Computational movement analysis using Brownian bridges

Sijben, S.M.A.

Award date:
2013
Computational Movement Analysis
Using Brownian Bridges

Master's Thesis

Stef Sijben
Abstract

Widespread availability of location tracking devices leads to a rapidly increasing amount of movement data and to the need to analyse these data. These data typically consist of location measurements taken at discrete times, often with a non-negligible error margin in the measured locations. A low sampling rate leads to high uncertainty of the locations between the times at which these measurements were obtained. So far this uncertainty is usually ignored in the computational analysis of movement data. In this thesis we investigate how to incorporate this uncertainty into the analysis by the use of movement models that model the movement as a random process. Specifically we consider the Brownian bridge movement model, which assumes that an entity performs Brownian motion that is conditioned on the measured locations. In previous work this model was used only to derive a probability distribution of space use or to visualize movement data. In this thesis we contribute to the computational analysis of movement data in the following ways.

Firstly, we provide algorithmic tools for analysing movement data using the Brownian bridge movement model. We develop a framework for detecting movement patterns from movement data in this model. Many patterns are composed from basic building blocks like position, distance, speed and direction. We derive their distributions in the Brownian bridge movement model. Using these building blocks, we show how to detect the patterns encounter, regular visit and following.

Secondly, we combine our algorithmic tools with the movement ecology paradigm by Nathan et al., which models influences on the movement path from the environment and the internal state of an entity. We demonstrate how these influences can be integrated into our algorithmic framework by the example of two studies on the speed of animals. In one study, we integrate external factors a priori through the parameters of the movement model. In the other we use the external variables in the analysis after obtaining the results from the model.

Thirdly, we show how to refine the Brownian bridge movement model using information on the behavioural state of the entity being studied. We demonstrate a new method to estimate the parameter of the model. In order to obtain this behaviour information from accelerometer data, we discuss approaches to augment existing classifiers for data that have dependencies between consecutive observations.

Finally, we present a tool with easy to use implementations of many of the methods discussed in this thesis. We provide this tool as a package for the R environment for statistical computing, which is commonly used in movement ecology. Many of the tasks allow efficient parallelization using graphics processors and therefore we have implemented some functions using OpenCL, a standard for parallel programming on various hardware platforms.
I could not have completed this thesis without the contributions of many people. First of all, I am grateful to my supervisor, Kevin Buchin, whose ideas and guidance allowed me to complete this work. Thank you for your time, the productive discussions and the helpful comments on this thesis. I also thank Eduard Belitser and Michel Westenberg for serving on the assessment committee.

I thank Erik Willems, Jean Arseneau, Nir Sapir and Stéphanie Mercier for their insights in behavioural ecology, and Emiel van Loon who contributed his expertise on the statistical analysis of movement data. I am grateful to Huub van de Wetering and Roeland Scheepens for their advice on visualizing movement data.

I thank Emiel for posing research questions on the use of behaviour data in movement analysis. I am grateful to Eduard Belitser for a helpful discussion on incorporating time into classification methods, and to Maike Buchin for ideas on segmentation by behavioural states and how to combine these with classification.

I am thankful to Kamran Safi for the invitation for a short research visit at the Max Planck Institute for Ornithology in Radolfszell. I thank Erik, Emiel, Kamran and Bart Kranstauber for the many inspiring discussions during our meetings. I thank Erik and his students, who have tested the R package, despite all the bugs in it, and suggested many improvements.

I thank everyone in the Algorithms group, without whom the past months would not have been as interesting and fun.

Finally, I thank my friends and family, who always supported me during my studies.
4 Bridges Integrating Behavioural States

4.1 Time-dependent behaviour classification 35
  4.1.1 The Bayesian classifier 36
  4.1.2 Segmentation based on the Bayesian information criterion 37
  4.1.3 Classification using hidden Markov models 38

4.2 Incorporating behaviour data in the BBMM 39

4.3 Estimating the diffusion coefficient 40

4.4 Oystercatcher case study 42
  4.4.1 Results of the classification algorithms 42
  4.4.2 Trajectories with behaviour information 44

5 R Package movementAnalysis

5.1 Class bbtraj 47

5.2 Functions 48
  5.2.1 Basic properties 48
  5.2.2 Encounter 48
  5.2.3 Spatial distributions 48
  5.2.4 Plotting functions 48

5.3 Implementation details 49
  5.3.1 Interfacing C and C++ code 49
  5.3.2 Using OpenCL from R 50

6 Conclusion 53

Bibliography 55
Widespread use of location-aware devices results in an increased demand for the analysis of movement data. While a trajectory, that is, the movement path over time, is continuous, trajectory data consists of location measurements at discrete points in time, as shown in Figure 1.1. Most of the existing methods have been developed for the case of very densely sampled data for which one can assume that movement between sample points is linear, or for which it is not even necessary to interpolate between sample points. These are viable models when the sampling rate is high, but their validity quickly decreases with lower sampling rates. This is demonstrated in Figure 1.2a. Assume that we want to detect whether the trajectory moves through the area $\mathcal{A}$. If the trajectory was sampled at a low rate and the position is interpolated linearly, the model might show that $\mathcal{A}$ is not traversed, while the trajectory does move through $\mathcal{A}$.

In movement ecology, constraints on the size and weight of location-tracking devices and the required battery life often lead to the use of sampling rates that are lower than those for which linear interpolation makes sense. Instead movement models based on random processes are used. The Brownian bridge movement model (BBMM) is a commonly used movement model assuming Brownian motion conditioned under the locations at the sample points. In this model, we would solve the problem with the linear interpolation by providing a probability that the trajectory would pass through $\mathcal{A}$, as some realizations of the Brownian motion traverse $\mathcal{A}$ while others do not. This is illustrated in Figure 1.2b. This probability can be approximated using Monte Carlo methods, i.e. simulating a

![Figure 1.1: Example of a trajectory and movement data. (a) The continuous trajectory that was actually followed. (b) Movement data with a high sampling rate, with measurement errors.](image-url)
Chapter 1. Introduction

Figure 1.2: The problem with linear interpolation when the sampling rate is low. (a) The trajectory traverses $A$, but the interpolated data does not. (b) Two realizations in the BBMM, one of which traverses $A$.

large number of realizations of the trajectory to estimate the value, but often we can also reason about the probability distributions to obtain such estimates. From a computational point of view, the latter method is usually preferable, since performing many simulations uses more resources and thus the user has to wait longer for the results. Section 1.3 provides more information about the BBMM. This model has so far been used to compute estimates for the space use of an animal [11, 24] and to visualize trajectory data [15, 36].

This analysis of trajectory data often focuses on the detection of movement patterns like encounter, following or regular visit. Figure 1.3 shows a taxonomy of movement patterns. For many patterns, efficient computational methods for detecting them have been developed [21]. These methods do not interpolate between the sampled locations or use the linear movement model for interpolation. No methods exist for reliably detecting these patterns in the presence of low sampling rates or uncertain locations.

Nathan et al. [33] proposed a paradigm, which incorporates four basic components that describe the factors influencing a movement path: external factors, the internal state of the moving organism, its navigational capacity and its motion capacity (Figure 1.4). An interesting algorithmic question is how to integrate these influences in the presence of uncertain data.

Next to environmental data, acceleration data that is obtained concurrently with location data is increasingly becoming available. Classification methods can be used to derive information about the animal’s behaviour from this data. Classification deals with the problem of identifying which of a set of categories (e.g. behaviours) applies to a data point, based on a training data set for which the correct categories of the data points are known. This training data set can be obtained by observing the animal and labelling its behaviour for part of the period over which the data points were obtained.

Segmentation is the problem of subdividing a trajectory into subtrajectories which are homogeneous in a certain sense [9], for example by requiring that the behaviour in each segment does not change. Ideas from segmentation can be used to improve the predictions made by classification algorithms.

1.1 Results and organization

In Chapter 2 we discuss how movement patterns can be detected when using the BBMM. These movement patterns can often be formulated in terms of basic properties like position, distance, speed
1.1. Results and organization

Figure 1.3: Taxonomy of movement patterns [17].

Figure 1.4: The movement ecology paradigm by Nathan et al. [33].
and direction. Therefore we derive probability distributions for these properties and then show how some example patterns can be formulated in the context of the BBMM.

In Chapter 3 we investigate how influences from the environment can be incorporated into movement analysis when using the BBMM. We discuss how the speed of animals is influenced by environmental variables in two settings. We also investigate how the detection of encounters is influenced by visibility in a terrain.

In Chapter 4 we discuss methods for classification of behaviours from acceleration data and how ideas from segmentation can be used to improve the quality of these predictions. We also describe how the BBMM can be adjusted to take advantage of this behavioural data.

From the application perspective it is crucial that these methods are also accessible in a widely used analysis framework. In the context of movement ecology R [35], an environment for statistical computing and modelling, is commonly used. To this end, we provide a frontend to access our framework from within R, which is described in Chapter 5.

1.2 Related work

Movement ecology studies the movement of micro-organisms, plants and animals. It aims at understanding the causes, mechanisms, patterns, and consequences of movement [33]. In movement ecology random movement models are commonly used to model movements [25, 14], in particular the BBMM is used to interpolate between sample points [11, 24]. Several variants of the BBMM have been considered [4, 27]. Our framework can also be used with these variants. If the sampling is extremely sparse (e.g., only one sample per day) then interpolating between sample points is no longer reasonable and kernel density estimation is frequently used [2, 43]. This then, however, completely ignores motion between sample points, does not account for temporal autocorrelation, and cannot cope with high data-volumes very well (algorithms used to calculate the kernel’s bandwidth fail to minimise). Kernel density estimation can be combined with methods from time geography to counter these shortcomings [18]. Kernel density estimation and random movement models are also used for visualizing movement data [15, 36]. In this thesis we focus on data analysis, but would like to note that such an analysis is most powerful when combined with such visualizations.

The movement pattern encounter was already considered in one of the first papers to study movement patterns in trajectory data [29]. It is basically defined as a set of trajectories that are in the same disk at the same time. For the patterns attraction and avoidance we are not aware of similar prior work, while these patterns have been proposed in the setting of reactive movement [30]. There are several definitions of following patterns, a direction-based [3] and a path-based definition [6]. In the first definition one entity is following another, if both entities move in a similar direction, and the leading entity is in a cone in front of the following one. In the second definition one entity is following the other if it visits approximately the same locations but with a small, variable time shift. Algorithms for detecting regular visits [16] find reoccurring visits of an entity to a certain region. Besides these movement patterns, many more have been considered. A taxonomy of movement patterns is given in [17]. All of the patterns above are also described (in the linear movement model) in a recent survey on algorithmic movement analysis [22].

Classification algorithms have been used used to identify animal behaviour from acceleration data [41, 34, 37]. Typically, these approaches use a decision-tree classifier. Segmentation algorithms exist to subdivide a trajectory based on location and time information [9] and by behavioural states [10]. Trajectory data has also been used to dynamically vary the parameters of the BBMM over time [27]. We are not aware of previous work using classification or segmentation based on behavioural states.
1.3. The Brownian bridge movement model

Throughout this thesis we use the Brownian bridge movement model, and denote it by BBMM. In this section we therefore recall its fundamental properties. The BBMM assumes that an entity exhibits Brownian motion between measured locations. Examples of Brownian motion are shown in Figure 1.5.

Standard Brownian motion is defined as a continuous-time random process $B_t$ that satisfies the following conditions:

1. $B_0 = 0$.
2. The process has independent increments, that is for times $t_1 \leq t_2 \leq t_3 \leq t_4$, the increments $B_{t_2} - B_{t_1}$ and $B_{t_4} - B_{t_3}$ are independent random variables.
3. For all $t \geq 0$ and $h > 0$, the increment $B_{t+h} - B_t$ is normally distributed with mean 0 and variance $h$.
4. The function $t \mapsto B_t$ is almost surely everywhere continuous.

In multiple dimensions, every component of $B_t$ performs independent standard Brownian motion. This implies that the distribution of the increment $B_{t+h} - B_t \sim \mathcal{N}(0, hI)$ is a circular normal distribution. Here, 0 denotes the null vector and $I$ is the identity matrix in the proper dimension. In the remainder of this document we shall use $\mathcal{N}(\mu, \sigma^2)$ to denote a circular normal distribution with mean $\mu$ and covariance matrix $\sigma^2 I$.

Figure 1.5: Example realizations of standard Brownian motion. (a) One-dimensional Brownian motion plotted over time. (b) Two-dimensional Brownian motion.

Brownian bridges are used for tasks like estimating animal home ranges and migration routes, and for evaluating the influence of resource selection on movement [24]. Methods for home range estimation based on the BBMM are provided in the R packages BBMM\textsuperscript{1}, adehabitatHR\textsuperscript{2} [12] and Move\textsuperscript{3}.

\textsuperscript{1}http://cran.r-project.org/web/packages/BBMM/
\textsuperscript{2}http://cran.r-project.org/web/packages/adehabitatHR/
\textsuperscript{3}http://computational-ecology.com/main-move.html
A Brownian motion can be characterized by a starting location $x$ and a scale parameter $\sigma_m^2$, which is called the diffusion coefficient. That is, if $X_t$ is a Brownian motion with parameters $x$ and $\sigma_m^2$, then $X_t = x + \sigma_m B_t$, where $B_t$ is a standard Brownian motion. This implies that $X_t \sim \mathcal{N}(x, t\sigma_m^2)$.

In the BBMM we have multiple location measurements and want to reason about the location at times between the times of these measurements. Therefore, we condition a Brownian motion on the location $X_T$ at some time $T$, as well as the starting location $X_0$. Such a conditioned Brownian motion is called a Brownian bridge. Figure 1.6 shows examples of Brownian bridges.

Brownian motion has the Markov property: If we know the state $X_t$ at a certain point in time, the state $X_{t'}$ for $t' > t$ depends only on $X_t$ and not on the state of the process before that. Using this property, one can show that for $0 \leq t \leq T$,

$$\left( X_t | X_0 = a \land X_T = b \right) \sim \mathcal{N}\left((1 - \alpha)a + \alpha b, T\alpha(1 - \alpha)\sigma_m^2\right),$$

where $\alpha := \frac{t}{T}$ [11]. This and other results in this thesis hold for arbitrary dimension, unless stated otherwise.

To model uncertainty in the measured locations and to avoid degenerate probability distributions at the time of a measurement, the locations are often assumed to be independently normally distributed around the measured locations. If we assume that we have two locations $x_{(i)}$, $x_{(i+1)}$ measured at times $t_i$, $t_{i+1}$ with variances $\delta_i^2$ and $\delta_{i+1}^2$ respectively, the position at a time $t \in [t_i, t_{i+1}]$ follows a circular normal distribution with parameters

$$\mu_i(t) = \alpha x_i + (1 - \alpha)x_{i+1},$$

$$\sigma_i^2(t) = (1 - \alpha)^2\delta_i^2 + \alpha^2\delta_{i+1}^2 + (t_{i+1} - t_i)\alpha(1 - \alpha)\sigma_m^2,$$

where $\alpha := \frac{t - t_i}{t_{i+1} - t_i}$.
1.3. The Brownian bridge movement model

where $\alpha := \frac{t_{i+1} - t_i}{t_{i+1} - t_i}$.

The diffusion coefficient is often estimated by a maximum likelihood method [24]. This method is discussed in more detail in Section 4.3. When the trajectory contains different movement states over time, a better estimate of the diffusion coefficient can be obtained by letting it vary with time [27].

Given a number of measured locations with time stamps and estimates for the variance, one can compute a utilization distribution, i.e. a probability distribution that describes the fraction of time the animal spends in various regions. Examples are shown in Figure 1.7. Let $x_i$, $t_i$ and $\sigma_i^2$ ($0 \leq i \leq n$) denote the measured position, time and variance of the $i$th observation. Then the density of the utilization distribution is given by

$$h(x) = \frac{1}{t_n - t_0} \int_{t_0}^{t_n} \phi \left( x; \mu_i(t), \sigma_i^2(t) \right) dt,$$

with $\mu_i(t)$ and $\sigma_i^2(t)$ as defined in Equation 1.2. $\phi \left( x; \mu, \sigma^2 \right)$ denotes the probability density of a circular normal distribution with parameters $\mu$ and $\sigma^2$, evaluated at $x$. Ecologists are often interested not in the utilization distribution as such, but in the home range of an animal. The home range is often estimated as the region within a contour such that a certain fraction of the probability density is contained in that contour, as shown in Figure 1.7b. While we cannot directly compute the probability that the animal visits the region $A$ since the locations at different times are not independent, we can straightforwardly compute the expected time the animal spends in this region from the utilization distribution.

Figure 1.7: Examples of utilization distributions. (a) Probability density for the fraction of time spent in different regions between two sample points which correspond two the two peaks [24]. (b) Utilization distribution (density indicated by shading) and 99% contour computed from sparsely sampled trajectory data.
In the past solutions for detecting a variety of movement patterns have been given, such as the patterns encounter, following, and regular visits. These solutions typically consist of a formal definition of the pattern and an efficient algorithm to detect it in trajectory data for the linear movement model. Here we formulate movement patterns in a meaningful way in a probabilistic setting. For this we provide in Section 2.1 basic building blocks to deal with problems concerning quantities like position, distance, speed and direction.

We show how these building blocks can be combined to describe movement patterns in Section 2.2. In particular, we show how to formulate and compute the following patterns under the Brownian bridge movement model (BBMM): encounter, avoidance and attraction, following, and regular visits. We have selected these patterns to demonstrate the use of the various basic building blocks, and because these are commonly needed patterns when analyzing interaction.

Computing movement patterns using Brownian bridges is computationally expensive, and we tackle the computational task by parallel computations as described in Section 2.3. Nowadays, graphics hardware (GPU) offer a widespread, relatively low-cost possibility for parallel computations. We show how to formulate our tasks such that the computations can be handled efficiently in this way.

We have evaluated our algorithms in a case study of encounters between groups of wild primates, specifically vervet monkeys. In Section 2.4 we summarize this study. We refer to the corresponding paper [7] for details on this study.

### 2.1 Basic properties

In previous analyses the BBMM has been used to obtain the distribution of the location at a given time, which is then typically integrated over time. In this section we use the distribution of location to derive distributions for other basic properties of moving entities: distance between two entities, velocity, speed and direction\(^1\) of an entity. Note that since we are dealing with distributions, we do not directly obtain speed and direction from velocity.

Deriving these distributions analytically is important for computational efficiency: while we could obtain them by numerical integration based on the location(s), such an integration would increase the computation time by a large factor. We present detailed derivations in this section, also to allow to

---

\(^1\)We express direction by its angle \(\alpha\) from the \(x\)-axis. Speed is the (scalar) absolute value of velocity (which is a vector).
use them as model for deriving properties (possibly also additional properties) under other random movement models.

2.1.1 Distance

For the distribution of distance consider two entities $A$ and $B$, the positions of which are independently normally distributed with mean positions $\mu_A(t)$ and $\mu_B(t)$ and variances $\sigma^2_A(t)$ and $\sigma^2_B(t)$. The difference of two independent normally distributed random variables is again normally distributed, so the distance between the positions of $A$ and $B$ at time $t$ again follows a circular normal distribution with mean $\mu_A(t) - \mu_B(t)$ and variance $\sigma^2_A(t) + \sigma^2_B(t)$.

The absolute value of a bivariate circular normal distribution is distributed according to the Rice distribution with parameters $\nu = \|\mu_A(t) - \mu_B(t)\|$ and $\sigma = \sqrt{\sigma^2_A(t) + \sigma^2_B(t)}$ [26]. See Figure 2.1 for examples of the density of this distribution.

For arbitrary dimension $d$, the distribution is related to the noncentral chi distribution with $d$ degrees of freedom. If $X = (X_1, \ldots, X_d)$ has a multivariate circular normal distribution with mean $\mu = (\mu_1, \ldots, \mu_d)$ and variance $\sigma^2$, then

$$Z = \frac{1}{\sigma} \sqrt{\sum_{i=1}^{d} X_i^2} = \frac{|X|}{\sigma}$$

has a noncentral chi distribution with $d$ degrees of freedom and noncentrality parameter

$$\lambda = \frac{1}{\sigma} \sqrt{\sum_{i=1}^{d} \mu_i^2} = \frac{|\mu|}{\sigma}.$$  

2.1.2 Velocity

Velocity (and therefore also speed and direction) are dependent on the temporal scale at which it is derived [28]. Thus instead of deriving a distribution for one point in time (as for distance), we derive a distribution for the average velocity over a time interval. In an analysis the length of the time interval then should be chosen according to the scale of the pattern considered and to the goal of the analysis.
The exact distribution of the velocity depends on the number of location measurements that were obtained during the interval of interest.

No measurements in the interval

Let $X_{(s)} \sim N\left(\mu_s, \sigma_s^2\right)$ and $X_{(f)} \sim N\left(\mu_f, \delta_f^2\right)$ represent the positions at the endpoints of a Brownian bridge that were obtained at times $t_s$ and $t_f$.

If we want to determine the velocity distribution over a time interval $[t_1, t_2]$ such that $t_s \leq t_1 < t_2 \leq t_f$, the positions $X_{(1)}, X_{(2)}$ of an entity at these times cannot be regarded as independent. Instead, we fix $X_{(1)}$ at a position $x$ and then use the Markov property of Brownian motion to derive a distribution for $X_{(2)}$ using Equation 1.1. We therefore obtain two Brownian bridges, one over $[t_s, t_f]$ parameterized by $\alpha = \frac{t_1 - t_s}{t_f - t_s}$ (where $T = t_f - t_s$), the other over $[t_1, t_f]$ parameterized by $\beta = \frac{t_2 - t_1}{t_f - t_1}$.

In the following, $\phi(x; \mu, \sigma^2)$ denotes the probability density function of a circular normal distribution with parameters $\mu$ and $\sigma^2$, evaluated at $x$, while $\phi(\mu, \sigma^2)$ denotes this density function without evaluating it. $f_X(x)$ denotes the probability density of a random variable $X$, evaluated at $x$. We integrate $X_{(s)}$, $X_{(1)}$ and $X_{(f)}$ over all positions they can take (i.e. $\mathbb{R}^d$) to obtain the distribution of the displacement over this interval:

$$fX_{(2)} - X_{(1)}(v) = \int_{\mathbb{R}^d} \int_{\mathbb{R}^d} fX_{(s)}(s) fX_{(f)}(f) fX_{(1)}(x | X_{(s)} = s \land X_{(f)} = f) dxdfds$$

$$= \int_{\mathbb{R}^d} \phi(s; \mu_s, \sigma_s^2) \phi(f; \mu_f, \delta_f^2) \int_{\mathbb{R}^d} \phi(x; (1 - \alpha)s + \alpha f, T\alpha(1 - \alpha)\sigma_m^2) \phi(x + v; (1 - \beta)x + \beta f, (t_f - t_1)\beta) dxdfds. \quad (2.1)$$

We first evaluate the inner integral to obtain the distribution of $X_{(2)} - X_{(1)}$ if $s$ and $f$ are fixed. We first rewrite the expression as a convolution and then use the property that the convolution of two normal densities is again the density of a normal distribution. Note that in this step we use

$$\frac{1 - \beta}{\beta} = \frac{(t_f - t_1) - (t_2 - t_1)}{t_2 - t_1} = \frac{t_f - t_2}{t_2 - t_1}.$$
Next we replace the inner integral of Equation 2.1 by this result and use the same technique to evaluate the middle integral:

\[
\int_{\mathbb{R}^d} \phi(f; \mu_f, \delta_f^2) \phi \left( 0; (1-\alpha)(s - f) + \frac{v}{\beta}, \left( T\alpha (1-\alpha) + \frac{(t_f - t_1)(t_f - t_2)}{t_2 - t_1} \right) \sigma_m^2 \right) df
\]

\[
= \int_{\mathbb{R}^d} \phi(f; \mu_f, \delta_f^2) \phi \left( (\alpha - 1)(s - f); \frac{v}{\beta}, \left( T\alpha (1-\alpha) + \frac{(t_f - t_1)(t_f - t_2)}{t_2 - t_1} \right) \sigma_m^2 \right) df
\]

\[
= \int_{\mathbb{R}^d} \phi(f; \mu_f, \delta_f^2) \phi \left( s - f; \frac{v}{(\alpha - 1)\beta}, \left( T\alpha (1-\alpha) + \frac{(t_f - t_1)(t_f - t_2)}{t_2 - t_1} \right) \sigma_m^2 \right) df
\]

\[
= \int_{\mathbb{R}^d} \phi(f; \mu_f, \delta_f^2) \phi \left( s - f; -\frac{T}{t_2 - t_1} v, \left( T\alpha (1-\alpha) + \frac{(t_f - t_1)(t_f - t_2)}{t_2 - t_1} \right) \sigma_m^2 \right) df
\]

\[
= \phi \left( s; \mu_f - \frac{T}{t_2 - t_1} v, \delta_f^2 + \left( T\alpha (1-\alpha) + \frac{(t_f - t_1)(t_f - t_2)}{t_2 - t_1} \right) \sigma_m^2 \right)
\]

Finally, we fill in this result and integrate with respect to \( s \) and simplify the resulting expression to obtain the displacement distribution over \([t_1, t_2]\). In the following we use \( \frac{\alpha}{1-\alpha} = \frac{t_1 - t_2}{t_f - t_1} \). In the last step we use \( \frac{t_2 - t_1}{t_f - t_1} = \beta \) and \( \frac{t_2 - t_1}{t_f - t_1} = \alpha \).

\[
\int_{\mathbb{R}^d} \phi(s; \mu_s, \delta_s^2) \phi \left( -s; \frac{T}{t_2 - t_1} v - \mu_f, \delta_f^2 + \left( T\alpha (1-\alpha) + \frac{(t_f - t_1)(t_f - t_2)}{t_2 - t_1} \right) \sigma_m^2 \right) ds
\]

\[
= \left[ \phi(\mu_s, \delta_s^2) * \phi \left( \frac{T}{t_2 - t_1} v - \mu_f, \delta_f^2 + \left( T\alpha (1-\alpha) + \frac{(t_f - t_1)(t_f - t_2)}{t_2 - t_1} \right) \sigma_m^2 \right) \right] (0)
\]

\[
= \phi \left( 0; \mu_f - \mu_s - \frac{T}{t_2 - t_1} v, \delta_s^2 + \delta_f^2 + \left( T(t_1 - t_s) + \frac{(t_f - t_2)T^2}{(t_f - t_1)(t_f - t_1)} \right) \sigma_m^2 \right)
\]

\[
= \phi \left( \frac{T}{t_2 - t_1} v; \mu_f - \mu_s, \delta_s^2 + \delta_f^2 + \left( T(t_1 - t_s) + \frac{(t_f - t_2)T^2}{(t_f - t_1)(t_f - t_1)} \right) \sigma_m^2 \right)
\]

\[
= \phi \left( v; \frac{t_2 - t_1}{T} (\mu_f - \mu_s), \left( \frac{t_2 - t_1}{T} \right)^2 \left( \delta_s^2 + \delta_f^2 \right) + \left( \frac{(t_f - t_2)(t_2 - t_1)}{T(t_f - t_1)} + \frac{(t_f - t_2)(t_2 - t_1)}{T(t_f - t_1)} \right) \sigma_m^2 \right)
\]

\[
= \phi \left( v; \frac{t_2 - t_1}{T} (\mu_f - \mu_s), \left( \frac{t_2 - t_1}{T} \right)^2 \left( \delta_s^2 + \delta_f^2 \right) + \beta (\alpha(t_2 - t_1) + (t_f - t_2)) \sigma_m^2 \right).
\]

From this distribution of the displacement, we obtain the velocity distribution by dividing by \( t_2 - t_1 \):

\[
V(t_1, t_2) = \frac{X_{(2)} - X_{(1)}}{t_2 - t_1}
\]

\[
\sim \mathcal{N} \left( \frac{\mu_f - \mu_s}{T}, \frac{\delta_s^2 + \delta_f^2}{T^2} + \beta \left( \frac{\alpha}{t_2 - t_1} + \frac{t_f - t_2}{(t_2 - t_1)^2} \right) \sigma_m^2 \right). \tag{2.2}
\]
2.1. Basic properties

We can simplify the last term of the variance using the definitions of $\alpha$ and $\beta$:

$$\frac{\alpha}{t_2-t_1} + \frac{t_f-t_2}{(t_2-t_1)^2} \sigma_m^2$$

$$= \frac{t_2-t_1}{t_f-t_1} \left( \frac{t_1-t_s}{(t_f-t_s)(t_2-t_1)} + \frac{t_f-t_2}{(t_2-t_1)^2} \right) \sigma_m^2$$

$$= \frac{1}{t_f-t_1} \left( \frac{t_1-t_s}{t_f-t_s} + \frac{t_f-t_2}{(t_2-t_1)^2} \sigma_m^2 \right)$$

$$= \frac{(t_1-t_s)(t_2-t_1) + (t_f-t_2)(t_f-t_s)}{(t_f-t_1)(t_f-t_s)(t_2-t_1)} \sigma_m^2$$

$$= \frac{(-t_1^2 + t_1(t_2 + t_s) - t_2t_s) + (t_f^2 - t_f(t_2 + t_s) + t_2t_s)}{(t_f-t_1)(t_f-t_s)(t_2-t_1)} \sigma_m^2$$

$$= \frac{(t_f-t_1)((t_f+1) - (t_2 + t_s)) \sigma_m^2}{(t_f-t_1)T(t_2-t_1)}$$

$$= \frac{(t_f-t_s) + (t_2 - t_1)}{(t_f-t_s)(t_2-t_1)} \sigma_m^2$$

$$= \frac{1}{t_f-t_s} + \frac{1}{t_2-t_1} \sigma_m^2$$

If we replace this in Equation 2.2, we obtain the following distribution for the velocity over $[t_2, t_1]$:

$$V(t_1, t_2) \sim N \left( \frac{\mu_f - \mu_s}{T}, \frac{\delta_s^2 + \delta_f^2}{T^2} + \left( \frac{1}{T} + \frac{1}{t_2-t_1} \right) \sigma_m^2 \right).$$

This result has intuitive appeal, since the mean velocity is exactly the velocity at which the mean location moves and identical to the velocity predicted by the linear movement model. One would expect the variance to grow as $t_2 - t_1$ decreases, and indeed the variance is inversely proportional to the length of the interval.

One measurement in the interval

Let $X_{(s)} \sim N(\mu_s, \delta_s^2)$, $X_{(i)} \sim N(\mu_i, \delta_i^2)$ and $X_{(f)} \sim N(\mu_f, \delta_f^2)$ represent the positions at three consecutive times $t_s$, $t_i$ and $t_f$ the location was sampled. If we want to determine the velocity distribution over a time interval $[t_1, t_2]$ such that $t_s \leq t_1 < t_i < t_2 \leq t_f$, the positions $X_{(1)}$, $X_{(2)}$ of an entity at these times cannot be regarded as independent, since they both depend on $X_{(i)}$. Instead, we fix $X_{(i)}$ at a position $x$ and then use the Markov property of Brownian motion to derive a distribution for $X_{(1)}$ and $X_{(2)}$. 
Define \( \alpha := \frac{t_2 - t_1}{t_1 - t_s} \) and \( \beta := \frac{t_2 - t_1}{t_2 - t_i} \).

\[
f_{X(2) - X(1)}(v) = \int_{\mathbb{R}^d} f_{X(i)}(x) f_{X(2) - X(1)}(v | X(i) = x) \, dx
\]
\[
= \int_{\mathbb{R}^d} \phi(x; \mu_i, \delta_i^2) \cdot \phi(v; \mu_x(t_2) - \mu_x(t_1), \sigma_x^2(t_1) + \sigma_x^2(t_2)) \, dx
\]
\[
= \int_{\mathbb{R}^d} \phi(x; \mu_i, \delta_i^2) \cdot \phi(v; (1 - \beta - \alpha)x + \beta \mu_e - (1 - \alpha) \mu_s, \sigma_x^2(t_1) + \sigma_x^2(t_2)) \, dx
\]
\[
= \int_{\mathbb{R}^d} \phi(x; \mu_i, \delta_i^2) \cdot \phi\left(\frac{v}{1 - \beta - \alpha} - x; \beta \mu_e + (\alpha - 1) \mu_s, \frac{\sigma_x^2(t_1) + \sigma_x^2(t_2)}{(1 - \beta - \alpha)^2}\right) \, dx
\]
\[
= \phi\left(\frac{v}{1 - \beta - \alpha}; \mu_i + \beta \mu_e + (\alpha - 1) \mu_s, \frac{\sigma_x^2(t_1) + \sigma_x^2(t_2)}{(1 - \beta - \alpha)^2}\right)
\]
\[
= \phi(v; (1 - \beta - \alpha) \mu_i + \beta \mu_e + (\alpha - 1) \mu_s, (1 - \beta - \alpha)^2 \delta_i^2 + \sigma_x^2(t_1) + \sigma_x^2(t_2))
\]
\[
= \phi(v; \mu(t_2) - \mu(t_1), \sigma^2(t_1) + \sigma^2(t_2) - 2\alpha(1 - \beta) \delta_i^2).
\]

We obtain the velocity distribution by dividing by \( t_2 - t_1 \):

\[
V(t_1, t_2) = \frac{X(2) - X(1)}{t_2 - t_1}
\]
\[
\sim \mathcal{N}\left(\frac{\mu(t_2) - \mu(t_1)}{t_2 - t_1}, \frac{\sigma^2(t_1) + \sigma^2(t_2) - 2\alpha(1 - \beta) \delta_i^2}{(t_2 - t_1)^2}\right).
\]

Again, the mean velocity is the average rate at which the mean is displaced over \([t_1, t_2]\). The variance differs from the variance for independent positions by \(-2\alpha(1 - \beta) \delta_i^2\), which is largest when one of the endpoints is at \( t_s \) and goes to zero as an endpoint approaches \( t_s \) or \( t_f \). This ensures that the distribution changes continuously as an interval of fixed length slides over a sampled trajectory.

**Two or more measurements in the interval**

Since we assume the positions at times of location measurements to be independent, \( X(1) \) and \( X(2) \) are independent if there are at least two measurements in \([t_1, t_2]\), so

\[
X(2) - X(1) \sim \mathcal{N}\left(\mu(t_2) - \mu(t_1), \sigma^2(t_1) + \sigma^2(t_2)\right).
\]

Thus we obtain the following expression for the velocity:

\[
V(t_1, t_2) = \frac{X(2) - X(1)}{t_2 - t_1}
\]
\[
\sim \mathcal{N}\left(\frac{\mu(t_2) - \mu(t_1)}{t_2 - t_1}, \frac{\sigma^2(t_1) + \sigma^2(t_2)}{(t_2 - t_1)^2}\right).
\]

**2.1.3 Speed and direction**

Now that we have established that the velocity follows a circular normal distribution, we use this to derive expressions for the average speed and the average direction of movement over a time interval \([t_1, t_2]\).
Figure 2.2: The variables involved in calculating the probability density of the direction of $X$ at $\gamma$ ($X$ is on the ray $r$) when $X$ has a circular normal distribution with mean $\mu$ with polar coordinates $(d, \theta)$.

Speed is the absolute value of velocity. As discussed in Section 2.1.1, this implies that in the two-dimensional case speed has a Rice distribution: for the velocity $V(t_1,t_2) \sim \mathcal{N}(\mu, \sigma^2)$, speed is distributed according to a Rice distribution with parameters $|\mu|$ and $\sigma$. In other dimensions, the distribution of speed is again related to the noncentral chi distribution.

For deriving the distribution of direction, it is convenient to work in polar coordinates. Let $d := |\mu|$, $\theta := \text{atan2}(\mu)$ represent $\mu$ in polar coordinates. See Figure 2.2 for an example of this situation. We consider direction only in two dimensions, the extension to higher dimensions is not directly obvious.

The probability density of a direction $\gamma$ may be considered to be the integral of the distribution of $V$ over an infinitesimally narrow cone with its apex at the origin. Let $u_\gamma$ be the unit vector at an angle $\gamma$ from the positive $x$-axis. This gives the following expression for the probability density function of the direction $\Gamma$:

$$f_{\Gamma}(\gamma) = \int_{-\infty}^{\infty} \phi(r u_\gamma; \mu, \sigma^2) \, r \, dr$$

$$= \int_{-\infty}^{\infty} \frac{1}{2\pi\sigma^2} e^{-\frac{|ru_\gamma - \mu|^2}{2\sigma^2}} \, r \, dr.$$

Now, rotate everything by an angle $-\gamma$ around the origin, such that $ru_\gamma = ru_0 = (r,0)$ runs over the positive $x$-axis. The new polar coordinates for $\mu$ are now $(d, \eta) := (d, \theta - \gamma)$, so we get the following expression for $f(\gamma)$.

$$f_{\Gamma}(\gamma) = \int_{0}^{\infty} \frac{1}{2\pi\sigma^2} e^{-\frac{|ru_\gamma - \mu|^2}{2\sigma^2}} \, r \, dr$$

$$= \frac{1}{2\pi\sigma^2} \int_{0}^{\infty} e^{-\frac{(r-d\cos\eta)^2 - (0-d\sin\eta)^2}{2\sigma^2}} \, r \, dr$$

$$= \frac{1}{2\pi\sigma^2} \int_{0}^{\infty} e^{-\frac{(r^2 - 2rd\cos\eta + d^2\cos^2\eta + d^2\sin^2\eta)}{2\sigma^2}} \, r \, dr$$

Evaluating this integral using Mathematica 8 [42] gives

$$f_{\Gamma}(\gamma) = \frac{e^{-\nu^2/2}}{2\pi} + \frac{\nu \cos\eta}{2\sqrt{2\pi}} \exp\left(\frac{\nu^2(\cos^2\eta - 1)}{2}\right) \left(1 + \text{erf}\left(\frac{\nu \cos\eta}{\sqrt{2}}\right)\right),$$

(2.3)

where $\nu := \frac{d}{\sigma}$ is the noncentrality of the velocity distribution and $\text{erf}(z) = \frac{2}{\sqrt{\pi}} \int_{0}^{z} e^{-t^2} \, dt$ is the Gauss error function. See Figure 2.3 for some examples of this distribution. For $\nu = 0$, the direction is
uniformly distributed as expected. For positive $\nu$, there is a preference for $\eta = 0$, i.e. the velocity is in the direction of the mean. This preference gets stronger as $\nu$ increases.

2.2 Movement patterns

In the following we formulate several movement patterns in the BBMM. These formulations also apply to other random movement models, but their computation would be different. In this section we focus on formulating the patterns and postpone most of the algorithmic aspects until Section 2.3. We first discuss some general questions concerning the formulation of movement patterns in the BBMM by the example of the encounter pattern. We then show how regular visit and following can be detected in the context of the BBMM.

2.2.1 Encounter and Avoidance/Attraction

If a pattern relates to an event at a certain point in time, we can express the pattern in terms of the probability of this event. For instance, an encounter occurs when the distance between two trajectories at a specific time is at most a certain threshold value $r$ [17]. In the case of linear interpolation it is easy to compute exactly when an encounter occurs. In the BBMM, it is not possible to say an encounter occurs at a specific time $t$, but we rather have a probability that there is an encounter at any time $t$. Thus, next to the threshold $r$ we use a threshold value $p_0$, and say that an encounter is likely if the distance is below $r$ with probability at least $p_0$.

In many cases, however, a pattern is also linked to a duration, for instance, we might want to know whether two entities encounter during a time span $[t_0, t_1]$. One might be tempted to formulate this as the probability that an encounter occurred at a time $t$ or all times $t \in [t_0, t_1]$. Such a formulation has the drawback that these probabilities are not easily computed based on the distributions of the underlying properties of the movement, since the probabilities for different values of $t$ are not independent. We therefore instead consider the expected duration of a pattern.

The probability of encounter at time $t$ is in the two-dimensional case determined by evaluating the cumulative distribution function (CDF) of the Rice distribution at the threshold distance. In other dimensions, the distance distribution is related to the noncentral chi distribution. Unfortunately, the cumulative distribution functions for both distributions cannot be evaluated directly. The expressions involve either an improper integral or an infinite sum, so we can only hope to approximate the CDF.

However, both distributions are closely related to the non-central chi-squared distribution [26], for which efficient implementations approximating the CDF exist. These implementations also give tight bounds on the error in the approximation. This relationship means that we can efficiently compute quantities related to the Rice and noncentral chi distribution as well.
Let $\mathbf{X}_{t,A}$ and $\mathbf{X}_{t,B}$ denote the positions at time $t$ of two entities $A$ and $B$, respectively. To get the expected duration of encounters with distance threshold $r$ over a time interval $[0,T]$ we integrate the probability of encounter with respect to time:

$$
E[\text{duration of encounters}] = \int_0^T P[|\mathbf{X}_{t,A} - \mathbf{X}_{t,B}| \leq r] \, dt. \tag{2.4}
$$

We compute a numerical approximation of this integral.

We define avoidance (and analogously attraction) as trajectories that have much space use in common, but rarely actually encounter each other. That is, both trajectories visit the same locations but at different times. We detect this by comparing the expected duration of encounters using the trajectory data (computed as described above) and using just the utilization distributions. The latter assumes that both entities move independently, so this comparison is a measure of correlation between their trajectories.

The utilization distribution is the probability distribution of the location of an entity. In the BBMM it can be estimated by integrating the distribution of location over the whole duration $T$ of observation, and dividing this integral by $T$. Given utilization distributions (UDs) $u_A(x)$ and $u_B(x)$ of two entities, one can compute the expected duration of encounters disregarding temporal information. This model assumes that the locations of both entities are independent. The probability of an encounter with distance threshold $r$ conditioned under the location $y$ of the first trajectory is therefore

$$
\int_{|x-y| \leq r} u_B(x) \, dx.
$$

The expected duration of an encounter is now $T$ times the probability of an encounter, that is,

$$
E(\text{duration of encounters based on UD}) = T \cdot \int_{\mathbb{R}^d} u_A(y) \int_{|x-y| \leq r} u_B(x) \, dx \, dy.
$$

We speak of avoidance if this expected value is considerably larger (that is, by some factor $f_0 > 1$) than the expected duration computed using Equation 2.4. We speak of attraction, if it is considerably smaller.

### 2.2.2 Regular visits

Another important pattern is detecting regular visits. A period $p$ and an offset $o$ define a sequence of $n$ time stamps $\{o + k \cdot p | k \in \{0, 1, \ldots, n - 1\}\}$. Given a trajectory $T$ and an area $A$, one can evaluate at each time stamp whether $T$ is inside $A$. A regular visit pattern is defined as a subsequence of these time stamps where the fraction of time stamps having $T$ inside $A$ is sufficiently high.

This problem is equivalent to finding the longest dense substring: Define a string with one bit $b_i$ for each time stamp such that $b_i = 1$ if and only if $T$ is inside $A$ at time $o + i \cdot p$. See Figure 2.4 for an example of this construction. The longest dense substring of a string of $n$ bits can be detected in $O(n)$ time, or $O(k)$ if the input string is run-length encoded with $k$ runs. [39, 13, 16].

The algorithm presented by Djordjevic et al. [16] detects the longest dense substring (with density $c$) in a run-length encoded string $s$ with $k$ runs in $O(k)$ time, regardless of the actual length of the string. The algorithm uses a function $f_1(i)$ that represents the number of 1’s in the first $i$ bits of $s$. A function $f_2(i) := f_1(i) - ci$ is derived from this function. A substring $s[i_1 \ldots i_2]$ of $s$ is dense if and only if $f_2(i_1) \leq f_2(i_2)$. So, finding the longest dense substring in $s$ is equivalent to finding the $i_1, i_2$ with $f_2(i_1) \leq f_2(i_2)$ such that $i_2 - i_1$ is maximal.
Chapter 2. Detecting Movement Patterns Using Brownian Bridges

Figure 2.4: Example demonstrating the equivalence of regular visit and longest dense substring. Each bit of the string is set if and only if $T$ is inside $A$ at the corresponding time stamp. Figure from [16].

Figure 2.5: Illustrations of the functions involved in the regular visit algorithm using the BBMM. (a) shows $f^*_1(i)$ and the line $c \cdot i$. In (b) we show $f^*_2(i) = f^*_1(i) - c \cdot i$ with its lower left and upper right envelopes.

In order to efficiently find these $i_1$ and $i_2$, their algorithm computes two envelopes: The lower left envelope $LLE(i) := \min_{j \in \{1, \ldots, i\}} f_2(j)$ and the upper right envelope $URE(i) := \max_{j \in \{i, \ldots, n\}} f_2(j)$. Then, two pointers $i_1$ and $i_2$ are walked along the LLE and URE, respectively. They both start at position 1, and $i_2$ is then advanced until $URE(i_2) < LLE(i_1)$. Then, $i_1$ is advanced until $URE(i_2) \geq LLE(i_1)$ holds again. This process is repeated until both pointers reach $n$. The largest difference that occurred between $i_2$ and $i_1$ during this process corresponds to the longest dense substring of $s$.

We adapt this algorithm to the BBMM. See Figure 2.5 for an example of the functions and envelopes used by the adapted algorithm. Once again, we can evaluate the probability that $T$ is inside $A$ at a certain time $t$, denoted $P[T(t) \in A]$, by integrating the density of the position distribution over $A$.

The original algorithm uses a function $f_1(i)$ which is the number of ones seen in the first $i$ bits of the bit string. Instead of counting the number of ones, we keep track of the expected number of ones in the first $i$ bits, i.e. the expected number of times that $T$ is inside $A$ in the first $i$ time stamps:

$$f^*_1(i) := \sum_{k=0}^{i} P[T(o + kp) \in A]$$

Running the original algorithm using this modified function, the algorithm produces the longest substring such that the expected density of this substring meets the threshold. So, the trajectory is inside the desired area at least the desired fraction of the time stamps on average.
2.2. Movement patterns

The original algorithm can process a run-length encoded bit string with \( k \) runs in \( O(k) \) time, regardless of the actual length \( n \) of the string. Unfortunately, it is not realistic for many consecutive time stamps to have exactly identical probabilities, so run-length encoding will not help us here. Instead, we need to evaluate the probability at each of the \( n \) time stamps.

When we have these probabilities, the rest of the algorithm only needs a constant number of passes over the data to compute the envelopes and find a solution. Therefore finding the longest expected dense substring can be solved in \( O(n \cdot a) \) time if \( \mathbb{P}[T(t) \in A] \) can be computed in \( O(a) \) time.

2.2.3 Following

Patterns expressing following behaviour have been defined in terms of the direction of movement [3] and in terms of the movement paths [6]. We here formulate both of these patterns in the setting of the BBMM.

Andersson et al. [3] define the front-region \( \text{front}(e) \) of an entity \( e \) as the wedge-shaped region with its apex at the entity’s position and centered around its movement direction. Its radius is \( r \) and its apex angle is \( \alpha \). If an entity \( B \) is in the front region of another \( A \) at some time \( t \), \( B \) is said to be in front of \( A \). They define that \( A \) is following \( B \) at time \( t \) if \( B \) is in front of \( A \) and \( |d_A - d_B| \leq \beta \), where \( d_A \) and \( d_B \) are the respective movement directions of \( A \) and \( B \). We illustrate these definitions in Figure 2.6.

In the linear model it is straightforward to detect when an entity is following another, since both their position and direction are known exactly, with the exception of the times that the location was sampled, as the direction is not defined then. In the BBMM these are not known exactly. We know the position at time \( t \) has a circular normal distribution. The distribution of direction does not give information about the instantaneous movement direction at time \( t \), but rather the average direction over a time interval. Let this interval have length \( \Delta t \) and be centered around \( t \). The distribution of the direction depends on the position of an entity. Let \( D_A \) and \( D_B \) be random variables representing the average direction over a time interval \( [t - \frac{\Delta t}{2}, t + \frac{\Delta t}{2}] \), with distributions as discussed in Section 2.1.3.

If we fix the position for both entities, we get a distribution for the directions and can therefore compute the probability that \( B \) is in front of \( A \). Let random variables \( X_A \) and \( X_B \) represent the positions of \( A \) and \( B \) respectively. Define \( d(x) \) as the direction of a vector \( x \). Now,

\[
\mathbb{P}[B \in \text{front}(A)|X_A = x_A \wedge X_B = x_B] = \begin{cases} \mathbb{P}[|D_A - d(x_B - x_A)| \leq \frac{\alpha}{2} |X_A = x_A] & \text{if } |x_A - x_B| \leq r \\ 0 & \text{otherwise.} \end{cases}
\]
Let $\delta_e$ denote the probability density function of the direction of entity $e$ as defined in Equation 2.3. We then get that

$$
\mathbb{P} \left[ |D_A - d(x_B - x_A)| \leq \frac{\alpha}{2} \bigg| X_A = x_A \right] = \int_{d(x_B - x_A) - \frac{\alpha}{2}}^{d(x_B - x_A) + \frac{\alpha}{2}} \delta_A(\theta) d\theta.
$$

We can also incorporate the criterion $|d_A - d_B| \leq \beta$ into this equation, obtaining

$$
\mathbb{P} \left[ |D_A - d(x_B - x_A)| \leq \frac{\alpha}{2} \wedge |D_A - D_B| \leq \beta \bigg| X_A = x_A \wedge X_B = x_B \right] = \int_{d(x_B - x_A) - \frac{\alpha}{2}}^{d(x_B - x_A) + \frac{\alpha}{2}} \delta_A(\theta) \int_{\theta - \beta}^{\theta + \beta} \delta_B(\gamma) d\gamma d\theta.
$$

Now, we can compute the probability that $A$ is following $B$ by integrating this expression over their positions, taking the probability of these positions into account:

$$
\mathbb{P} \left[ A \text{ is following } B \right] = \int_{\mathbb{R}^2} \int_{|x_A - x_B| \leq r} \frac{f_{X_A}(x_A) f_{X_B}(x_B)}{\mathbb{P}[|D_A - d(x_B - x_A)| \leq \frac{\alpha}{2} \wedge |D_A - D_B| \leq \beta \bigg| X_A = x_A \wedge X_B = x_B]} dxdA
$$

We can express this in terms of $\phi \left( x_A; \mu_A(t), \sigma_A^2(t) \bigg) \phi \left( x_B; \mu_B(t), \sigma_B^2(t) \bigg)$ as follows:

$$
\mathbb{P} \left[ A \text{ is following } B \right] = \int_{\mathbb{R}^2} \int_{|x_A - x_B| \leq r} \frac{\phi \left( x_A; \mu_A(t), \sigma_A^2(t) \bigg) \phi \left( x_B; \mu_B(t), \sigma_B^2(t) \bigg)}{\mathbb{P}[|D_A - d(x_B - x_A)| \leq \frac{\alpha}{2} \wedge |D_A - D_B| \leq \beta \bigg| X_A = x_A \wedge X_B = x_B]} dxdA
$$

Next we consider the path-based following pattern [6]. Intuitively this definition states that one entity is following another if it reaches approximately the same location but slightly later. Given two entities $A, B$ we consider their trajectories as functions of time, that is $A(t)$ (resp. $B(t)$) is the location of entity $A$ (resp. $B$) at time $t$. Now $B$ is following $A$ during the time interval $[s_1, t_1]$ with a time shift in $[\tau_{min}, \tau_{max}]$ and a distance threshold $r$ if there is an interval $[s_2, t_2]$ and a continuous, increasing function $\pi: [s_1, t_1] \rightarrow [s_2, t_2]$ such that for all $t \in [s_1, t_1]$ the distance $|A(\pi(t)) - B(t)|$ between $A(\pi(t))$ and $B(t)$ is at most $r$, and $\pi(t) \in [t - \tau_{max}, t - \tau_{min}]$. This definition is illustrated in Figure 2.7a.

In the linear motion model this pattern can be computed using a constrained free-space diagram [6], as illustrated in Figure 2.7b. This diagram is the subspace of the product of the parameter spaces of $A$ and $B$ of all pairs $(t, t')$ with the $|A(t') - B(t)| \leq r$ and $t' \in [t - \tau_{max}, t - \tau_{min}]$. Following then corresponds to an $xy$-monotone path in the constrained free space diagram. Computing the following pattern in this way is closely linked to computing the Fréchet distance between parameterized curves [1].

In the BBMM we have to deal with a distribution of distances. For any $(t, t')$ the distribution of the distance between $A(t')$ and $B(t)$ is a Rice distribution assuming that $A(t')$ and $B(t)$ are independent normal distributions. As with the encounter pattern, we formulate this pattern in the BBMM using expected duration. We therefore ask for the function $\pi$ that maximizes the expected length of time that $|A(\pi(t)) - B(t)| \leq r$ holds.
2.3 Algorithmic aspects

Many of the methods described above use in one way or another integrals of probability distributions and other functions. In many cases, these integrals cannot be evaluated analytically and so we have to use numerical integration. This often requires a large amount of resources to evaluate such an integral. Numerical integration involves many invocations of the same function with different arguments, as well as adding up the results of these function invocations. Both tasks can be efficiently implemented on a parallel processor like a GPU. In the following we give for the methods described in this chapter a simple analysis of the running times in terms of the most important parameters. Since we implemented several of the methods discussed in this chapter, we can also confirm that the parallel implementation has a large benefit in practice. We refer to Chapter 5 for a short discussion on these results.

In the following analysis, we use $N$ to denote the number of measurements in the trajectory being analysed and by $P$ the number of processors available. When numerically integrating a function over time, $S$ denotes the number of time steps. Similarly, $G$ is the size of the grid when integrating a function over space. Further, $t_0, \ldots, t_N$ are the time stamps at which the location was measured.

For the basic properties discussed in Section 2.1, an important operation is finding the parameters for their distribution at a specific time $t$. For example, we might want to obtain the mean and variance for the position of an entity. To obtain these parameters, we must first identify in which Brownian bridge $t$ lies, i.e. find the index $i$ such that $t_i \leq t \leq t_{i+1}$.

In Figure 2.7: Illustrations of path-based following. (a) Two trajectories with $B$ following $A$. The entity $B$ reaches the location $p_2$ shortly after $A$ has been at the nearby location $p_1$, and the time-shifted trajectory of $B$ from then on stays close to the trajectory of $A$. (b) Using a constrained free-space diagram to detect the following behaviour. The white area shows the parts of the product of the parameter space where $A$ and $B$ are close. Figures adapted from [6].

Instead of a path in the constrained free-space diagram we now have to deal with a line integral of probabilities over the product of parameter spaces. This is closely linked to the so called summed/integral Fréchet distance (or average Fréchet distance if divided by the duration) [5, 8]. While no exact algorithm for computing this distance is known, it can be efficiently approximated by sampling $A(t)$ and $B(t)$ and by approximating the line integral by a sum [5].
Chapter 2. Detecting Movement Patterns Using Brownian Bridges

Figure 2.8: Expected total encounter time between each pair of groups.

Since a trajectory is ordered on time, we can divide it into $P$ equal parts and use binary search on each part in parallel to obtain $i$ in $\Theta\left(\log \frac{N}{P}\right)$ time. Then we have to interpolate between measurements $i$ and $i + 1$ to obtain the value at time $t$. This can be done in constant time for all properties, so obtaining the parameters of the distribution at time $t$ takes $\Theta\left(1 + \log \frac{N}{P}\right)$ time.

Another important operation is computing the expected value for some metric, for example the expected duration of encounters described in Section 2.2.1. We divide the time to be integrated over in $P$ equal chunks.

Every processor has to identify in which bridge to start working, taking $\Theta\left(\log N\right)$ time. Then each processor has to evaluate the integrand $\frac{S}{P}$ times and add up the results, which requires $\Theta\left(\frac{S}{P}\right)$ time, assuming the integrand can be evaluated in constant time. Finally the results of all $P$ processors must be added together, which can be done in $\Theta\left(\log P\right)$ time. Therefore integrating a function over time takes $\Theta\left(\log N + \log P + \frac{S}{P}\right)$ time. Similarly we can integrate a function over space in $\Theta\left(\log N + \log P + \frac{G}{P}\right)$ time.

An operation that is not covered by this generic analysis is determining whether an entity follows another according to the model by Andersson et al., described in Section 2.2.3. This expression contains four nested integrals. The inner integral does not need to be fully evaluated for every step of the third integral, but instead its previous value can be updated in constant time.

The total number of operations in these integrations is $\Theta\left(G r^2 A(\alpha + \beta)\right)$, where $A$ is the angular separation of the steps in the inner integrals and $\alpha$, $\beta$ and $r$ are the parameters of the direction-based following pattern as described in Section 2.2.3. This algorithm is perfectly suitable to be run in parallel, so the running time is $\Theta\left(\frac{G r^2 A(\alpha + \beta)}{P}\right)$.

For the path-based following pattern, we refer to the discussion in Section 2.2.3: The computations for this pattern boil down to computing the summed Fréchet distance, which can be done efficiently when using numerical approximation as discussed in [5].

2.4 Case study

In the following we summarize our case study in [7] which applies the framework presented in this chapter. Here, we focus on how the framework is used, and refer for any ecological conclusions to [7].
2.4. Case study

The framework was implemented in R, an environment for statistical computing. For a more detailed discussion on this implementation, see Chapter 5.

The data set used in this case study contains data on four groups of vervet monkeys, with one representative for each group carrying a GPS tracker. The groups are identified as AK, BD, NH and RK. The trackers obtained one GPS-coordinate per hour over the daily activity period (5:00 – 19:00), for about three months.

This study focuses on the detection of encounters. We set the maximum distance at which an encounter is considered to occur at 100 meters. First we compare the (expected) duration of encounters in the linear model and the BBMM.

In Figure 2.8 we show the total duration of encounters between each pair of groups detected by both the linear motion model and the BBMM. Most groups have little to no interaction, but the pair BD-NH stands out here. For certain pairs of groups like NH and RK, the BBMM reports some encounter time while the linear model does not. This is to some extent expected since the BBMM always assigns a positive (although possibly very small) probability of encounter. However, the difference cannot be solely attributed to such small probabilities. An interesting note is that for the pair with the highest
Chapter 2. Detecting Movement Patterns Using Brownian Bridges

Figure 2.11: Distance distribution over time. The shaded band is the interval between the 5th and 95th percentile in the BBMM, the black line the distance in the linear model.

Figure 2.12: Probability of encounters over time. The background is shaded when the linear model detects an encounter.

encounter time, the BBMM reports a lower value. This is explained by the same reasoning: Using the BBMM the probability that there is no encounter is nonzero at any time.

Following the consideration of total duration of encounters between the different group dyads, we now turn our attention to days on which long encounters are expected to have occurred. Figure 2.9 shows pairs of daily trajectories for which either the linear model reports an encounter, or the BBMM indicates an expected duration of encounter exceeding 15 minutes. For each of these days we present the identities of the groups involved and show bar charts of the expected duration of encounter according to both movement models.

Focusing on a particular encounter, BD-NH day 1417 from Figure 2.9 is displayed in geographic space in Figure 2.10. Trajectories of both groups are given for both the linear movement model and the BBMM and are superimposed upon a high resolution satellite image, indicative of local primary vegetation productivity (darker shades of grey represent more photosynthetically active vegetation, with the river standing out as a light white band).

For the same groups and day Figure 2.11 shows the distance between the groups over time and Figure 2.12 shows the probability of encounter over time. Starting at approximately 15:00 hours until
the end of the measurement period, the groups moved along approximately parallel paths and both models recognize this as a likely encounter. Around 18:00 the measured distance is slightly larger than 100 meters and thus the linear model claims that no encounter occurs at that time. However, it seems likely that the groups were still encountering, but for some reason at the measurement time the distance was somewhat larger. The BBMM here reports a reasonable probability that an encounter is in progress at that time.

In Figure 2.11 we see that the distance estimate has a smaller error estimate at the measurement times (every full hour) than in between these times, as is expected with the BBMM. We also note that the distance estimate in the BBMM goes up when the mean is small compared to the standard deviation. This implies that the probability of encounter decreases between measurements, when the variance is larger. This effect can be clearly seen in Figure 2.12.

We can conclude from this study that using the BBMM we can detect interesting patterns (that is in this study, encounters), which we could have missed when using the linear movement model. Generally the results for the BBMM and the linear movement model were fairly similar in this study, but for long encounters the BBMM reports a shorter duration than the linear movement model, while for short encounters it reports a longer duration. This is the result of explicitly taking the location uncertainty at and between measurements into account, which can be investigated in more detail by for instance considering the quantiles of the distributions involved. This we see also as the main advantage of the BBMM in this study: It provides sound results under the reasonable (while not totally accurate) assumption of random movement between measurements, and it allows us to take the uncertainty at and between measurements into account in the evaluation of the results.
In this chapter we discuss how to combine computational movement analysis using the BBMM with the movement ecology paradigm by Nathan et al. [33]. Specifically we explore how various factors influencing a movement path can be incorporated into such an analysis. We differentiate between static and dynamic movement models, depending on whether the factors influence the models: If the movement model is static the movement path is computed based on the measurements and the model, and the factor is only used in the analysis. In a dynamic movement model the factor is incorporated in the model, thus influencing the movement path, which is then analysed.

The movement ecology paradigm provides a cohesive framework to study the movement of an organism. The paradigm distinguishes in general four components, which influence the movement (see also Figure 1.4): the organism’s internal state, its motion capacities, its navigation capacities and external factors. In this chapter we will focus on external factors and their influence on speed. In the next chapter we will consider the influence of behavioural states.

In Section 3.1 we extend the discussion of static and dynamic movement models. Since we will focus on how the factors influence speed, we need to get a measure of the speed of an animal at a particular location. In Section 3.2 we discuss how to obtain this spatial distribution of (average) speed. In Section 3.3, we discuss how visibility of other animals may influence a movement path. In Section 3.4 we demonstrate the case of a static model by a study on groups of vervet monkeys, and the case of a dynamic model by a study on the flight mode of European bee-eaters during migration.

### 3.1 Incorporating external factors

We distinguish two different ways to integrate external factors into the BBMM. Firstly, one could compute certain statistics related to the movement path and compare this a posteriori to an external influence. In this case the external influence does not affect the trajectory, and we therefore speak of a static model. An example would be to compute how fast an animal is moving in various areas and then compare these speeds to the availability of food resources in order to detect any correlations. The distribution of these resources is not considered when computing the distribution of speeds, but is only used after obtaining this distribution to find possible correlations. Another example is related to the detection of encounters as discussed in the previous chapter: Instead of using a distance threshold in a featureless space, we want to determine whether another animal is visible based on a digital elevation
model (DEM). This can be achieved by computing the position distribution of an animal using the BBMM and then using the DEM to evaluate the probability of seeing the other animal.

Secondly, external influences can be incorporated into the model a priori, thus affecting the computed trajectory. Since the movement model changes with the environment we speak of a dynamic model. For example, one could set the diffusion coefficient of the Brownian motion for each bridge separately based on a movement mode estimated from an external factor, e.g. for an airborne animal based on atmospheric conditions. Then the distribution of speeds can be computed using this model with a dynamically changing diffusion coefficient and used to evaluate the relation between the movement path and the atmospheric conditions.

In order to be suitable for a priori use, data will have to augment the trajectory, that is, it is ideally collected along with the location measurements. An example would be collecting weather data like temperature, pressure or light level from the same device that is tracking the animal’s location. In Chapter 4 we present another example using acceleration data. This situation is very different when environment data is given as a distribution of some values over space. In this case, it is more suitable for a posteriori analysis by comparing it to some spatial distribution derived from the trajectory data. This is because it is difficult to relate the trajectory at any time to a particular value of this field, since the location is not known exactly.

In this chapter we mainly focus on how speed is related to external factors and therefore we discuss how to derive a spatial distribution of speed in the next section.

3.2 Spatial distributions of speed

To obtain a spatial distribution of speed, we fix the position at a time \( t \) and then determine the mean speed over a time interval \([t + \Delta t_s, t + \Delta t_f]\). The mean speed at position \( x \) is obtained by averaging this value over \( t \), weighted by the probability density \( f_{X_t}(x) \). Since \( X_t \) is fixed, the position at a time before \( t \) and a time after \( t \) are independent by the Markov property of Brownian motion. Thus, if \( \Delta t_s \leq 0 \) and \( \Delta t_f \geq 0 \) we can determine the distribution of \( X_{t+\Delta t_s} \) and \( X_{t+\Delta t_f} \) independently and derive the velocity distribution from those.

### 3.2.1 Distribution of position with fixed \( X_t \)

We now derive the distribution of \( X_{t+\Delta t} \) for the case \( \Delta t > 0 \). The case \( \Delta t < 0 \) is symmetric and is therefore not discussed. If \( \Delta t = 0 \), then \( X_{t+\Delta t} = X_t \sim \mathcal{N}(x, 0) \) has a degenerate distribution and is thus known exactly.

Assume that the measurements directly before and after \( t \) were obtained at the times \( t_{i-1} \) and \( t_i \) with locations \( X_{(i-1)} \sim \mathcal{N}(\mu_{i-1}, \delta_{i-1}^2) \) and \( X_{(i)} \sim \mathcal{N}(\mu_i, \delta_i^2) \), respectively. We will use \( \alpha \) here as parameter for the Brownian bridge between \( t_{i-1} \) and \( t_i \) (as in Equation 1.1). The distribution of \( X_{(i)} \) is influenced by the location of \( x \). We use Bayes’ theorem to obtain the distribution of \( (X_{(i)}|X_t = x) \):

\[
fx_i(q|X_t = x) = \frac{fx_i(x|X_{(i)} = q) fx_{(i)}(q)}{fx_i(x)}
\]

\[
= \frac{\phi(x; (1-\alpha)\mu_{i-1} + \alpha q, (1-\alpha)^2\delta_{i-1}^2 + (t_i - t_{i-1})\alpha(1-\alpha)\sigma_m^2)}{fx_i(x)}
\]

\[
= \frac{\phi(q; \frac{\alpha - 1}{\alpha}\mu_{i-1} + \frac{x}{\alpha}, (\frac{1-\alpha}{\alpha})^2\delta_{i-1}^2 + \frac{(t_i - t_{i-1})(1-\alpha)}{\alpha}(1-\alpha)\sigma_m^2)}{fx_i(x)}
\]  

(3.1)
3.2. Spatial distributions of speed

We can define a smoothed Brownian bridge starting at \( x \) and ending at \( t = t_i \), conditioned on \( X_t = x \). Let \( \beta := \Delta t / \Delta i \). Then, \( X_{t+\Delta t} \sim N \left( (1 - \beta)x + \beta \mu_{i'}, \beta^2 \sigma_{i'}^2 + (t_i - t)\beta(1 - \beta)\sigma_{m}^2 \right) \).
We assume there that an encounter occurs when the distance is below some threshold \( r \). Therefore we will refine the encounter pattern by taking visibility into account.

The encounter pattern as described in Section 2.2.1 does not include any external factors into the model. Thus, we can straightforwardly derive the velocity distribution over \([t, t + \Delta t]\) for any point in time only estimate the probability that the groups see each other and the probability that they are closer than \( r \). However, if the animals involved do not perceive each other, the encounter is unlikely to influence the trajectories.

### 3.2.2 Conditioned speed distribution

Now that we know the distribution at any time conditioned on \( X_t \), we can determine this position distribution for two separate times and derive the distribution of the velocity from that. As discussed in Section 2.1.3, when the velocity is normally distributed, speed (its absolute value) as independent if \( \Delta t_s \leq 0 \) and \( \Delta t_f \geq 0 \). Let \( X_{t+\Delta t_s} \sim \mathcal{N}(\mu_{s}, \sigma_{s}^2) \) and \( X_{t+\Delta t_f} \sim \mathcal{N}(\mu_{f}, \sigma_{f}^2) \).

The velocity over the interval \([t + \Delta t_s, t + \Delta t_f]\) conditioned on \( X_t = x \) is given by

\[
V_{x,t}(t + \Delta t_s, t + \Delta t_f) = \frac{X_{t+\Delta t_f} - X_{t+\Delta t_s}}{\Delta t_f - \Delta t_s} \sim \mathcal{N}\left(\frac{\mu_{f} - \mu_{s}}{\Delta t_f - \Delta t_s}, \frac{\sigma_{s}^2 + \sigma_{f}^2}{(\Delta t_f - \Delta t_s)^2}\right).
\]

As discussed in Section 2.1.3, when the velocity is normally distributed, speed (its absolute value) is Rice distributed (in two dimensions). In order to determine a spatial distribution of speeds we compute a weighted average of the mean speed over time, keeping the interval over which the speed is computed fixed relative to the time at which the position is fixed. The weight is given by \( f_{X_t}(x) \). That is,

\[
S(x) := \frac{1}{\int f_{X_t}(x) \, dt} \int f_{X_t}(x) \mathbb{E}[|V_{x,t}(t + \Delta t_s, t + \Delta t_f)|] \, dt. \tag{3.4}
\]

The interval of integration is chosen such that the integrand can be evaluated for every point in the interval. That is, location data must be available at times \( t, t + \Delta t_s \) and \( t + \Delta t_f \).

### 3.3 Visibility

The encounter pattern as described in Section 2.2.1 does not include any external factors into the model. We assume there that an encounter occurs when the distance is below some threshold \( r \). However, if the animals involved do not perceive each other, the encounter is unlikely to influence the trajectories. Therefore we will refine the encounter pattern by taking visibility into account.

We define that an *encounter* happens if the groups can see each other and are closer than a certain distance \( r \) from each other. The value of \( r \) may depend on the identities of the groups. Due to the various sources of uncertainty, we can for any point in time only estimate the probability that the groups see each other and the probability that they are closer than \( r \).
3.4. Case studies

The first step is determining the view shed, i.e. the visible region, from each point on a grid using a digital elevation model (DEM). A DEM consists of elevation values at particular points in space; we assume that the elevation is given for each grid point on a rectangular grid.

The naïve algorithm to determine view sheds computes the line of sight between each pair of grid points and determines whether this line passes underneath the terrain anywhere. In particular, at every edge of the grid that is crossed by the line of sight, the height of the line of sight must be greater than the terrain elevation in order for the two points to see each other. On a square grid of \( n \) points, i.e. dimensions \( \sqrt{n} \cdot \sqrt{n} \), this takes \( O(n^2 \sqrt{n}) \) time, since there are \( O(n^2) \) pairs of grid points and each line of sight crosses at most \( O(\sqrt{n}) \) grid edges. The result has size \( \Theta(n^2) \) in the worst case.

However, in addition to the visibility requirement we also have the distance threshold, so it is not necessary to compute the full viewshed of each point, but only the intersection of the viewshed with the disc of radius \( r \) around the point. Computing this takes \( O(n \cdot (r/c)^3) \) time and the result has size \( O(n \cdot (r/c)^2) \), where \( c \) is the side length of a grid cell. In practice \( r \) is often equivalent to a small number of grid cells, so this is much better than the bound given above.

A problem with many DEMs obtained in practice is that they have poor vertical accuracy. To take the low vertical accuracy into account we model the elevation error by the method proposed by Fisher [19, 40]. The method deals with errors by adding a random error field with the desired properties (mean, standard deviation and autocorrelation) to the DEM, and then computing the viewshed on this modified DEM. The probability of a specific point being visible is estimated by the fraction of the runs in which the point is visible over many runs with different error fields.

To determine the probability of an encounter, we compute the position distributions of the two animals involved. These are then integrated over the relevant area while taking the visibility between each pair of locations into account. That is,

\[
P[\text{Encounter at time } t] = \int_{\mathbb{R}^2} \int_{|y-x| \leq r} f_{X_t,A}(x) f_{X_t,B}(y) P[y \text{ is visible from } x] \, dy \, dx,
\]

where \( X_{t,A} \) and \( X_{t,B} \) represent the locations of entities \( A \) and \( B \) at time \( t \), respectively. These encounter probabilities can be integrated over time to obtain the expected duration of encounters between groups.

3.4 Case studies

The results in this chapter are linked to two case studies that are currently work in progress\(^1\). We focus on the computation of speed distributions here, as the studies to the ecological relevance are out of the scope of this thesis.

In the first case study, we study the same data set collected on vervet monkeys that we used in Section 2.4. We focus on a single group of monkeys and want to determine how the speed of these monkeys is influenced by the amount of vegetation at their current location. Therefore we compute a spatial distribution of their speed as shown in Figure 3.1. The resulting distribution will then be compared to a map indicating the amount of vegetation present in this area to determine whether there are correlations between these distributions.

The speed distribution was computed at different time scales. For a given time scale \( \Delta t \), the mean speed was computed as defined in Equation 3.4, over two time intervals relative to the fixed point: one directly preceding it (i.e. \( \Delta t_s = -\Delta t \), \( \Delta t_f = 0 \)), and one directly following it (i.e. \( \Delta t_s = 0 \), \( \Delta t_f = \Delta t \)).

\(^1\text{Joint work with Kevin Buchin, Emiel van Loon, Nir Sapir, Stéphanie Mercier, T. Jean Marie Arseneau and Erik P. Willems}\)
Chapter 3. An Algorithmic Perspective on the Movement Ecology Paradigm

Figure 3.1: Spatial distribution of speed of a vervet monkey at different time scales, clipped to the 99% level home range.

If we had used only one of these intervals, we would not have been able to compute a speed near the beginning or end of the daily measurement period, which might lead to missing values in the distribution. We did not consider bridges at the beginning of the day that stayed very close ($\leq 50\text{m}$) to the starting location, as this indicates the monkeys had not started their daily activity period yet, and similarly at the end of the day near the final location.

There are no large differences in the results at different time scales, but generally a larger time scale leads to a smoother distribution.

In the second study, we evaluate how the flight mode of European bee-eaters affects their motion during migration. A bee-eater can perform flapping flight or soaring-gliding flight. In the latter which they use columns of rising air to gain altitude and then glide until they reach another such updraught. Which flight mode is used depends on the atmospheric conditions, as soaring-gliding flight uses less energy but is not always possible.

Since these flight modes have such different properties and information on which flight mode was used in each bridge is available, we computed a different diffusion coefficient for each mode (flapping: $2965\text{ m}^2/\text{s}$, soaring-gliding: $4505\text{ m}^2/\text{s}$), and used these in the model. For some bridges, a mixed flight mode was reported. In this case we assumed that the animal used both flight modes during this bridge. We used a logistic model learned from the available data to predict the fraction of soaring flight in these cases and used a weighted average of the two values for the diffusion coefficient. The distributions obtained for different time scales are shown in Figure 3.2.

Finally, we have computed the viewsheds from the digital elevation model of the area where the monkeys were observed. In Figure 3.3 we show some examples of viewsheds from a few different points. These can be used to refine estimates of encounter probabilities, though we do not show these here.

From these case studies we conclude that we can incorporate certain external factors when using the BBMM. The linear movement model does not provide any measure similar to the speed distributions shown here.
3.4. Case studies

Figure 3.2: Spatial distribution of speed of bee-eaters at different time scales, clipped to the 99% level home range.

Figure 3.3: Viewsheds for several locations drawn on the DEM. The viewing locations are indicated with a red dot, the green shaded areas indicate the viewsheds. The inner area is certainly visible, the outer area is possibly visible, based on 1000 simulated DEMs with error fields.
In the previous chapter we integrated our algorithmic methods with the movement ecology paradigm. In that chapter we focused on environmental factors. Now we will investigate how the internal state of an animal can be used to improve the predictions made by the BBMM. Specifically, we will investigate how to vary the diffusion coefficient based on the behavioural state of an animal.

While the main focus of this chapter is on this augmentation to the model, we will first discuss how the behavioural state of an animal can be derived from sensor data such as acceleration and GPS data. Well-known classification algorithms can be used to learn what an animal is doing from these sources. However, all classifiers make incorrect predictions for a reasonable fraction of data points.

If the behaviour of the animal is relatively stable over time or the data is sampled at a high rate, methods that take the time series data into account might have an advantage over classifiers that look at the data points in isolation. In Section 4.1 we present a classification algorithm and two approaches that use the time information to improve the results from this algorithm.

We will show how to integrate behaviour information into the BBMM by varying the diffusion coefficient in Section 4.2. We show how the BBMM can be used when the diffusion coefficient changes within one bridge or is allowed to vary continuously.

The method that we describe uses a specific interpretation of the diffusion coefficient, which is not compatible with the existing method to estimate its value. In Section 4.3 we present another method to estimate the diffusion coefficient that is based on an interpretation consistent with our method.

Finally, we use the methods described in this chapter in a case study on Oystercatchers. We show the results of the classification algorithms and how the predictions of the BBMM are improved by incorporating this behaviour data.

4.1 Time-dependent behaviour classification

Classification is the problem of assigning a class from a fixed set to an observation, using a model that was learned from a training data set of observations labelled with the correct classes. Many classification methods have been developed and their properties are well-known [23]. Acceleration data is frequently used for behaviour classification in movement ecology [34]. In this chapter we use data discussed in [37], where the authors used a decision tree classifier to derive the behaviour of Oystercatchers from high-resolution tri-axial acceleration data.

We first discuss a variation on the naive Bayes classifier for this behaviour classification. This classifier serves as the base for the time-dependent approaches that we present later in this section.
We observe that the behaviour of an animal is often stable for extended periods of time. The classifier sometimes mispredicts the behaviour for an isolated observation in a long series of identical behaviours. We augment the classifier using two approaches that consider the predictions for observations that are close in time, in addition to the probabilities predicted by the classifier.

The first of these approaches is based on segmentation [9, 10], which is the problem of dividing a trajectory into a (preferably small) number of subtrajectories such that each of these subtrajectories is homogeneous in some sense. Here, each subtrajectory consists of observations with identical behaviour and we use a penalty for changing behaviours to reduce the number of subtrajectories.

The second approach uses hidden Markov models, which have the advantage that each behaviour can be treated differently, based on data about the frequency of each behaviour and of transitions between a pair of behaviours.

4.1.1 The Bayesian classifier

The Naive Bayes classifier is a well-known method for classification, based on Bayes’ theorem:

\[
P(Y = y | X = x) = \frac{P(Y = y) f_X(x | Y = y)}{f_X(x)}.
\]

Here, \( Y \) represents the classes the classifier may predict and \( X \) is a vector of attribute values.

Its advantages are that it is robust to noise and irrelevant attributes. Another feature that is required for our method is that it provides an estimate of the probability that a data point belongs to each class, instead of just reporting a single most likely class.

A disadvantage is that Naive Bayes does not perform well for correlated attributes. It assumes that each attribute is independent, which does not work well for the data set at hand.

Therefore, we used a variation on this classifier that can take correlation into account. The classifier assumes that for each class, the data points are distributed according to a multivariate normal distribution, where the dimension is the number of attributes.

To learn the model, the classifier estimates \( P(Y = y) \), the prior probability of each class, by counting its frequency in the training data. The parameters of the normal distribution that describes the class-conditional probability density of the measurements, \( f_X(x | Y = y) \), are estimated using the sample mean and the sample covariance matrix of the data points that are labelled with class \( y \).

The denominator in Bayes’ theorem, \( f_X(x) \) can be ignored, since it is a constant for all classes. Alternatively, it may be used as a normalization constant, such that

\[
\sum_y P(Y = y | X = x) = \sum_y \frac{P(Y = y) f_X(x | Y = y)}{f_X(x)} = 1.
\]

To obtain the probability of class \( y \) given data point \( x \), we evaluate the above formula using the model obtained during the learning stage. That is,

\[
P(Y = y) = \text{the fraction of the training data set labelled as } y,
\]

\[
f_X(x | Y = y) = (2\pi)^{-\frac{k}{2}} |\Sigma_y|^{-\frac{1}{2}} \exp \left( -\frac{1}{2} (x - \mu_y)^T \Sigma_y^{-1} (x - \mu_y) \right).
\]

The latter function is the probability density function of a \( k \)-dimensional normal distribution with mean \( \mu_y \) and covariance matrix \( \Sigma_y \). Their values depend on the class \( y \) and are estimated during the learning stage.

The classifier computes the probability for each class and then assigns the class with the highest probability to the data point.
4.1. Time-dependent behaviour classification

4.1.2 Segmentation based on the Bayesian information criterion

We observe that classifiers often produce incorrect predictions for isolated data points, while the surrounding points were classified correctly. Often this would occur during long runs of identical observations, so taking the predictions for these neighbouring points into account could remove these isolated errors.

We use the Bayesian information criterion (BIC) to penalize switching behaviours too often. The BIC is a method to evaluate the quality of models while avoiding too detailed models that would probably be overfitted to the training data. The BIC has been used previously to obtain a time-varying diffusion coefficient for the BBMM [27]. The BIC is defined as

\[ \text{BIC} = -2 \log L + k \log n. \]

Here \( L \) is the likelihood of the selected model, \( k \) is the number of free parameters in the model and \( n \) the number of data points. In our case \( k \) is the number of times the class changes between two consecutive observations, i.e. the number of segments with identical classes.

The likelihood of a sequence of behaviours \((y_1, \ldots, y_n)\) given observations \((x_1, \ldots, x_n)\) is estimated using the classifier described in the previous section:

\[ L = \prod_{i=1}^{n} \frac{\mathbb{P}[Y_i = y_i \mid X_i = x_i]}{f_{X_i}(x_i)}, \]

or equivalently:

\[ \log L = \sum_{i=1}^{n} \log \left( \frac{\mathbb{P}[Y_i = y_i \mid X_i = x_i]}{f_{X_i}(x_i)} \right). \]

Define \( I_{y_i \neq y_{i-1}} \) as the indicator variable for the condition \( y_i \neq y_{i-1} \). Then we obtain the following expression for the BIC:

\[ \text{BIC} = -2 \log L + k \log n \]

\[ = \sum_{i=1}^{n} -2 \log \left( \frac{\mathbb{P}[Y_i = y_i \mid X_i = x_i]}{f_{X_i}(x_i)} \right) + \sum_{i=1}^{n} I_{y_i \neq y_{i-1}} \log n \]

\[ = \sum_{i=1}^{n} \left( -2 \log \left( \frac{\mathbb{P}[Y_i = y_i \mid X_i = x_i]}{f_{X_i}(x_i)} \right) + I_{y_i \neq y_{i-1}} \log n \right) \]

The model with the minimum BIC is considered optimal.

We propose a dynamic programming algorithm to efficiently find the optimal model. The algorithm computes a table of size \( n \cdot |B| \), with \( B \) the set of behaviours. Every row of the table represents one behaviour, every column represents one observation. A cell \((i, b)\) contains the BIC of the best model for the first \( i \) observations that ends with behaviour \( b \). With each cell, we also store a pointer to a cell in the previous column to identify the behaviour that preceded this behaviour in the optimal model.

The BIC for the cells in the first column can be computed as

\[ \text{BIC}(1, b) = -2 \log \mathbb{P}[Y_1 = b \mid X_i = x_i]. \]

In computing the value of the optimal BIC for an other cell \((i, b)\) we distinguish two cases:

- Do not switch behaviours from the previous observation and do not get a penalty. The optimal BIC is obtained by updating the value in cell \((i-1, b)\).
Switch behaviours and get a penalty for adding a new parameter to the model. Since the penalty does not depend on the states involved in this switch, we only need to update the value from the cell with the lowest BIC in column \( i - 1 \).

Formally, we can summarize this as:

\[
BIC(i, b) = \min \left\{ \begin{array}{l}
BIC(i - 1, b) - 2 \log P \left[ Y_i = b \mid X_i = x_i \right], \\
\min_{b^* \in B} BIC(i - 1, b^*) - 2 \log P \left[ Y_i = b \mid X_i = x_i \right] + \log n \end{array} \right. 
\]

The predecessor is set to either \( b \) or the behaviour with the best BIC in the previous column, depending on what choice produced the minimum BIC.

After all cell values are computed, the optimal model has \( BIC = \min_{b^* \in B} BIC(n, b^*) \), i.e. the smallest value occurring in the last column of the table. The cell in which it occurs gives us the behaviour at the last observation, the others can be obtained by following the predecessor pointers from that cell back to the first column.

Assuming that the underlying classifier computes the probability of a class given an observation in constant time, the values of the cells in the first column can be evaluated in constant time. For the other cells, the expression \( \min_{b^* \in B} BIC(i - 1, b^*) \) takes \( O(|B|) \) time to compute, but since it does not depend on \( b \) it has to be computed only once for every column of the table, giving an \( O(1) \) amortized time per cell. Given this minimum, the value of each cell in the table takes \( O(1) \) time to compute. Since there are \( n \cdot |B| \) cells, filling the table takes \( O(n|B|) \) time. Finally, the model can be reconstructed in \( O(n) \) time once the table is filled, so the algorithm runs in time \( O(n|B|) \).

### 4.1.3 Classification using hidden Markov models

Another way to estimate the probability of a sequence of observations is using a hidden Markov model. This consists of three parts:

- \( \pi = (\pi_1, \ldots, \pi_{|B|}) \); the prior probability of each class.
- \( A = \{a_{st}\} \), with \( a_{st} = P \left[ y_i = t \mid y_{i-1} = s \right] \); Transition matrix, indicating the transition probability between each pair of states.
- \( E_{y}(x) = f_X(x \mid Y = y) \); Emission probabilities, a set of functions giving the probability density of observation \( x \) when the model is in state \( y \).

For the emission probability and the prior probabilities, we again use the values computed by the Bayesian classifier. The prior probability of a class is estimated by the fraction of observations that belong to that class and the transition matrix is estimated by counting the number of transitions between each pair of states that occur in the training data set.

The likelihood of a sequence of behaviours \( (y_1, \ldots, y_n) \) given observations \( (x_1, \ldots, x_n) \) is

\[
P \left[ (y_1, \ldots, y_n) \mid (x_1, \ldots, x_n) \right] = \pi_{y_1} E_{y_1}(x_1) \prod_{i=2}^{n} a_{y_{i-1}y_i} E_{y_i}(x_i).
\]

We use the Viterbi algorithm [38, 20] to obtain the most likely sequence in time \( \Theta(n|B|^2) \).

We evaluate our time-dependent classification algorithms in a case study in Section 4.4, for which the time-dependent approaches indeed yield an improvement.
4.2 Incorporating behaviour data in the BBMM

Behavioural information can be integrated into the BBMM by varying the diffusion coefficient. If the diffusion coefficient varies only between bridges, this is straightforward since the distributions of relevant quantities depend only on local information. However, if the behaviour data has a higher sampling rate than the location data, this is not straightforward. Molini et al. [31] discuss Brownian motion with time-dependent drift and diffusion coefficient. We do not consider Brownian motion with drift here, but the diffusion coefficient may indeed vary over time. They show that if \( X_t \) is a Brownian motion which starts at the origin and with a positive integrable function of time \( \sigma_m^2(t) \) representing the diffusion coefficient, then its distribution is

\[
X_t \sim \mathcal{N} \left( 0, \int_0^t \sigma_m^2(s) ds \right).
\]

By conditioning such a Brownian motion on its position at a particular time \( T \), we can derive the distribution of a Brownian bridge with time-dependent diffusion. Define \( D_t' := \int_0^t \sigma_m^2(s) ds \). We derive the following distribution for \( X_t^c \):

\[
f_{X_t^c}(x | X_0 = a \land X_T = b) = \frac{f_{X_t}(x) f_{X_T}(b | X_t = x)}{f_{X_T}(b)} = \frac{\phi(x; a, D_0') \phi(b; x, D_t')}{\phi(b; a, D_t')}
\]

\[
= \frac{1}{2\pi D_0'} \exp \left( \frac{-|x-a|^2}{2D_0'} \right) \frac{1}{2\pi D_t'} \exp \left( \frac{-|b-x|^2}{2D_t'} \right)
\]

\[
= \frac{1}{2\pi \frac{D_0'}{D_t'} \frac{D_t'}{D_0'}} \exp \left( \frac{-|x-a|^2}{2D_0'} + \frac{-|b-x|^2}{2D_t'} - \frac{2(b \cdot a)}{2D_0'} \right). \quad (4.1)
\]

We rewrite the expression in the exponent to obtain:

\[
\frac{-|x-a|^2}{2D_0'} + \frac{-|b-x|^2}{2D_t'} - \frac{2(b \cdot a)}{2D_0'}
\]

\[
= \frac{D_0'}{2D_0'D_t'} \left( \frac{D_t'}{D_0'} |x-a|^2 - \frac{D_0'}{D_t'} |b-x|^2 + \frac{D_t'D_0'}{(D_0')^2} |b-a|^2 \right)
\]

\[
= \frac{D_0'}{2D_0'D_t'} \left( \frac{-D_t'}{D_0'} (|x|^2 - 2a \cdot x + |a|^2) - \frac{D_0'}{D_t'} (|x|^2 - 2b \cdot x + |b|^2) + \frac{D_t'D_0'}{(D_0')^2} (b^2 - 2a \cdot b + |a|^2) \right)
\]

\[
= \frac{D_0'}{2D_0'D_t'} \left( -|x|^2 + 2x \cdot \left( \frac{D_t'}{D_0'} a + \frac{D_0'}{D_t'} b \right) - \left( \frac{D_t'}{D_0'} \right)^2 |a|^2 + 2 \frac{D_t'D_0'}{(D_0')^2} a \cdot b + \left( \frac{D_0'}{D_t'} \right)^2 |b|^2 \right)
\]

\[
= \frac{D_0'}{2D_0'D_t'} \left( -|x|^2 + 2x \cdot \left( \frac{D_t'}{D_0'} a + \frac{D_0'}{D_t'} b \right) - \left| \frac{D_t'}{D_0'} a + \frac{D_0'}{D_t'} b \right|^2 \right)
\]

\[
- |x - \left( \frac{D_t'}{D_0'} a + \frac{D_0'}{D_t'} b \right)|^2.
\]
Figure 4.1: The method for estimating the diffusion coefficient by Horne et al. [24]. While the two trajectories are very similar, the estimates produced by this method are very different: For the trajectory in (a), zero is the most likely estimate, while the estimate for the trajectory in (b) will be rather large.

If we substitute this expression into Equation 4.1, we obtain the following result:

$$\frac{1}{2\pi \frac{D_t}{D_0} \frac{D_T}{D_0}} \exp \left( -\frac{\left| x - \left( \frac{D_T}{D_0} a + \frac{D_t}{D_0} b \right) \right|^2}{2 \frac{D_t}{D_0} \frac{D_T}{D_0}} \right) = \phi \left( x; \frac{D_T}{D_0} a + \frac{D_t}{D_0} b, \frac{D_t}{D_0} \right).$$

In this derivation, we used the equation for a bivariate normal distribution, but it works in any dimension. Thus, $X_t$ has a normal distribution with parameters

$$\mu(t) = \frac{D_T}{D_0} a + \frac{D_t}{D_0} b = (1 - \alpha) a + \alpha b,$$

$$\sigma^2(t) = \frac{D_t^2}{D_0^2} + \alpha (1 - \alpha) D_0^2,$$

with

$$\alpha := \frac{D_t}{D_0} = \frac{\int_0^t \sigma_m^2(s) ds}{\int_0^T \sigma_m^2(s) ds}.$$

We can also use smoothed Brownian bridges where the locations at the endpoints are not fixed, but rather described by normal distributions $X_0 \sim \mathcal{N}(a, \delta_a^2)$ and $X_T \sim \mathcal{N}(b, \delta_b^2)$. That does not affect the mean location, and the expression for the variance becomes

$$\sigma^2(t) = (1 - \alpha)^2 \delta_a^2 + \alpha^2 \delta_b^2 + \alpha (1 - \alpha) D_0^2.$$

This result is very similar to the distribution of position when the diffusion coefficient is constant as defined in Equation 1.2, and in that case the parameters derived above are identical to those defined there. The main difference is that $\alpha$ does not move from zero to one at a constant rate as $t$ goes from 0 to $T$, but at a rate that is proportional to the diffusion coefficient at $t$.

### 4.3 Estimating the diffusion coefficient

Horne et al. [24] presented a method for estimating the diffusion coefficient using a maximum likelihood method. Let $x(0), \ldots, x(n)$ be the measured locations for a trajectory, measured at times $t_0, \ldots, t_n$, and let $X(0), \ldots, X(n)$ be the random variables representing the actual locations at these times.
4.3. Estimating the diffusion coefficient

In this approach, one would use the even-numbered measurements to construct Brownian bridges on the time intervals \([t_0, t_2], [t_2, t_4], \ldots, [t_{n-2}, t_n]\). The odd-numbered measurements are regarded as independent measurements and used to estimate the diffusion coefficient. This is illustrated in Figure 4.1. Given a candidate value \(s^2\) of the diffusion coefficient, the likelihood of this value given a certain location \(x_{(i)}\) is

\[
L_i(s^2) = \phi(x_{(i)}; 1 - \alpha)x_{(i-1)} + \alpha x_{(i+1)}, (1 - \alpha)^2 \delta_{i-1}^2 + \alpha^2 \delta_{i+1}^2 + (t_{i+1} - t_{i-1})\alpha(1 - \alpha)s^2,\]

with \(\alpha = \frac{t_{i+1} - t_{i-1}}{t_{i+1} - t_{i-1}}\). That is, the probability density \(f_{X_{(i)}}(x_{(i)})\) of observing \(x_{(i)}\) in a model with diffusion coefficient \(s^2\). The likelihood of observing \(x_{(1)}, x_{(3)}, \ldots, x_{(n-1)}\) given a diffusion coefficient \(s^2\) is then given by the product of these individual likelihoods for all odd-numbered observations:

\[
L(s^2) = \prod_{i=1}^{n/2} L_{2i-1}(s^2).
\]

The value for \(s^2\) that maximizes the likelihood is used as the diffusion coefficient. This method treats the measurements differently depending on where they occur in the sequence of measurements. Even-numbered measurements are treated as part of the model, while the odd-numbered measurements are treated as the observations to which we are trying to fit the model. This implies that the time at which the observations start or a single missing location (e.g. due to loss of GPS signal) can greatly influence the estimate, as shown in Figure 4.1.

In (unconstrained) Brownian motion, the diffusion coefficient is a measure for how fast the position distribution diffuses. Informally, this is proportional to the average speed at which an entity moves while performing Brownian motion.

In the method described here the diffusion coefficient is a measure for how much the motion deviates from piecewise straight-line motion with constant speed, with the breaks occurring at the even-numbered observations. This might often work well in practice, since the motion of an animal is never truly Brownian, but it leads to problems when using the model to allow a diffusion coefficient that varies over time, that was described in the previous section.

There is an implicit assumption in the way the model was derived: The diffusion coefficient is a measure for the rate at which the position distribution diffuses if not constrained. This does not align well with the existing method for estimating the diffusion coefficient. Therefore, we present a new method for estimating the diffusion coefficient that is based on this assumption. This method also has the advantage that all observations are treated equally and is less sensitive to small differences in the data like a missing location.

Given two locations \(x_{(i)}\) and \(x_{(i+1)}\), we can compute the distribution of \(X_{(i+1)}\) at time \(t_{i+1}\) for a Brownian motion with diffusion coefficient \(s^2\), starting from \(x_{(i)}\) at time \(t_i\):

\[
(X_{(i+1)} | X_{(i)} = x_{(i)}) \sim \mathcal{N}(x_{(i)}, (t_{i+1} - t_i)s^2).
\]

Equivalently, the likelihood of a diffusion coefficient \(s^2\) given these two observations is described by the density of this distribution at \(x_{(i+1)}\), i.e.

\[
L_i(s^2) = \phi(x_{(i+1)}; x_{(i)}, (t_{i+1} - t_i)s^2).
\]

In this model, the most likely value for \(s^2\) is proportional to the distance between \(x_{(i)}\) and \(x_{(i+1)}\), as is expected for the diffusion coefficient of a Brownian motion. If we take the uncertainty of these locations into account, we obtain

\[
L_i(s^2) = \phi(x_{(i+1)}; x_{(i)}, \delta_i^2 + \delta_{i+1}^2 + (t_{i+1} - t_i)s^2).
\]
Given an arbitrary number of observations \( x_0, \ldots, x_n \), we can treat each consecutive pair of observations as an independent Brownian motion due to the Markov property. Therefore, the likelihood of a value \( s^2 \) is the product of the likelihoods for the individual bridges:

\[
L(s^2) = \prod_{i=0}^{n-1} L_i(s^2).
\]

The diffusion coefficient that is used is the value that maximizes this likelihood. Looking back at the example in Figure 4.1, we observe that this method would predict exactly the same value of the diffusion coefficient for both trajectories, as both trajectories have the same number of bridges of each particular length.

When each measurement is classified with a behaviour, we can estimate the diffusion coefficient for each behaviour separately. Let \( B_b \) be the set of bridges for which the behaviour is \( b \) for the whole duration of the bridge. We use the maximum likelihood method described above on this set of bridges to estimate the diffusion coefficient for this behaviour.

If a bridge has different observed behaviours over its duration, one can interpolate the value of the diffusion coefficient between the times at which the behaviour is known. If the times at which the behaviour changes are known exactly, a piecewise constant function can be used to describe the diffusion coefficient.

### 4.4 Oystercatcher case study

We investigate what improvements the methods developed in this chapter can offer using data collected from Oystercatchers [37]. Part of the data was manually labelled with the observed behaviours, which provides an excellent opportunity to train and evaluate classification methods, as well as evaluating the augmented bridges without errors introduced by the classifier.

#### 4.4.1 Results of the classification algorithms

We used a data set of 695 data points which were labelled with observed behaviours. The provided data set contains 702 such labelled points, but 7 of them had no GPS data and therefore also no timestamp when the data point was collected. Since our methods depend on the order of the data points, we could not use these 7 points. The behaviours were classified at three different granularities into 3, 8 or 16 classes, as described in Table 1 of [37].

We used 16 attributes that were derived from the accelerometer or GPS data in the classifier. These attributes are described in [37], Table 2. We did not use odba, since this is a linear combination of the attributes odbaX, odbaY and odbaZ and thus adds no new information for this particular classifier.

To evaluate the performance of the methods, we used 10-fold cross-validation, meaning the data set was split in 10 partitions. 9 partitions were used for training the models and 1 partition for evaluating their performance. This was repeated 10 times with a different partition for evaluation.

We investigate the quality of the classification methods on the classification into 8 classes. The results are summarized in Table 4.1 in the form of confusion matrices. The accuracy of the Bayesian classifier is 67.2%. We observe that the models that take time into account both perform better than the classifier alone, with accuracies of 70.6% and 68.6% for the segmentation model and hidden Markov model based approaches, respectively. If we focus on classification into two classes, namely detecting when foraging (the most common behaviour) occurs, the accuracies are 80.7%, 83.1% and 82.7%, respectively.
### 4.4. Oystercatcher case study

<table>
<thead>
<tr>
<th>predicted:</th>
<th>aggression</th>
<th>body care</th>
<th>fly</th>
<th>forage</th>
<th>handle</th>
<th>sit</th>
<th>stand</th>
<th>walk</th>
<th>Totals predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>aggression</td>
<td>15</td>
<td>6</td>
<td>3</td>
<td>7</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>37</td>
</tr>
<tr>
<td>body care</td>
<td>14</td>
<td>61</td>
<td>0</td>
<td>14</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>2</td>
<td>100</td>
</tr>
<tr>
<td>fly</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>forage</td>
<td>9</td>
<td>8</td>
<td>3</td>
<td>194</td>
<td>39</td>
<td>14</td>
<td>5</td>
<td>12</td>
<td>284</td>
</tr>
<tr>
<td>handle</td>
<td>4</td>
<td>0</td>
<td>26</td>
<td>9</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>42</td>
</tr>
<tr>
<td>sit</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>79</td>
<td>2</td>
<td>0</td>
<td>82</td>
</tr>
<tr>
<td>stand</td>
<td>6</td>
<td>8</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>4</td>
<td>107</td>
<td>1</td>
<td>128</td>
</tr>
<tr>
<td>walk</td>
<td>1</td>
<td>2</td>
<td>7</td>
<td>8</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>2</td>
<td>22</td>
</tr>
<tr>
<td>Totals obs.</td>
<td>49</td>
<td>85</td>
<td>13</td>
<td>252</td>
<td>51</td>
<td>98</td>
<td>122</td>
<td>25</td>
<td>695</td>
</tr>
</tbody>
</table>

(a) Bayesian classifier, accuracy 67.2%.

<table>
<thead>
<tr>
<th>predicted:</th>
<th>aggression</th>
<th>body care</th>
<th>fly</th>
<th>forage</th>
<th>handle</th>
<th>sit</th>
<th>stand</th>
<th>walk</th>
<th>Totals predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>aggression</td>
<td>17</td>
<td>5</td>
<td>3</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>4</td>
<td>36</td>
</tr>
<tr>
<td>body care</td>
<td>11</td>
<td>63</td>
<td>0</td>
<td>9</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>91</td>
</tr>
<tr>
<td>fly</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>forage</td>
<td>12</td>
<td>8</td>
<td>3</td>
<td>222</td>
<td>48</td>
<td>14</td>
<td>6</td>
<td>15</td>
<td>328</td>
</tr>
<tr>
<td>handle</td>
<td>2</td>
<td>0</td>
<td>0</td>
<td>7</td>
<td>3</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>15</td>
</tr>
<tr>
<td>sit</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>78</td>
<td>2</td>
<td>0</td>
<td>81</td>
</tr>
<tr>
<td>stand</td>
<td>6</td>
<td>8</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>5</td>
<td>108</td>
<td>1</td>
<td>130</td>
</tr>
<tr>
<td>walk</td>
<td>1</td>
<td>1</td>
<td>7</td>
<td>5</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>Totals obs.</td>
<td>49</td>
<td>85</td>
<td>13</td>
<td>252</td>
<td>51</td>
<td>98</td>
<td>122</td>
<td>25</td>
<td>695</td>
</tr>
</tbody>
</table>

(b) BIC-based segmentation, accuracy 70.6%.

<table>
<thead>
<tr>
<th>predicted:</th>
<th>aggression</th>
<th>body care</th>
<th>fly</th>
<th>forage</th>
<th>handle</th>
<th>sit</th>
<th>stand</th>
<th>walk</th>
<th>Totals predicted</th>
</tr>
</thead>
<tbody>
<tr>
<td>aggression</td>
<td>20</td>
<td>7</td>
<td>4</td>
<td>6</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>5</td>
<td>44</td>
</tr>
<tr>
<td>body care</td>
<td>12</td>
<td>61</td>
<td>0</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>5</td>
<td>2</td>
<td>86</td>
</tr>
<tr>
<td>fly</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>forage</td>
<td>10</td>
<td>8</td>
<td>3</td>
<td>204</td>
<td>42</td>
<td>16</td>
<td>6</td>
<td>14</td>
<td>303</td>
</tr>
<tr>
<td>handle</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>27</td>
<td>8</td>
<td>0</td>
<td>0</td>
<td>3</td>
<td>38</td>
</tr>
<tr>
<td>sit</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>76</td>
<td>2</td>
<td>0</td>
<td>79</td>
<td>79</td>
</tr>
<tr>
<td>stand</td>
<td>6</td>
<td>8</td>
<td>0</td>
<td>2</td>
<td>0</td>
<td>6</td>
<td>108</td>
<td>1</td>
<td>131</td>
</tr>
<tr>
<td>walk</td>
<td>1</td>
<td>1</td>
<td>6</td>
<td>6</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>14</td>
</tr>
<tr>
<td>Totals obs.</td>
<td>49</td>
<td>85</td>
<td>13</td>
<td>252</td>
<td>51</td>
<td>98</td>
<td>122</td>
<td>25</td>
<td>695</td>
</tr>
</tbody>
</table>

(c) Hidden Markov models, accuracy 68.6%.

Table 4.1: Results of the classification methods. Correct predictions shown in bold.
Chapter 4. Bridges Integrating Behavioural States

Figure 4.2: Computed densities and 99% level contours using different methods from the trajectory in (a). The 99% contour is drawn in red, while the 99% contour using all relocations is drawn in grey. The red shading indicates the density. We compare a constant diffusion coefficient against one that is based on the observed behaviours and use two different methods for estimating the diffusion coefficient. Horne refers to the maximum likelihood approach described by Horne et al. [24], new refers to the method described in Section 4.3.

<table>
<thead>
<tr>
<th>Method</th>
<th>Overall</th>
<th>Forage</th>
<th>Body c.</th>
<th>Stand</th>
<th>Handle</th>
<th>Aggr</th>
<th>Fly</th>
<th>Sit</th>
<th>Walk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Horne</td>
<td>32.5</td>
<td>0.366</td>
<td>0.571</td>
<td>0.0973</td>
<td>0.187</td>
<td>0.770</td>
<td>46.8</td>
<td>0.115</td>
<td>0.632</td>
</tr>
<tr>
<td>New</td>
<td>44.5</td>
<td>0.503</td>
<td>0.359</td>
<td>0.0521</td>
<td>0.312</td>
<td>1.03</td>
<td>1479</td>
<td>0.0046</td>
<td>1.11</td>
</tr>
</tbody>
</table>

Table 4.2: Estimated diffusion coefficients (in $m^2/s$) by the two methods. Overall refers to the most likely value over all data, without considering behaviours separately.

The accuracy is still quite low for all methods, but the time-dependent methods augment the underlying classifier. If a different classifier is used that performs better, we expect our methods to still improve the results. This however requires that the classifier produces a probability for each class, given an observation.

4.4.2 Trajectories with behaviour information

We investigate the quality of the BBMM augmented with behaviour information by investigating the likelihood of the model against some observations that were ignored while constructing the model. Figure 4.2a shows a trajectory that was collected from an Oystercatcher. The bird’s location was measured 80 times over a time span of approximately 25 minutes, and each location was tagged with the behaviour observed by an expert watching the bird.

What is difficult to see in the figure is that the bird spends a large fraction of time close to the first and last locations. The trajectory starts at the top, where the bird is foraging for 10 minutes. Then, the bird flies for almost a minute to the cluster of locations slightly lower. It stays there for six minutes and then it flies to the final location at the bottom over 4.5 minutes, where it stays the last three minutes of the observation period.
4.4. Oystercatcher case study

We have constructed utilization distributions by removing all location measurements, except the first and last. That is, the UD is computed from a single bridge with a duration of about 25 minutes, over which the movement speed of the animal varied greatly. When we use a single value of the diffusion coefficient, the results are similar in both methods of estimation (Figures 4.2b and 4.2c). The density is smeared out over the area between the observed starting and ending locations, which does not match well with the actual trajectory. Our new method of estimating the diffusion coefficient predicts a larger value than the method by Horne et al. [24] (see Table 4.2), resulting in a slightly wider 99% contour.

If we use the behaviour data from the intermediate observations, without using the locations, the results of both methods are radically different. Horne’s method predicts a very narrow bridge, since it predicts a rather low diffusion coefficient for the flying behaviour. The bridge is a bit wider where it expects the bird to make its stopover, but we see that this is positioned slightly too low. Most density is concentrated at the starting and ending location, since most time is spent there. The new method predicts a very wide bridge, since it predicts a higher diffusion coefficient for flying. That also means that the location of the stopover is less certain, but that is better than assigning a high density to the wrong location. The density is even more concentrated at the endpoints.

We make the above observations quantitative by investigating the likelihood of each model under the intermediate locations that were left out so far. The results are summarized in Figure 4.3. For both trajectories, using the behaviour data leads to a much higher likelihood compared to ignoring them. The new method for estimating the diffusion coefficient performs at least as good as the existing method and makes better predictions in most cases.

Thus, we have shown how the locations predicted by the BBMM can be improved using information on an animal’s behavioural state, which may in turn be derived from acceleration data. However, a larger study is required to make any strong claims about the quality of the predictions made by the approaches described in this chapter.

Figure 4.3: Log-likelihoods of the four models on two trajectories. 166_2 corresponds to the trajectory depicted in Figure 4.2.
In order to evaluate the results obtained in the previous chapters on movement data and to make our computational framework available to others, we implemented many of the methods described in the previous chapters in the R environment for statistical computing [35]. This environment is commonly used by ecologists to analyse movement data. We implemented these methods in the form of a package\(^1\), which is a collection of functions and objects that can easily be distributed to other users and installed in their R setup. The case studies that appear in this thesis were conducted using this package.

### 5.1 Class `bbtraj`

The package defines a few classes to hold trajectory data and data derived from that. The main class is `bbtraj`, which is based on the class `ltraj` in the package adehabitatLT [12]. An object of this class can hold multiple disjoint trajectories, which may or may not be from the same animal. One such trajectory is called a *burst*, while animals are identified by an *ID*. Multiple bursts may have the same ID, indicating that they contain data from the same animal, but the data was not collected continuously and quantities should not be interpolated at times between bursts.

A `bbtraj` object can be created from a table of locations, containing at least the times and measured locations. For the measurement error, one can either provide a separate value for each location or provide a single value that is reused for all. On construction, the diffusion coefficient is automatically estimated using the maximum likelihood method from [24]. The method described in Section 4.3 is implemented as well, but not used by default. Both methods produce the most likely value for each ID.

For compatibility with the Move package\(^2\), we intend to transition our package to use the classes from in that package, or new classes derived from those.

---


5.2 Functions

We have implemented various functions that operate on a trajectory of class \texttt{bbtraj}. Most functions use the BBMM, but there are some that assume linear motion between relocations. These are mainly used to compare certain features in the BBMM to the simpler linear model.

5.2.1 Basic properties

The package contains functions that return the values of the basic properties from Section 2.1. For example, there is a function named \texttt{position} which takes a \texttt{bbtraj} object and a list of times as input and returns the parameters for the position distribution of each animal at each of the requested times. There are also functions for computing the density, distribution function and quantiles of the distance between each pair of animals at a number of times.

We also have similar functions that compute quantities related to the distribution of velocity, speed and direction, but these operate on time intervals of finite length instead of points in time, since the instantaneous velocity of a Brownian motion is undefined.

5.2.2 Encounter

The package provides a function \texttt{encounter} to determine whether an encounter occurs at a number of specific times, in the linear model. A similar function for the BBMM is not necessary, as the probability of an encounter is given by evaluating the CDF of the distance at the threshold distance.

We include a function to compute the (expected) duration between all pairs of animals in both movement models. For the linear model, we can solve this analytically, for the BBMM the probability of encounter is numerically integrated over time. Alternatively, this function can provide the encounter duration between all pairs of bursts that have some overlap in time for their measurement intervals.

5.2.3 Spatial distributions

We provide functions that compute the spatial distribution of several important quantities. The most well-known is the utilization distribution (UD), that describes the space use of an animal. In addition we provide functions to compute where encounters are likely to occur and to determine the average speed at a specific location.

These functions are the most resource-intensive among all functions in the package, as they require the numerical integration of a quantity over time for every cell of a grid. The user can choose to sacrifice some accuracy for speed by allowing the functions to ignore regions with very low probabilities. The performance can also be tuned by setting the grid size and time step in the integration properly. We have also implemented the UD function in OpenCL to further speed up the computations on platforms that support this.

Furthermore, we have added a function that computes contour polygons from such a distribution. A contour polygon is the smallest (set of) polygons that contains at least a certain fraction of the probability mass of the distribution.

5.2.4 Plotting functions

The \texttt{bbtraj} class inherits its plotting functions from its parent class. One plotting function we provide is to easily generate bar plots comparing the duration of encounters, both between groups and on specific days.
5.3 Implementation details

Part of R is a programming language that is optimized for data analysis and related tasks. While this means that many common analysis tasks can be performed easily in R, for some recurring tasks that require more than one simple statement one would still want to define functions that make these tasks easy. These functions can themselves be implemented in R and in most cases this is indeed sufficient. However, since R is an interpreted language, loops with many iterations and a body consisting of many statements are not evaluated efficiently.

This problem occurs for example when computing a utilization distribution on a large grid. For every cell of the grid, the probability density of a normal distribution has to be evaluated many times to obtain the value of the UD in that cell. Luckily, R code can be easily interfaced to code written in other languages. We implemented the most resource-intensive parts of commonly used functions in C++ to speed up these functions.

Modern computers have many CPU cores and/or fast GPUs, which generally have a much higher instruction throughput than a single CPU. In order to take advantage of these devices, we have also implemented some functions in OpenCL [32], a standard for parallel programming on various hardware platforms.

We will use a simple example to demonstrate how this foreign code is interfaced to R. The example we will use is squaring a vector of real numbers. If \( x \) is a vector containing the numbers 1 through 10, squaring it in R is trivial:

\[
> x^2
\]

[1] 1 4 9 16 25 36 49 64 81 100

5.3.1 Interfacing C and C++ code

While most of the methods we needed can be implemented rather easily in R, this would be too slow to be useful in practice. Therefore we decided to implement some resource-intensive tasks in C++. We use mechanisms built into R that make interfacing to C and C++ code rather easy.

To implement our squaring example, we first implement a C++ function that performs this task:

```c++
void square(double *out, double *in, int *n) {
    for (int i = 0; i < *n; i++) {
        out[i] = in[i] * in[i];
    }
}
```

This function does not return anything and all parameters are passed as pointers. By convention, the first parameter contains the result. While we could have squared the array in place, this does not agree with the philosophy of R that functions should have no side effects, and thus we return the result in a separate array.

To call this function from our R code, we use:
> cResult <- .C("square", double(length(x)), as.double(x), +                  \[1\] 1 4 9 16 25 36 49 64 81 100

The function .C calls a C or C++ function with the name of the function in the first argument. Then, the arguments to the C function are listed. We create a new numeric vector of the proper length to store the result in, and we make sure that all arguments are casted to the proper type before calling the C++ function. Finally, the package argument states that the function can be found in a library that is distributed as part of package example. The call to .C returns a list of the values of all parameters passed to the C++ function, and we extract the first element of this list to obtain the result.

There is another interface to call C++ functions, named .Call. This allows complex objects in R to be passed to and from the C++ function. However, for most uses the .C interface suffices and is simpler to use.

5.3.2 Using OpenCL from R

While R does not support OpenCL by default, the OpenCL package\(^3\) provides an interface to OpenCL programs that frees the R developer from most of the memory management tasks at the expense of some flexibility.

To implement the example in OpenCL, we first define a kernel:

```c
__kernel void square(__global double *out, const int size, __global double *in) {
    int i = get_global_id(0);
    out[i] = in[i] * in[i];
}
```

This kernel is invoked once for every element in the array, each in a separate thread. In the first line, we request the ID of the current thread from the OpenCL runtime, and then we square the element at that ID. Since many threads can be executing at once, this is potentially much faster than the C implementation. Note that this code is far from optimal in terms of performance, but it suffices for the sake of this example.

Let `src` be a variable in R that holds the source code of the OpenCL kernel. Then, we execute the kernel from R:

```r
> library(OpenCL)
> kernel <- oclSimpleKernel(oclDevices()[[1]], "square", src, +                              precision=\"double\")
> oclRun(kernel, length(x), as.double(x))
[1] 1 4 9 16 25 36 49 64 81 100
```

We load the OpenCL package and then compile the kernel for the first OpenCL device that is detected on the machine. The kernel is shipped in source form, since different devices need different binaries, and this way it is compatible with all OpenCL-compliant devices. We have to specify that we want to use double precision numbers for the result, since by default `oclRun` allocates a vector of type `float`. We prefer double precision, because R always uses double precision for its floating point operations.

\(^3\)http://cran.r-project.org/web/packages/OpenCL/
5.3. Implementation details

Figure 5.1: Performance comparison between C and OpenCL implementations for utilizationDistribution. Grid size refers to the number of cells on the smallest dimension of the grid.

Note that in this call we do not provide an argument in which to store the result, but only its size. oclRun creates a vector of the appropriate size in the memory on the device and transfers it to main memory only after the kernel is executed. It then returns only this value. If the input vector is altered in the kernel, this will not be visible from the R perspective as it is only modified in device memory and discarded after the kernel finishes execution.

Our package detects what OpenCL devices are present when it is loaded and selects a default device to use. When a function with an OpenCL implementation is called, it will automatically run it on the selected device, or use the C implementation if no devices were detected. Thus, there is always a fallback for systems that do not support OpenCL.

To evaluate the performance gain using OpenCL, we consider the running time of the function that computes utilization distributions in OpenCL compared to a C++ implementation. We let it compute the utilization distributions of four groups of vervet monkeys over several months, using differently sized grids. The results are summarized in Figure 5.1. We see that using all four cores of the CPU through OpenCL consistently runs about four times as fast, so we get almost perfect scaling there. On the GPU there seems to be a startup overhead of about one second, but for larger instances it scales linearly with the problem size. For the largest grids, the speedup from using the GPU is about 120 compared to the C implementation.

---

4We used the following hardware in this comparison:

CPU: Intel Core i5-3570K; 4 cores at 3.40GHz.

GPU: AMD Radeon HD7950; 1792 stream processors at 950 MHz.
In this thesis we developed methods for algorithmic movement analysis using the Brownian bridge movement model (BBMM), with a focus on the analysis of animal trajectory data.

Strong previous work exists in algorithmic movement analysis, which we wanted to use as much as possible. We did so by developing a framework for detecting movement patterns in the context of the BBMM. Movement patterns composed from basic movement properties can be computed efficiently in this framework. Some movement patterns that we considered can not be composed in such a way, for instance, the regular visits and path-based following. For these patterns the formulation in the BBMM resulted in interesting algorithmic questions of their own right.

To make our algorithmic results useful to users of animal trajectory data, in particular ecologists, we had to make a link to concepts used in ecology. We did so by showing how to integrate our framework with the movement ecology paradigm. We presented two approaches to integrate external factors into our framework, depending on the nature of the available data. We have also developed ways to refine the predictions of the BBMM when information on the behavioural state of the animal is available.

While developing these refinements, we have obtained new results on estimating the parameters for the BBMM and on integrating time-dependencies into classification algorithms. Both results led to improved predictions in our case studies.

For the theory developed in this thesis to be truly useful, we have to provide tools to ecologists that allow them to perform this analysis in an environment that they are familiar with. Our R package provides this functionality and is optimized to perform the most important operations fast by utilizing parallelism.

We have demonstrated the usefulness of this package in several case studies. Our study on groups of vervet monkeys shows that the BBMM can detect interesting patterns that are not detected by the linear model. Instead of relying on the assumption that movement is linear between samples and the location known at all times, the BBMM can detect situations where the condition is likely to be satisfied. For the simultaneous movement data of the case study, the encounter pattern in combination with the BBMM proved a powerful tool for studying movement interaction.

The study on bee-eaters and the second study on vervet monkeys show that we can compute spatial distributions of the speed of an animal, which cannot be derived in the linear model. We have shown that the BBMM makes a clear distinction between the two flight modes of bee-eaters in terms of the estimated diffusion coefficient.

Finally, we demonstrated that the use of behaviour data can increase the quality of the predictions made by the BBMM. A new method for estimating the diffusion coefficient may further improve
these results. Combining trajectory segmentation with classification leads to improved accuracy in determining the behavioural state from acceleration data.

With this thesis, we have put in place many of the fundamentals for algorithmic movement analysis using the BBMM. However, we believe this is only a first step toward this goal. We have considered how to compute movement patterns in this new setting by giving examples for specific patterns. We expect more interesting research questions to arise when further movement patterns are (re-)formulated in our framework. Similarly, in the integration of the movement ecology paradigm we have considered several external factors and ways to include behavioural states, but there are many more such factors that can be integrated. One example would be including information about the environment in which an animal moves a priori, which likely requires that trajectories are simulated efficiently. Deriving behavioural information from acceleration and location data is an open challenge in which many opportunities for future work exist. We have demonstrated links between classification and segmentation of movement data, which would be interesting to be explored further.

All these results apply to the BBMM. It would be interesting to see if these results can be adapted to other movement models, which may be more suitable in certain situations. We have focused on the computational analysis of movement data, but in order to optimally extract information from these results, interactive visualizations are needed.

For new theoretical results, it is interesting to evaluate these in practice and to make them available to ecologists. Therefore, development of the R package should be continued as these new results become available. We believe this thesis shows that developing algorithms in close collaboration with expert users and directly providing the corresponding tools is a powerful approach to computational movement analysis.


