.

[53] Rory Gibb, Akiko Shoji, Annette L Fayet, Chris M Perrins, Tim Guilford, and Robin Freeman. Remotely sensed wind speed predicts soaring behaviour in a wide-ranging pelagic seabird. *Journal of the Royal Society Interface*, 14(132):20170262, 2017.

[54] Ran Nathan, Orr Spiegel, Scott Fortmann-Roe, Roi Harel, Martin Wikelski, and Wayne M Getz. Using tri-axial acceleration data to identify behavioral modes of free-ranging animals: general concepts and tools illustrated for griffon vultures. *Journal of Experimental Biology*, 215(6):986–996, 2012.

[55] Ella Browning, Mark Bolton, Ellie Owen, Akiko Shoji, Tim Guilford, and Robin Freeman. Predicting animal behaviour using deep learning: Gps data alone accurately predict diving in seabirds. *Methods in Ecology and Evolution*, 9(3):681–692, 2018.

[56] Tsubasa Hirakawa, Takayoshi Yamashita, Toru Tamaki, Hironobu Fujiyoshi, Yuta Umezu, Ichiro Takeuchi, Sakiko Matsumoto, and Ken Yoda. Can ai predict animal movements? filling gaps in animal trajectories using inverse reinforcement learning. *Ecosphere*, 9(10), 2018.

[57] J. D. Hunter. Matplotlib: A 2d graphics environment. *Computing in Science & Engineering*, 9(3):90–95, 2007.

[58] John D Hunter. Matplotlib: A 2d graphics environment. *Computing in science & engineering*, 9(3):90–95, 2007.

[59] HoloViews: Stop plotting your data - annotate your data and let it visualize itself. `http://holoviews.org/`. Last Accessed: 2019-06-30.

[60] Google maps platform – documentation. `https://developers.google.com/maps/documentation/`. Last Accessed: 2019-06-30.

[61] GeoPy - geopy's documentation. `https://geopy.readthedocs.io/en/stable/`. Last Accessed: 2019-06-30.

[62] Stefan Van Der Walt, S Chris Colbert, and Gael Varoquaux. The numpy array: a structure for efficient numerical computation. *Computing in Science & Engineering*, 13(2):22, 2011.

[63] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[64] Wes McKinney. Data structures for statistical computing in python. In Stéfan van der Walt and Jarrod Millman, editors, *Proceedings of the 9th Python in Science Conference*, pages 51 – 56, 2010.

[65] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[66] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of machine learning research*, 12(Oct):2825–2830, 2011.

[67] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–. [Online; accessed ¡today¿].

[68] Rami Al-Rfou, Guillaume Alain, Amjad Almahairi, Christof Angermueller, Dzmitry Bahdanau, Nicolas Ballas, Frédéric Bastien, Justin Bayer, Anatoly Belikov, Alexander Belopolsky, Yoshua Bengio, Arnaud Bergeron, James Bergstra, Valentin Bisson, Josh Bleecher Snyder, Nicolas Bouchard, Nicolas Boulanger-Lewandowski, Xavier Bouthillier, Alexandre de Brébisson, Olivier Breuleux, Pierre-Luc Carrier, Kyunghyun Cho, Jan Chorowski, Paul Christiano, Tim Cooijmans, Marc-Alexandre Côté, Myriam Côté, Aaron Courville, Yann N. Dauphin, Olivier Delalleau, Julien Demouth, Guillaume Desjardins, Sander Dieleman, Laurent Dinh, Mélanie Ducoffe, Vincent Dumoulin, Samira Ebrahimi Kahou, Dumitru Erhan, Ziye Fan, Orhan Firat, Mathieu Germain, Xavier Glorot, Ian Goodfellow, Matt Graham, Caglar

Gulcehre, Philippe Hamel, Iban Harlouchet, Jean-Philippe Heng, Balázs Hidasi, Sina Honari, Arjun Jain, Sébastien Jean, Kai Jia, Mikhail Korobov, Vivek Kulkarni, Alex Lamb, Pascal Lamblin, Eric Larsen, César Laurent, Sean Lee, Simon Lefrancois, Simon Lemieux, Nicholas Léonard, Zhouhan Lin, Jesse A. Livezey, Cory Lorenz, Jeremiah Lowin, Qianli Ma, Pierre-Antoine Manzagol, Olivier Mastropietro, Robert T. McGibbon, Roland Memisevic, Bart van Merriënboer, Vincent Michalski, Mehdi Mirza, Alberto Orlandi, Christopher Pal, Razvan Pascanu, Mohammad Pezeshki, Colin Raffel, Daniel Renshaw, Matthew Rocklin, Adriana Romero, Markus Roth, Peter Sadowski, John Salvatier, François Savard, Jan Schlüter, John Schulman, Gabriel Schwartz, Iulian Vlad Serban, Dmitriy Serdyuk, Samira Shabanian, Étienne Simon, Sigurd Spieckermann, S. Ramana Subramanyam, Jakub Sygnowski, Jérémie Tanguay, Gijs van Tulder, Joseph Turian, Sebastian Urban, Pascal Vincent, Francesco Visin, Harm de Vries, David Warde-Farley, Dustin J. Webb, Matthew Willson, Kelvin Xu, Lijun Xue, Li Yao, Saizheng Zhang, and Ying Zhang. Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688, May 2016.

[69] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, and Xiaoqiang Zheng. TensorFlow: Large-

scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[70] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: a system for large-scale machine learning. In *OSDI*, volume 16, pages 265–283, 2016.

[71] Jasmine Banks and Peter Corke. Quantitative evaluation of matching methods and validity measures for stereo vision. *The International Journal of Robotics Research*, 20(7):512–532, 2001.

[72] Geoffrey J McLachlan and Kaye E Basford. *Mixture models: Inference and applications to clustering*, volume 84. M. Dekker New York, 1988.

[73] Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1):1–22, 1977.

[74] Geoffrey J McLachlan and Thriyambakam Krishnan. Wiley series in probability and statistics. the em algorithm and extensions, 1997.

[75] Christopher M Bishop. *Pattern recognition and machine learning*. springer, 2006.

[76] Judea Pearl. *Reverend Bayes on inference engines: A distributed hierarchical approach*. Cognitive Systems Laboratory, School of Engineering and Applied Science . . . , 1982.

[77] Brendan J Frey, J Frey Brendan, and Brendan J Frey. *Graphical models for machine learning and digital communication*. MIT press, 1998.

[78] Leonid Iakov Rudin. Images, numerical analysis of singularities and shock filters. 1987.

[79] Stanley Osher and Leonid I Rudin. Feature-oriented image enhancement using shock filters. *SIAM Journal on numerical analysis*, 27(4):919–940, 1990.

[80] Leonid I Rudin, Stanley Osher, and Emad Fatemi. Nonlinear total variation based noise removal algorithms. *Physica D: nonlinear phenomena*, 60(1-4):259–268, 1992.

[81] Tony F Chan and Selim Esedoglu. Aspects of total variation regularized l 1 function approximation. *SIAM Journal on Applied Mathematics*, 65(5):1817–1837, 2005.

[82] Javier Sánchez Pérez, Enric Meinhardt-Llopis, and Gabriele Facciolo. Tv-l1 optical flow estimation. *Image Processing On Line*, 2013:137–150, 2013.

[83] Thomas Brox, Andrés Bruhn, Nils Papenberg, and Joachim Weickert. High accuracy optical flow estimation based on a theory for warping. In *European conference on computer vision*, pages 25–36. Springer, 2004.

[84] Pedro F Felzenszwalb and Daniel P Huttenlocher. Efficient belief propagation for early vision. *International journal of computer vision*, 70(1):41–54, 2006.

[85] Andreas Wedel, Thomas Brox, Tobi Vaudrey, Clemens Rabe, Uwe Franke, and Daniel Cremers. Stereoscopic scene flow computation for 3d motion understanding. *International Journal of Computer Vision*, 95(1):29–51, 2011.

[86] W Eric L Grimson, Chris Stauffer, Raquel Romano, and Lily Lee. Using adaptive tracking to classify and monitor activities in a site. In *Proceedings. 1998 IEEE*

Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No. 98CB36231), pages 22–29. IEEE, 1998.

[87] Ahmed Elgammal, David Harwood, and Larry Davis. Non-parametric model for background subtraction. In *European conference on computer vision*, pages 751–767. Springer, 2000.

[88] Pakorn KaewTraKulPong and Richard Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In *Video-based surveillance systems*, pages 135–144. Springer, 2002.

[89] Dar-Shyang Lee. Effective gaussian mixture learning for video background subtraction. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (5):827–832, 2005.

[90] Zoran Zivkovic and Ferdinand Van Der Heijden. Efficient adaptive density estimation per image pixel for the task of background subtraction. *Pattern recognition letters*, 27(7):773–780, 2006.

[91] Namrata Vaswani, Thierry Bouwmans, Sajid Javed, and Praneeth Narayanamurthy. Robust subspace learning: Robust pca, robust subspace tracking, and robust subspace recovery. *IEEE signal processing magazine*, 35(4):32–55, 2018.

[92] Hongwei Yong, Deyu Meng, Wangmeng Zuo, and Lei Zhang. Robust online matrix factorization for dynamic background subtraction. *IEEE transactions on pattern analysis and machine intelligence*, 40(7):1726–1740, 2017.

[93] D Michael Titterington. Recursive parameter estimation using incomplete data.

Journal of the Royal Statistical Society: Series B (Methodological), 46(2):257–267, 1984.

[94] Jian Sun, Nan-Ning Zheng, and Heung-Yeung Shum. Stereo matching using belief propagation. *IEEE Transactions on Pattern Analysis & Machine Intelligence*, (7):787–800, 2003.

[95] Antoni Buades, Bartomeu Coll, and Jean-Michel Morel. Denoising image sequences does not require motion estimation. In *IEEE Conference on Advanced Video and Signal Based Surveillance, 2005.*, pages 70–74. IEEE, 2005.

[96] Antoni Buades, Bartomeu Coll, and J-M Morel. A non-local algorithm for image denoising. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 2, pages 60–65. IEEE, 2005.

[97] Antonin Chambolle, Vicent Caselles, Daniel Cremers, Matteo Novaga, and Thomas Pock. An introduction to total variation for image analysis. *Theoretical foundations and numerical methods for sparse recovery*, 9(263-340):227, 2010.

[98] Tobi Vaudrey, Andreas Wedel, Chia-Yen Chen, and Reinhard Klette. Improving optical flow using residual and sobel edge images. In *International Conference on Arts and Technology*, pages 215–222. Springer, 2009.

[99] Helmut Grabner, Michael Grabner, and Horst Bischof. Real-time tracking via online boosting. In *Bmvc*, volume 1, page 6, 2006.

[100] Alan Lukežič, Tomáš Vojíř, Luka Čehovin Zajc, Jiří Matas, and Matej Kristan. Discriminative correlation filter tracker with channel and spatial reliability. *International Journal of Computer Vision*, 7(126):671–688, 2018.

[101] David Held, Sebastian Thrun, and Silvio Savarese. Learning to track at 100 fps with deep regression networks. In *European Conference on Computer Vision*, pages 749–765. Springer, 2016.

[102] Martin Danelljan, Fahad Shahbaz Khan, Michael Felsberg, and Joost Van de Weijer. Adaptive color attributes for real-time visual tracking. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1090–1097, 2014.

[103] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Forward-backward error: Automatic detection of tracking failures. In *2010 20th International Conference on Pattern Recognition*, pages 2756–2759. IEEE, 2010.

[104] Boris Babenko, Ming-Hsuan Yang, and Serge Belongie. Visual tracking with online multiple instance learning. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 983–990. IEEE, 2009.

[105] David S Bolme, J Ross Beveridge, Bruce A Draper, and Yui Man Lui. Visual object tracking using adaptive correlation filters. In *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, pages 2544–2550. IEEE, 2010.

[106] Zdenek Kalal, Krystian Mikolajczyk, and Jiri Matas. Tracking-learning-detection. *IEEE transactions on pattern analysis and machine intelligence*, 34(7):1409–1422, 2011.

[107] Ilya Ardakani, Koichi Hashimoto, and Ken Yoda. Understanding animal behavior using their trajectories. In *International Conference on Distributed, Ambient, and Pervasive Interactions*, pages 3–22. Springer, 2018.

[108] Quannan Li, Yu Zheng, Xing Xie, Yukun Chen, Wenyu Liu, and Wei-Ying Ma.

Mining user similarity based on location history. In *Proceedings of the 16th ACM SIGSPATIAL international conference on Advances in geographic information systems*, page 34. ACM, 2008.

[109] Yu Zheng, Lizhu Zhang, Xing Xie, and Wei-Ying Ma. Mining interesting locations and travel sequences from gps trajectories. In *Proceedings of the 18th international conference on World wide web*, pages 791–800. ACM, 2009.

[110] Jing Yuan, Yu Zheng, Liuhang Zhang, XIng Xie, and Guangzhong Sun. Where to find my next passenger. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 109–118. ACM, 2011.

[111] David Sculley. Web-scale k-means clustering. In *Proceedings of the 19th international conference on World wide web*, pages 1177–1178. ACM, 2010.

[112] Scott Gaffney and Padhraic Smyth. Trajectory clustering with mixtures of regression models. 1999.

[113] Martin Ester, Hans-Peter Kriegel, Jörg Sander, Xiaowei Xu, et al. A density-based algorithm for discovering clusters in large spatial databases with noise. In *Kdd*, volume 96, pages 226–231, 1996.

[114] Mihael Ankerst, Markus M Breunig, Hans-Peter Kriegel, and Jörg Sander. Optics: ordering points to identify the clustering structure. In *ACM Sigmod record*, volume 28, pages 49–60. ACM, 1999.

[115] Mirco Nanni and Dino Pedreschi. Time-focused clustering of trajectories of moving objects. *Journal of Intelligent Information Systems*, 27(3):267–289, 2006.

[116] Derya Birant and Alp Kut. St-dbscan: An algorithm for clustering spatial–temporal data. *Data & Knowledge Engineering*, 60(1):208–221, 2007.

[117] Tian Zhang, Raghu Ramakrishnan, and Miron Livny. Birch: an efficient data clustering method for very large databases. In *ACM Sigmod Record*, volume 25, pages 103–114. ACM, 1996.

[118] Ulrike Von Luxburg. A tutorial on spectral clustering. *Statistics and computing*, 17(4):395–416, 2007.

[119] David J Aldous. Exchangeability and related topics. In *École d'Été de Probabilités de Saint-Flour XIII—1983*, pages 1–198. Springer, 1985.

[120] Nadia FF Da Silva, Eduardo R Hruschka, and Estevam R Hruschka Jr. Tweet sentiment analysis with classifier ensembles. *Decision Support Systems*, 66:170–179, 2014.

[121] Alec Go, Lei Huang, and Richa Bhayani. Twitter sentiment analysis. *Entropy*, 17:252, 2009.

[122] Ho Chung Wu, Robert Wing Pong Luk, Kam Fai Wong, and Kui Lam Kwok. Interpreting tf-idf term weights as making relevance decisions. *ACM Transactions on Information Systems (TOIS)*, 26(3):13, 2008.

[123] Scikit-learn manual. section 4.2 – feature extraction. `http://scikit-learn.org/stable/modules/feature_extraction.html#text-feature-extraction`. Last Accessed: 2019-06-30.

[124] David Blei, Lawrence Carin, and David Dunson. Probabilistic topic models: A focus

on graphical model design and applications to document and image analysis. *IEEE signal processing magazine*, 27(6):55, 2010.

[125] David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *Journal of machine Learning research*, 3(Jan):993–1022, 2003.

[126] Matthew Hoffman, Francis R Bach, and David M Blei. Online learning for latent dirichlet allocation. In *advances in neural information processing systems*, pages 856–864, 2010.

[127] Alexander Pak and Patrick Paroubek. Twitter as a corpus for sentiment analysis and opinion mining. In *LREc*, volume 10, pages 1320–1326, 2010.

[128] Harry Zhang. The optimality of naive bayes. *AA*, 1(2):3, 2004.

[129] Bianca Zadrozny and Charles Elkan. Obtaining calibrated probability estimates from decision trees and naive bayesian classifiers. In *Icml*, volume 1, pages 609–616. Citeseer, 2001.

[130] Bianca Zadrozny and Charles Elkan. Transforming classifier scores into accurate multiclass probability estimates. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 694–699. ACM, 2002.

[131] John Platt et al. Probabilistic outputs for support vector machines and comparisons to regularized likelihood methods. *Advances in large margin classifiers*, 10(3):61–74, 1999.

[132] Alexandru Niculescu-Mizil and Rich Caruana. Predicting good probabilities with

supervised learning. In *Proceedings of the 22nd international conference on Machine learning*, pages 625–632. ACM, 2005.

[133] Brian W Matthews. Comparison of the predicted and observed secondary structure of t4 phage lysozyme. *Biochimica et Biophysica Acta (BBA)-Protein Structure*, 405(2):442–451, 1975.

[134] Sabri Boughorbel, Fethi Jarray, and Mohammed El-Anbari. Optimal classifier for imbalanced data using matthews correlation coefficient metric. *PloS one*, 12(6):e0177678, 2017.

[135] Yoda lab – ethology & ecology. `http://yoda-ken.sakura.ne.jp/yoda_lab/English.html`. Last Accessed: 2019-06-30.

[136] Hiroshi ARIMA and Hisashi SUGAWA. Correlation between the pitch of calls and external measurements of streaked shearwaters calonectris leucomelas breeding on kanmuri island. *Japanese Journal of Ornithology*, 53(1):40–44, 2004.

[137] Graeme D Ruxton. The unequal variance t-test is an underused alternative to student's t-test and the mann–whitney u test. *Behavioral Ecology*, 17(4):688–690, 2006.

[138] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[139] Ilya Ardakani, Koichi Hashimoto, and Ken Yoda. Context-based semantical vector representations for animal trajectories. *Advanced Robotics*, 33:1–16, 02 2019.

[140] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

[141] Andriy Mnih and Yee Whye Teh. A fast and simple algorithm for training neural probabilistic language models. *arXiv preprint arXiv:1206.6426*, 2012.

[142] Scikit-learn – comparing different clustering algorithms. `https://scikit-learn.org/stable/auto_examples/cluster/plot_cluster_comparison.html`. Last Accessed: 2018-12-30.

[143] Codalab – animal behaviour challenge abc 2018. `https://competitions.codalab.org/competitions/16283`. Last Accessed: 2018-12-30.

[144] Svante Wold, Kim Esbensen, and Paul Geladi. Principal component analysis. *Chemometrics and intelligent laboratory systems*, 2(1-3):37–52, 1987.

[145] Laurens van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of machine learning research*, 9(Nov):2579–2605, 2008.

[146] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[147] Alex Graves and Navdeep Jaitly. Towards end-to-end speech recognition with recurrent neural networks. In *International conference on machine learning*, pages 1764–1772, 2014.

[148] Kenneth O Stanley, Bobby D Bryant, and Risto Miikkulainen. Evolving neural network agents in the nero video game. *Proceedings of the IEEE*, pages 182–189, 2005.

[149] Oriol Vinyals, Łukasz Kaiser, Terry Koo, Slav Petrov, Ilya Sutskever, and Geoffrey Hinton. Grammar as a foreign language. In *Advances in neural information processing systems*, pages 2773–2781, 2015.

[150] Muhua Zhu, Yue Zhang, Wenliang Chen, Min Zhang, and Jingbo Zhu. Fast and accurate shift-reduce constituent parsing. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 434–443, 2013.

[151] Nitish Srivastava, Elman Mansimov, and Ruslan Salakhudinov. Unsupervised learning of video representations using lstms. In *International conference on machine learning*, pages 843–852, 2015.

[152] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112, 2014.

[153] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[154] Ilya S Ardakani and Koichi Hashimoto. Encoding bird's trajectory using recurrent neural networks. In *Mechatronics and Automation (ICMA), 2017 IEEE International Conference on*, pages 1644–1649. IEEE, 2017.

[155] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm networks. In *Proceedings. 2005 IEEE International Joint Conference on Neural Networks, 2005.*, volume 4, pages 2047–2052. IEEE, 2005.

[156] Christopher M Bishop. Mixture density networks. Technical report, Citeseer, 1994.

[157] Chris M Bishop and Claire Legleye. Estimating conditional probability densities for

periodic variables. In *Advances in Neural Information Processing Systems*, pages 641–648, 1995.

[158] Salah Rifai, Pascal Vincent, Xavier Muller, Xavier Glorot, and Yoshua Bengio. Contractive auto-encoders: Explicit invariance during feature extraction. In *Proceedings of the 28th International Conference on International Conference on Machine Learning*, pages 833–840. Omnipress, 2011.

[159] Honglak Lee, Chaitanya Ekanadham, and Andrew Y Ng. Sparse deep belief net model for visual area v2. In *Advances in neural information processing systems*, pages 873–880, 2008.

[160] Koray Kavukcuoglu, Rob Fergus, Yann LeCun, et al. Learning invariant features through topographic filter maps. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1605–1612. IEEE, 2009.

[161] Andrew Ng et al. Sparse autoencoder. *CS294A Lecture notes*, 72(2011):1–19, 2011.

[162] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning*. MIT press, 2016.

[163] Pascal Vincent, Hugo Larochelle, Yoshua Bengio, and Pierre-Antoine Manzagol. Extracting and composing robust features with denoising autoencoders. In *Proceedings of the 25th international conference on Machine learning*, pages 1096–1103. ACM, 2008.

[164] Yoshua Bengio, Li Yao, Guillaume Alain, and Pascal Vincent. Generalized denoising auto-encoders as generative models. In *Advances in Neural Information Processing Systems*, pages 899–907, 2013.

[165] Pascal Vincent. A connection between score matching and denoising autoencoders. *Neural computation*, 23(7):1661–1674, 2011.

[166] Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[167] Otto Fabius and Joost R van Amersfoort. Variational recurrent auto-encoders. *arXiv preprint arXiv:1412.6581*, 2014.

[168] Michael Mathieu, Camille Couprie, and Yann LeCun. Deep multi-scale video prediction beyond mean square error. *arXiv preprint arXiv:1511.05440*, 2015.

[169] Magdalena Klapper-Rybicka, Nicol N Schraudolph, and Jürgen Schmidhuber. Unsupervised learning in lstm recurrent neural networks. In *International Conference on Artificial Neural Networks*, pages 684–691. Springer, 2001.

[170] Alex Graves. Generating sequences with recurrent neural networks. *arXiv preprint arXiv:1308.0850*, 2013.

[171] Bernhard Hofmann-Wellenhof, Herbert Lichtenegger, and James Collins. *Global positioning system: theory and practice*. Springer Science & Business Media, 2001.

[172] Samuel Picton Drake. Converting gps coordinates [phi, lambda, h] to navigation coordinates (enu). 2002.

[173] Geoffrey Hinton, Nitish Srivastava, and Kevin Swersky. Neural networks for machine learning lecture 6a overview of mini-batch gradient descent.

[174] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[175] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

[176] Ilya S. ARDAKANI and Koichi HASHIMOTO. Marine birds' behaviour visualization. *The Proceedings of JSME annual Conference on Robotics and Mechatronics (Robomec)*, 2017:2A2–Q03, 11 2017.

[177] Erddap – national oceanic and atmospheric administration (noaa), easier access to scientific data. `https://coastwatch.pfeg.noaa.gov/erddap/index.html`. Last Accessed: 2019-06-30.

[178] Opendap – advanced software for remote data retrieval. `https://www.opendap.org/`. Last Accessed: 2019-06-30.

[179] Netcdf – network common data form. `https://www.unidata.ucar.edu/software/netcdf/`. Last Accessed: 2019-06-30.

[180] Sarah E Battersby, Michael P Finn, E Lynn Usery, and Kristina H Yamamoto. Implications of web mercator and its use in online mapping. *Cartographica: The International Journal for Geographic Information and Geovisualization*, 49(2):85–101, 2014.

List of Publications

Journals

1. Ardakani, Ilya and Hashimoto, Koichi and Yoda, Ken, “Context-based semantical vector representations for animal trajectories”, *Advanced Robotics*, 2019, 33.3-4: 118-133.

Proceedings of International Conferences

2. Ardakani, Ilya and Hashimoto, Koichi and Yoda, Ken, “Understanding animal behavior using their trajectories”, *International Conference on Distributed, Ambient, and Pervasive Interactions*. Springer, Cham, 2018. p. 3-22.

3. Ardakani, Ilya S and Hashimoto, Koichi, “Encoding bird’s trajectory using Recurrent Neural Networks”, *2017 IEEE International Conference on Mechatronics and Automation (ICMA)*. IEEE, 2017. p. 1644-1649.

Acknowledgments

First of all, I would like to express my sincere appreciation to my advisor, Prof. Koichi Hashimoto, for his kind and endless support for me, and of course, his patience with my long presentations. With no exception, his critical advice and recommendations helped me to orient and advance my research to this day. I should also acknowledge that he, by introducing me to Bio-Navigation field, fundamentally changed my view towards the utilization of the natural environment and helped me to recognize the importance of environment conservation.

I would like to appreciate Prof. Ken Yoda for providing me advice, perspective and support in movement ecology. Specifically, I would like to appreciate him and his team for sharing the results of their hard and precious work, fuel of my research, trajectory data of marine birds. I would like to thank Prof. Shizuko Hiryu for supporting me and allowing me the opportunity of imaging bats in flight chamber. In addition, I would like to recognize the support of her students and lab members, Emyo Fujioka, Kazuma Hase and Saori Sugiwara.

Furthermore, I would like to thank Shingo Kagami and Akihiko Yamaguchi for their advice and constructive comments, my colleagues and fellow students for their assistance, comments and collaborations through these years. I would like to thank my family and particularly my wife for the encouragement and support. Lastly, I am truly grateful to Japan's Ministry of Education, Culture, Sports, Science and Technology (MEXT) for providing me financial support during my doctoral studies. In addition, I would like to acknowledge that this work was also supported by JSPS Kakenhi Grant numbers 16H06536 and 16H06541I.