Interpretable and Comparable Measurement for Computational Ethology

Interpretable and Comparable Measurement for Computational Ethology

By Konrad Urban

Adviser: Venkatesh N. Murthy

Thesis Advisor: Venkatesh N. Murthy Konrad Urban

Interpretable and Comparable Measurement for Computational Ethology

Abstract

Computational ethology, the field of automated behavioral analysis, has introduced a number of new techniques in recent years, including supervised, unsupervised, and statistical approaches for measuring behavior. These approaches have made the measurement of behavior more precise and efficient, improving significantly on human annotation of behavior. Despite this improvement, however, researchers in the field often interpret and report results primarily with reference to traditional, linguistic categories of behavior, which reduces measurement precision. I argue that using interpretable measurement techniques can help researchers to overcome this imprecision. Further, I present *Basin*, a system for performing interpretable behavioral measurement. Pioneering work has recently developed deep learning systems which can accurately estimate the pose of animals in videos. Basin builds on this work by creating a system in which researchers can construct behavioral measurements which are functions of an animal's pose. Specifically, Basin provides a graphical interface in which users can compose mathematical operations to construct behavioral measurements, without writing any code. Additionally, Basin provides a variety of features to visualize and refine behavioral measurements. I demonstrate that Basin can be used to replicate results from past studies of behavior and further show that Basin can be used to analyze the manifold structure of ant walking behavior.

Contents

Abstract **ii**

Acknowledgments **v**

1 Principles of Computational Ethology **1**

1.1 A Summary of Computational Ethology 1

1.1.1 The Tools of Computational Ethology 2

1.2 The Process of Measurement in Computational Ethology 5

1.2.1 The Complexities of Model-Based Measurement 6

1.3 Principles of Measurement in Computational Ethology 10

1.3.1 Interpretability 10

1.3.2 Comparability 12

2 The Design of Basin **15**

2.1 Introducing *Basin* 15

2.1.1 The User Experience of Basin 15

2.1.2 Measuring Behavior in Basin 17

2.1.3 The Generality of Basin Definitions 21

2.2 Basin is Interpretable and Comparable 22

3 Measuring Behavior With Basin **25**

3.1 Analysis of Mouse Behavior 25

3.2 Analysis of Ant Behavior 28

3.2.1 Replicating Measurements of Trail-Following Behavior 28

3.2.2 Manifold Structure of Ant Locomotion 32

4 The Architecture of Basin 39

4.1 The Software Systems of Basin 39

4.1.1 Front-End 39

4.1.2 Inter-Process Communication 39

4.1.3 Back-End and Storage 42

References 45

A Definitions of Primitive Operations 49

Acknowledgments

I am deeply grateful to my advisor, Professor Venki Murthy, for his incredible support over the past few years of research. His lab is an incredibly supportive environment, and his commitment to the growth of his students is unparalleled. His patience and encouragement were absolutely integral in the creation of the work presented in this thesis, which is the product of a long period of at-times circuitous development.

I am also very much indebted to my advisor, Alexander Mathis. Alexander first introduced me to automated behavioral analysis and familiarized me with the literature of automated behavioral analysis. Alexander provided me with significant guidance every step of the way, and the insights I gained from my conversations with him are deeply woven into this thesis. I am very grateful to have had his support for over two years.

I would also like to thank my roommates and friends for their support and sympathy through the inevitable, frustrating roadblocks in the course of this research. They were very kind to listen to my troubles and share in my triumphs, even when those triumphs meant little to them.

Finally, I would like to thank my parents. It would be impossible to fully state their impact on this work, so I would just like to make it clear that without them, none of this would have been possible. Thank you.

Chapter 1

Principles of Computational Ethology

The field of computational ethology, the study of behavior with computational techniques, has advanced rapidly in the past several years. The aim of this opening chapter is first to review major themes and techniques within this field, to ground the reader in the quickly moving literature. Despite the field's rapid growth, however, some commentators have suggested that important philosophical questions in the field remain unanswered (4). The primary aim of this thesis is to introduce a system whose development is in part a reaction to trends in computational ethology. Thus, in addition to providing background, this chapter argues for a particular way of thinking about computational ethology, a framework whose principles guide the design of Basin, a tool of computational ethology which is the subject of this thesis. I will argue why this shift in conception is important, and I will show how this shift informs the development of principles by which the study of computational ethology can be conducted.

1.1 A Summary of Computational Ethology

The term "computational ethology" was coined by Anderson and Perona in 2014 to characterize an emergent field of behavioral analysis (3). The authors describe the field as follows:

> "We argue here that a new interdisciplinary field ... [has] the potential to revolutionize the way in which we measure and model behavior. We call this field 'computational ethology' (CE), to emphasize

> both its roots in the study of natural behavior in freely moving animals and the application of modern quantitative tools for measuring, describing, and analyzing behavior." (3)

Here, the field is construed as containing research activity at the intersection of two other fields: computer science/statistics, and ethology. This naturally means that the set of research which can be classified as computational ethology is relatively broad. The term "modern quantitative tools" is broad enough to permit a diverse set of analysis with different underlying methodological bases; consider, for example unsupervised and supervised machine learning. In addition, the "study of natural behavior," ethology, is a huge field which has existed for decades (35).

Part of the reason the definition is broad is that computational ethology is a young field. One of the reasons for its emergence is the recent advancement in machine vision. While there exist many modalities through which behavior can be analyzed, vision (and thus, video) remains by far the most prevalent. In large part, this prevalence is due to the fact that it is cheap to collect and it provides quite detailed information about the underlying behavior. Thus, many developments in computational ethology have been fueled by the ability to extract increasingly detailed information from video. For example, behavioral classification through supervised learning was made possible in part by advancements in the ability to extract visual features from video, with which supervised learning algorithms could be trained (20). More recently, tools of computational ethology have started to rely on sophisticated deep learning methods for extracting pose directly from videos (26; 22); such methods extract arguably the most informative low-dimensional features from video.

1.1.1 The Tools of Computational Ethology

The tools used in computational ethology can vary widely in technique and purpose, so to gain some sense of the field, it is helpful to divide these tools into loosely defined categories. First, some tools are intended to classify data collected about behaving animals (e.g. video frames) into behavioral states which are defined by humans. Second, some tools leverage unsupervised or statistical learning to implicitly define behavioral classes identified when the learning technique is applied to the data. I refer to the first category as "explicit behavioral classification", and the second as "implicit behavioral classification". In what follows, I examine the development of some important tools within each category.

Explicit Behavioral Classification

One of the earliest systems for automated behavioral analysis was EthoVision, a tool first introduced by Noldus Information Technology in 1993 (33). EthoVision was intended for the behavioral analysis of mice, and featured tracking and behavioral classification capabilities, including the ability to classify a few basic behaviors such as moving and rearing. In 2013, a group at Janelia introduced the Janelia Automatic Animal Behavior Annotator (JAABA), a system for supervised classification of behavior (20). JAABA provides users with a framework for viewing videos of behaving animals, labeling video frames with behavior classes, and then running a supervised learning algorithm to learn how to classify future video frames based on the user-created labels.

Even more recently, a system for multi-mouse tracking system called LiveMouseTracker (LMT) was developed (14). One component of LMT is a set of behavioral classifiers which can classify both individual and social mouse behaviors. These classifiers work by fitting ellipses to the mouse bodies, and then performing classification using hand-written functions which take as input parameters of these ellipses.

Implicit Behavioral Classification

A second class of behavioral definitions are derived either by clustering video frames according to some measure of distance, or by fitting a statistical model to behavior and then interrogating the statistical model for information about behavior. Each approach is highlighted below.

Unsupervised Definitions

One of the first examples of dimensionality reduction applied to behavioral analysis was (34), a paper which showed that much of the variation in *C. elegans* movement could be explained by considering only four fundamental behavioral states, through which the worm transitions over time. This paper was quite influential because it introduced the idea that animal behavior could be explained more successfully by considering a lower-dimensional version of the behavior. The development of these approaches continued in (6), a paper in which the authors perform unsupervised learning on fly behavioral data. In this paper, the authors start from raw video frames, reduce the dimension of the frames using principle components analysis, apply wavelet transformations to extract temporal information about periodic movement, and then finally apply the t-Distributed Stochastic Neighbors Embedding (t-SNE) algorithm to the transformed features to project the data into two dimensions (21). t-SNE's primary aim is to preserve local structure in the original data within the lower-dimensional representation, and thus clusters of related points often naturally appear in the lower dimensional manifold (21). The authors of (6) then cluster the two-dimensional manifold using a watershed algorithm, and find that clusters within the lower-dimensional data correspond to visually distinct fly behaviors, including locomotion and courtship behaviors.

The work described in (6) was influential, in that it produced a generic four-step process for behavioral analysis, the steps of which include: 1) reduce dimensionality of the original video frames, 2) extract temporal information via a wavelet convolution, 3) project data into a lower dimension, and 4) cluster points in the lower-dimensional space. Since that paper, researchers have investigated variations on the specific algorithms used. For example, in (36), the authors systematically exchange various stages of the algorithm, including using PCA instead of t-SNE, and applying a Gaussian mixture model instead of using the watershed algorithm to identify clusters. Another development, mentioned above, which has further aided unsupervised classification of behavior is the construction of deep neural networks aimed at pose extraction (22). These networks use transfer learning to reliably estimate the pose of animals in unseen videos, while only needing to be trained on a few hundred environment-specific frames of hand-labeled pose information. Recently, these networks have been leveraged for unsupervised behavioral classification (19; 26). For example, in (19), researchers apply t-SNE directly to the pose extracted by a deep neural network and then cluster behavioral states using a Gaussian mixture model.

Model-based Definitions

A final approach under the umbrella of implicitly-defined behavioral classifiers is to fit statistical or deep learning models directly to the behavioral data, and then infer behavioral states by probing aspects of the statistical model. One example of this work is (37), which fits an auto-regressive Hidden Markov Model (AR-HMM) to behavioral data of behaving mice. The hidden states identified by the AR-HMM are then taken to be the behavioral definitions, which can be then be visually inspected and interpreted. To test the efficacy of their classification method, the authors manipulate the animals through the addition of olfactory cues and the manipulation of relevant neural circuitry, and then observe changes in the distribution of the AR-HMM modules. Another example is (15). Here, the authors fit a recurrent neural network to features extracted about a fly's movement, and the network is then trained to classify user-specified behaviors, as well as generate upcoming frame data about fly movement. The authors find that they are able to generate qualitatively similar behavior from their network.

1.2 The Process of Measurement in Computational Ethology

The previous section illustrated a broad range of approaches within computational ethology. However, as I will argue here, while researchers have applied a variety computational tools to ethology, the scientific goals of the field have remained less clear, which has complicated the field's development. Thus, to start this section, I give the definition of computational ethology which guides this body of work: *computational ethology is a field whose goal is to construct computationally-driven theories of behavior*. My definition builds on Anderson and Perona's view that computational ethology aims to use computational tools for "measuring, describing, and analyzing behavior," but crucially emphasizes that no matter the tools used or the analysis done, the aim of the field remains to create new knowledge about behavior through theory development.

This definition is significant because if the aim of a field is theory development, then the field needs data to support or refute proposed theories. Data is collected by measuring some properties of the subject of study. Thus, measurement plays a crucial role in theory development. Often, when researchers discuss the field of computational ethology, they speak of behavioral *classification*, which to some extent obscures the fact that measurement need not be in the form of binary, or even discrete behavioral classifications. Thus, to open the door for richer forms of analysis, I often use the term *measurement* rather than classification. Using the term classification also suggests an emphasis on supervised machine learning techniques, but as seen above, there are many ways to measure behavior, most of which are not supervised machine learning techniques.

The goal of this section is to discuss the downsides and limitations of certain techniques for behavioral measurement. The basic argument is that if the goal of computational ethology is theory development, then measurements of behavior in computational ethology must be interpretable by humans. This emphasis on intepretability is one of the guiding principles of the system presented in this thesis.

1.2.1 The Complexities of Model-Based Measurement

In the summary of the work in computational ethology above, I discussed examples of so-called “implicit behavioral classification”; i.e., unsupervised and statistical tools of behavioral analysis which construct a set of behavioral states through the model itself, without being defined apriori by humans. In many ways, this strategy for defining behavioral states seems better than labeling performed by humans. This point is made well by a number of people, including the authors of (37), the paper described above which presents the AR-HMM to model mouse behavior, work which falls under the umbrella of implicit behavioral classification:

> The “unsupervised” modeling approach taken here is also transparent ... This makes clear the precise bounds of human influence on the output of the AR-HMM and insulates key aspects of that output from the vagaries of human perception and intuition ... developing methods free from observer bias is essential for informatively characterizing mouse behavior (37).

The authors make the very important point that human labeling is unreliable, a finding that is backed up by others, who find significant disagreement between human annotators (17; 3). However, I claim that despite its success in certain respects, model-based, implicit behavioral measurement is actually most often not free from human bias, because results from this analysis are still reported through the imprecise vocabulary of human behavioral categorization.

Model-Based Measurements are Poorly Understood

When behavioral modules are identified through implicit classification methods, one of the primary strategies for analyzing and communicating the results is to view the video frames which are mapped to each behavioral state, and then report the behavior observed in these frames as that “discovered” by the supervised or statistical model. Indeed, the authors of (37) describe “visual inspection of the 3D movies assigned to different modules reveals that each encoded a coherent pattern of 3D motion that post hoc can be distinguished and labeled with descriptors” and the authors of (6) (the paper which used t-SNE embeddings of fly behavior) report “Observing segments of the original movies corresponding to pauses in one of the regions, we consistently observe the flies performing a distinct action that corresponds to a recognizable behaviour when viewed by eye”. Each paper dedicates significant portions of the results to interpreting the results by commenting on the

the labeled behavioral states identified.

However, when behavioral states defined by algorithms are visually characterized by humans, several issues with precision arise. First, the relationship between the human understanding of a behavioral state and the algorithmic definition of a behavioral state is usually ill-defined. Assume for the sake of argument that an individual has some conception of how a behavior should be defined. We can approximate this conception with a function $F : X \to \{0, 1\}$, where $X \subseteq \mathbb{R}^{\ell \times w \times n}$ is the set of sequences of video frames, and $F(x)$ for $x \in X$ classifies the sequence of frames as containing or not containing some behavior. Let $B \subseteq X$ be the set of frames which are considered to be illustrating a particular behavior state by a model. Note that in most cases, the function which classifies frame sequences as belonging to B is quite complex, and uninterpretable to humans (e.g. the inference algorithm for hidden state classification in HMMs in (37), or the dimensionality and clustering algorithms for t-SNE in (6)).

Let's assume that through some analysis, humans decide that the set of frames classified as belonging to behavioral state B correspond to the behavior defined by F. The ideal correspondence would be that $F(x) = 1$ if and only if $x \in B$. However, this would require intensive checking. First, all frames in B must be checked for some behavior so that $F(b) = 1$ for all $b \in B$. However, even with this analysis, there is no guarantee that there doesn't exist some sequence of frames $b' \in X$ such that $b' \notin B$ but still $F(b') = 1$. The only solution to ensuring this second condition is to literally check all frames. In reality, given the huge number of frames classified, neither side of the if and only if can be checked, so existing studies provide no guarantees regarding the relationship of B and F[1].

This is an important problem. For example, consider a scenario in which a study reports a result such as "behavior X, which corresponds to behavioral state Y (identified by my algorithm), is altered from condition A to condition B." Since there are in fact no guarantees that the human conception of "behavior X" is in a one-to-one relationship with "behavior state Y", a variety of misunderstandings could result from this report. For instance, if a researcher claims "behavioral state Y" defines escape behavior, it may in fact define only locomotion (an oversight which resulted because the researcher didn't look at frames with other types of locomotion which were also classified in state Y). Thus, the interpretation of the behavioral state may shift dramatically from being a fear response to simply being a general activity

[1]Of course, it may be possible to obtain high statistical confidence about an iff by sampling some frames and manually checking the frames sampled. This approach however, has not been investigated thus far to my knowledge.

state. Conversely, the researcher could claim that "behavioral state Y" defines locomotion, when in fact it only defines escape behavior. This oversight could occur if the researcher fails to look at all frames classified outside of behavioral state Y, one or more of which may include locomotion behaviors. Again, the interpretation would shift dramatically.

Model-Based and Supervised Measurements Cannot Be Precisely Communicated

Even if a one-to-one correspondence between a model-identified behavioral state B and an internal behavioral classification function F could be established, model-based measurements still suffer from fundamental issues related to communication and reporting. Specifically, different people tend to have quite varying perceptions of what constitutes a specific behavior. The authors of (17) report that the concordance[2], a measure of similarity between multiple observers' classifications of behavior, is generally only between 50% and 70%. The authors of (3) also suggest a similar number.

Thus, even if a researcher establishes a one-to-one relationship between behavioral state B and her internal behavioral classification function F for behavior A, when she reports results about behavior A, other researchers likely have a very different conception of behavior A, and thus a different classification function F'. From the above studies, $F(x) = F'(x)$ only about 50-70% of the time. Thus when researchers claim that unsupervised or model-based classifications cut out human variability in defining behavior, but then report behavioral classifications by assigning labels to the behavioral states their algorithms identify, they are in fact returning to the very same subjectivity which they sought to avoid. To drive the point home, consider if physicists measured the movement of falling objects by subjectively considering whether they "fall quickly" or "fall slowly". Indeed, even if physicists had a sophisticated algorithm to partition objects into "falling quickly" or "falling slowly", if they reported their results still using the human terms of "falling quickly" or "falling slowly", the precision of their results would be greatly hindered, due to the reliance on imprecise and subjective human language.

One could argue that the problem then is reporting behavioral classifications at all using language; namely, that we might be able to make advances in ethology without ever reporting human-interpretable information about the behavior we find. Indeed, in some cases, implicitly defined behavioral classifications may be used in

[2]Concordance is measured as $\frac{\text{agreements}}{\text{agreements+disagreements}}$ (see (17))

ways that largely ignore the content of each classification (see (5) for work that de-emphasizes providing human-interpretable labels to behavioral states).

However, if we accept that the aim of computational ethology is theory development, then it is important to remember that theory development is a human-driven enterprise. Understanding the instruments of measurement has always been an essential skill for researchers in any field, so it would require a radical departure from the traditional scientific method to utilize measurements which were collected in a mostly opaque manner.

Model-Based Measurements are Often Not Comparable

Earlier, I discussed how reporting results using the labels of behavior found in human language can be problematic, due to the imprecision of the labels of behavior found in language. Another way of framing this problem is as a problem of comparability: namely, that if two researchers perform separate behavioral analyses, even if they report results about the same behavior linguistically speaking, they may in fact be referring to two very different measurement functions. One could suggest, however, that this issue can be easily resolved simply by using the same behavioral classification function across datasets. In this scenario, one researcher would train her behavioral classification function, and then share it with another researcher, who could then use it on his own dataset, without worrying about whether the classification function is the same. In this case, comparisons could perhaps be made without ever applying a linguistic label to the behavior[3]. Although this is theoretically a sound approach, in practice, the algorithms used do not readily permit this possibility.

There are two ways in which this approach could practically manifest. The first relies on the fact that when a model is trained, it will inherently contain some method for behavioral state classification. In the case of an AR-HMM, this is an algorithm for hidden-state inference; in the case of dimensionality reduction, such as with t-SNE, this is the mapping from the higher-dimension representation to the lower dimensionality representation. Call this function $G : X \to \{0, 1\}$. In theory, this function G could be used across multiple datasets to classify behavior, thus allowing researchers to compare results without having to assign labels to the behavioral states. However, this approach suffers from an important problem: G is only valid on the distribution of frames on which it was trained. Thus, if G is

[3]Although even if this could work, I would still argue that the approach hinders effective theory development, for the reasons I give above.

trained on a dataset D drawn from distribution P, a probability distribution on frames, if G is then applied to D' drawn from distribution P', where $P' \neq P$, then G applied to D' is unlikely to be comparable in any important sense to G applied to D. For example, changes in background, animal color, camera location, and many other things irrelevant to behavior could generate drastically different behavioral classifications, for entirely spurious reasons.

The second approach could be to train a new model M' on the new dataset D', and then compare the classifications generated by classification function G' (for model M') and G (for model M, trained on dataset D). Unfortunately, even this approach runs into serious problems, the most important of which is that for the algorithms typically used, there is no guarantee at all that a retrained algorithm will converge to the same behavioral states. Without such a guarantee of mutual convergence, it is difficult to do a comparison using this approach.

1.3 Principles of Measurement in Computational Ethology

While the above discussion focused on the pitfalls of various techniques for measuring behavior, this section focuses on best practices, and outlines possible approaches for developing effective measurements of behavior. Specifically, I highlight two important principles: interpretability and repeatability. These two principles form the basis for the design of the system which is the primary subject of this thesis.

1.3.1 Interpretability

Parsimonious Measurement Functions

While interpretability is a nebulous concept to some degree, recent work has begun to study it in more depth. For example, the authors of (10) provide an important definition of interpretability: “the ability to explain or to present in understandable terms to a human.” While traditional labels of behavior (“running”, “rearing”, etc.) may qualify for this definition, they are imprecise. So instead, I propose emphasizing that behavioral classification functions themselves should be interpretable. In general, classification functions which are defined through complex statistical or unsupervised models are too complex to be understood by humans. Thus, emphasizing parsimony in the construction of behavioral classification functions may be one strategy to ensure that such classification functions are interpretable.

Parsimony has a long history in the statistics literature, and is often invoked in reference to principles of model selection such as the Akaike and Bayesian Information Criteria, or the Minimum Description Length principle (16). While parsimony has a very strong theoretical basis for successful model selection (31; 18), it also has a more practical value: parsimonious models are often easier to understand and interpret.

Parsimonious behavioral classification functions serve two helpful roles: first, they enable researchers to think more clearly and rigorously about what is actually being measured, and second, they enable clearer and richer communication between researchers about the behavior being measured, since simple behavioral classification functions can actually be written down and communicated in writing. While it may not be immediately obvious which functions are "simple", the concept of Kolmogorov complexity, the idea which inspired the MDL principle (31), may provide a natural guiding principle. Put simply, the Kolmogorov complexity of a function is the length of the shortest computer program which computes that function. Unfortunately, Kolmogorov complexity is uncomputable (31), but again, such a theoretical principle may guide the development of classification methods.

The literature has several examples of approaches which achieve parsimonious classification functions. One example is (14), a system described above which tracks multiple mice, and defines behaviors based on the movements and posture of the mice. Specifically, given the underlying posture and positions of the mouse, the authors define a number of simple, parsimonious algorithms to classify individual and social behaviors. Because these algorithms are defined explicitly by researchers, based on behaviors which they desire to analyze, the algorithms themselves are readily interpretable, and can thus be communicated to others. Thus, there is no ambiguity regarding exactly how the behavioral classification operates. In some sense, the work presented here can be viewed as a generalization of this approach to behavioral analysis.

A slightly different example is (9), a study which applied the UMAP dimensionality reduction technique to fly walking data. While the UMAP dimensionality reduction function is uninterpretable, the authors perform post-hoc mathematical analysis to carefully understand the structure of the lower-dimensional manifold which the walking data is projected onto. Thus, by characterizing the structure of the manifold mathematically, they make the manifold interpretable in a precise manner, since the mathematics describing the manifold can be readily understood. Thus, the authors effectively turn a complex dimensionality reduction function into a parsimonious mathematical description of the variation in fly walking.

1.3.2 Comparability

I discussed above how some model-based behavioral definitions may be incomparable. Here, I discuss best practices to ensure behavioral classifications can be compared between experimental conditions. In my view, comparability requires at least two elements, outlined below: invariance to recording environment, and invariance to irrelevant organism variation.

Invariance to Recording Environment

One of they key aspects of scientific communication is the ability to compare the results different groups find, to either verify the truth of some mutually found fact or to better understand when discrepancies occur. As discussed above, when models are trained on raw video frames, their classification functions can only provide valid classifications within the particular distribution of images on which they were trained. Thus, a trained model would likely fail on behavior performed in a different environment, even if such a change were as trivial as changing the background color of the environment.

While this problem may seem intractable because any algorithm must start from the raw video frame, it is in fact not as bad as it first seems. Advances in deep learning highlighted earlier allow for fast and highly accurate pose extraction (22), which enables behavioral classification algorithms to be applied directly to pose (19). Even when the recording environment changes, pose is invariant, and so behavioral classification techniques which start from pose can be applied to arbitrarily many different environments - the only difference is that a new pose extraction model must be learned. The high accuracy of trained pose extraction models (22) make this a much more robust technique for generalizing between environments. Thus, one key aspect of comparability is invariance to the recording environment, which can be achieved in part, for example, through the extraction of invariants such as pose. Once behavioral classification functions are created which only rely on pose, they can be more easily shared between researchers.

Invariance to Irrelevant Animal Features

However, even pose extraction is not a panacea, because the skeletal structure of animals in the same species is often different. For example, different mice may be different sizes, and different ant classes (e.g. worker versus soldier ants) may have very different body types. Clearly, having a different body morphology may affect behavior. The question of whether behavioral differences which are due in part to variation in body morphology are significant behavioral differences is complex, and likely dependent on the analysis one wishes to perform. Even a behavior as simple as walking reveals the complexity of this problem. On the one hand, we commonly define walking independently of body size; for example, we don't classify the behavior of tall and short people different when each are walking, despite differences in the gait of each. However, we may say that tall or short people are performing a different behavior when the taller person is able to walk over large objects, whereas a smaller person is forced to climb over the objects to get to the other side.

The difficulty of quantifying inter-subject behavior becomes immediately apparent when we consider even simple behaviors. Taking up the walking example again, if we define walking versus running on the basis of the speed of a foot, this definition may yield very different results for tall versus short people since, to walk at the same speed, shorter people must move their feet more quickly to complete more gait cycles. Thus, for this scenario, the most relevant definition may not be based on an absolute walking/running speed threshold, but rather a percentile, calculated per subject, which defines the difference between walking and running.

The examples serve to show that even when difficulties of environmental variation are factored out, inter-animal variation remains a relevant challenge for behavioral analysis. While the solution to this problem will inevitably be specific to the type of behavioral analysis desired, it remains quite important to consider this definitional issue when creating tools of computational ethology. Also, this concern further reinforces the importance of parsimonious models, in which the complexities of these issues can be much more readily understood.

The aim of this thesis is to introduce a system for behavioral measurement, Basin, which I created. Basin enables researchers to construct interpretable, comparable behavioral measurement functions for a variety of behaviors and organisms. In chapter two, I discuss how Basin works, and then in chapter three, I show several examples of behavioral analyses of ant and mouse data, which were conducted with Basin. Finally, in chapter four, I discuss the architecture of the Basin software more closely. Chapter four is aimed at readers with a technical interest in the software constructs of Basin, and can be safely skipped by most readers who only desire an understanding of what Basin is and how it can be used.

Chapter 2

The Design of Basin

2.1 Introducing *Basin*

Basin (an abbreviation of "behaviors-insight") is a software platform for measuring behavior which I created and which is the subject of this thesis. This chapter will seek to explain what Basin is and how it is used, and then justify the design of Basin by applying the principles described in the preceding section.

2.1.1 The User Experience of Basin

Figure 2.1 provides an overview of how Basin works. Basin builds upon the work of Alexander Mathis and colleagues, who developed a system called DeepLabCut (22), which leverages deep learning to perform accurate automatic pose estimation from videos. Basin uses estimated pose, and optionally uses segmentation masks which correspond to regions of the environment, to enable users to measure behavior by defining measurements over pose and regions.

Analysis Mode

Basin has two modes of operation, the first of which is the analysis mode. Once users have imported pose measurements and corresponding videos into the system, they are then able to define behavioral measurements, through a process which will be explained in more detail below (this process is shown in figure 2.1(2a)). Then, users are able to visualize behavioral measurements in real-time as the video of the behaving animal plays, to confirm the efficacy of their behavioral measurements

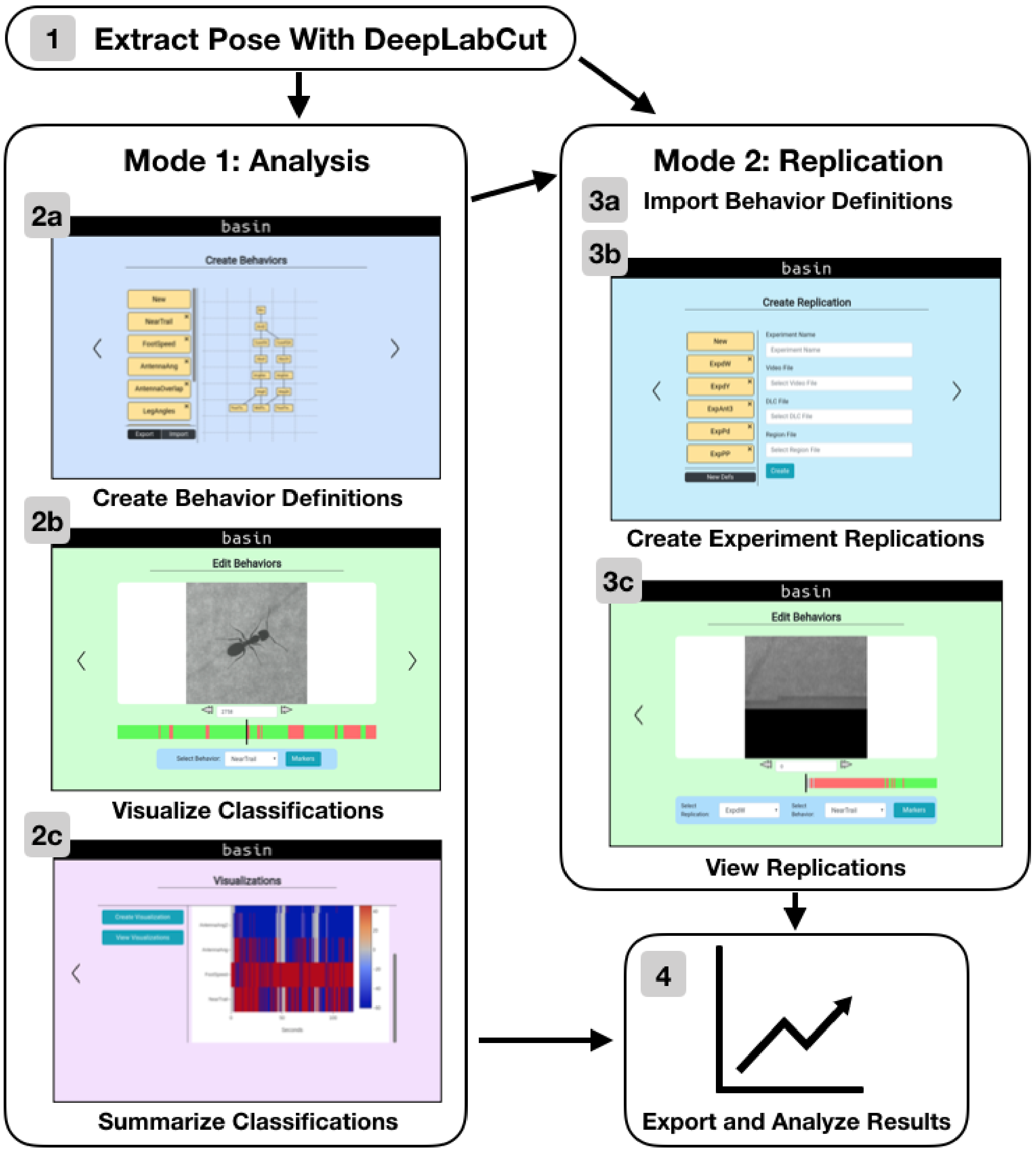
1
Extract Pose With DeepLabCut
Mode 1: Analysis
2a
basin
Create Behaviors
Create Behavior Definitions
2b
basin
Edit Behaviors
Visualize Classifications
2c
basin
Visualizations
Summarize Classifications
Mode 2: Replication
3a
Import Behavior Definitions
3b
basin
Create Replication
Create Experiment Replications
3c
basin
Edit Behaviors
View Replications
4
Export and Analyze Results

Figure 2.1: The user view of Basin. Typical order of steps followed in Basin analysis are indicated by numerical and alphabetical ordering.

(fig. 2.1(2b)). Often, users will realize their behavior definitions are incomplete, in which case they will return to the behavior definition view to edit the way the measurements are computed. Finally, users can visualize whole-video summaries of the behavioral classifications in the form of ethograms, a standard method of depicting behavior measurements (fig 2.1(2c)).

Replication Mode

Once users have defined behavioral measurements, they can then export the definitions of the measurements they have created from the analysis mode. The exported definitions (which are in a simple JSON format) can be then used by a single researcher to replicate their measurements on one video across other videos, or they can be sent to other researchers, who can use them to replicate results on their own data. Basin provides a separate set of views for replicating results, which I refer to as the "replication" mode of Basin. Once a behavior definition file has been imported into Basin, users then create experiment replications by importing DeepLabCut pose estimates, along with the corresponding video file, a process shown in 2.1(3b). Then, users can once again view the classifications of behavior made by Basin in real time, for any video on which the replication has been performed 2.1(3c).

Finally, once behavior has been classified in videos, the classifications can be exported and analyzed (fig. 2.1(4)). The extraction of the classifications is done through a Python library written to interface with the databases which Basin creates during analysis.

2.1.2 Measuring Behavior in Basin

Modular Measurement

Basin measures behavior using pose estimates. Here, I define pose simply as a collection of body points. Different behaviors are often best measured using different body points, and Basin is agnostic to which body points are used. Deep learning pose estimation systems such as DeepLabCut can be trained to estimate a variety of different body points across many organisms, so researchers can flexibly extract the body points which are most relevant for the behavior they desire to measure. These body points can then be input into Basin as the pose of the animal.

The crux of Basin's efficacy is its system for measuring behavior. Basin simplifies the process of constructing measurements by providing an interface in which users can compose primitive mathematical operations to construct behavioral measurements. To see how this works, consider three simple operations: subtraction of two vectors, taking the norm of a vector, and the operation which calculates the angle between two vectors. Now, consider composing these functions, possibly multiple times, over the points which constitute the pose of an animal. Together, these three operations can produce a variety of interesting behavioral measurements: the angle of a mouse's head relative to its body; the distance between two mice; the expansion or contraction of a mouse's spine; the distance from an ant's antenna to it's head; the angle of an ant's leg relative to it's spine.

Thus, it is apparent that even with three relatively basic mathematical operations, a wide variety of behavioral measurements can be defined for two very different animals. Basin extends this idea further by implementing over twenty of these operations in code (these operations are enumerated in appendix A). Then, Basin provides a graphical interface in which users can visually construct behavioral measurements by composing the operations, each of which takes as input pose estimates or the output of other operations. Thus, Basin separates the implementation of primitive operations (such as angle and distance calculations) in code from the composition and application of these functions over pose, which is done through an abstract graphical interface.

Because Basin implements over twenty of these operations, it provides a rich "vocabulary" through which users can construct behavioral measurements. Users often do not need to write any code within the graphical interface of Basin, since these primitive operations are already implemented. However, the utility of Basin is not limited to simply avoiding using programming for measuring behavior. While some of the primitive operations are relatively simple, some are a bit more complex (such as numerical derivatives, projection operators, and especially the region-based operators discussed below), and so it is helpful, even for researchers with programming ability, to define measurements in Basin to simplify the construction of behavioral measurements. Further, Basin contains a type system which automatically ensures that researchers can only compose functions with appropriate type matches. Additionally, Basin automatically deals with circumstances in which DeepLabCut fails to detect animal body points with a high probability (perhaps due to occlusion or something similar) by altering the output of the measurement, an important detail for maintaining measurement validity. Finally, Basin closely integrates its system for behavioral measurement with the systems for visualizing measurement described above, enabling researchers to quickly verify the efficacy of their measurement definitions.

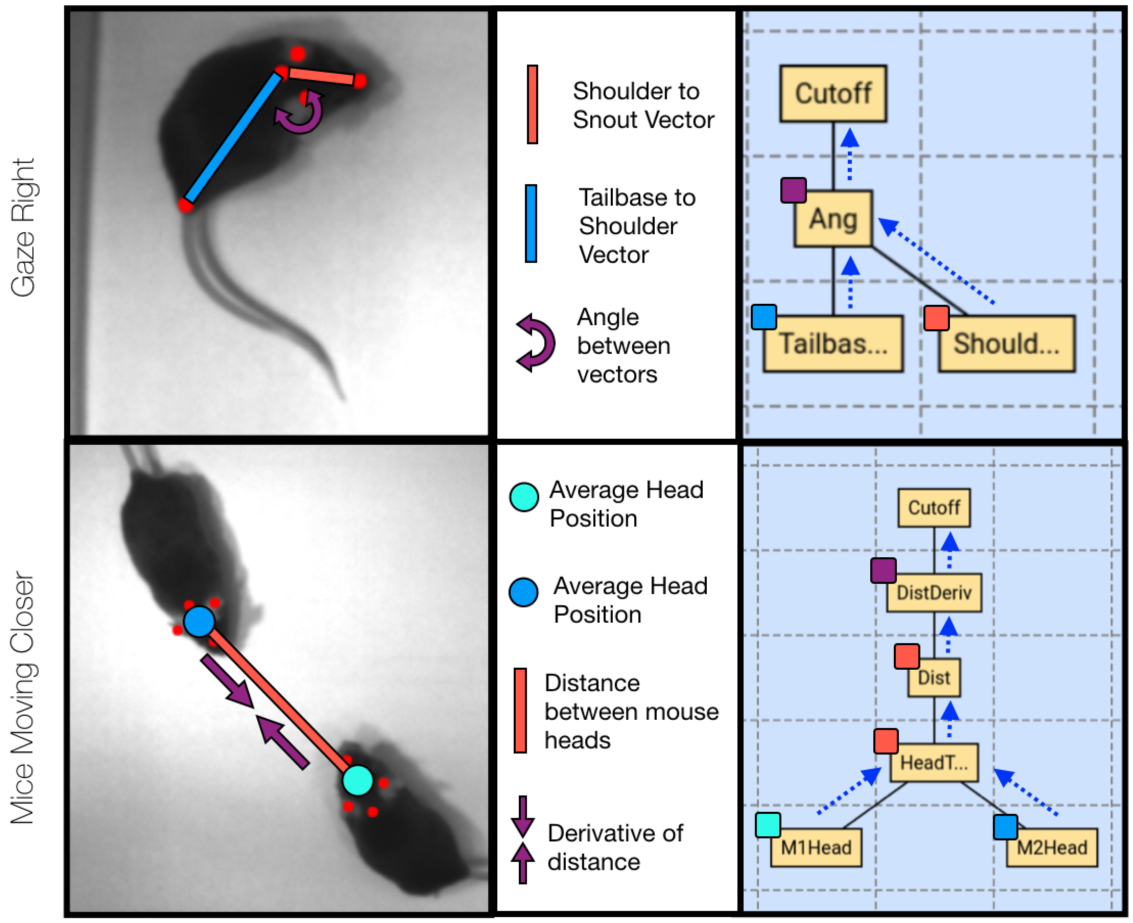

Figure 2.2: Illustration of how behavior is defined in Basin.

Examples of Basin Measurement

Figure 2.2 illustrates the manner in which behaviors are defined in Basin. The figure depicts images of mice behaving on the left, and then depicts the Basin behavior measurement system on the right. The rightmost images of connected blocks are screenshots of the Basin behavior definitions, overlaid with small colored tiles and dotted arrows for illustration purposes. Users assemble behavioral measurements in Basin by double-clicking on empty tiles (the tiles are demarcated by grey dotted lines in Basin, shown in 2.2), which enables users to instantiate a primitive operation by selecting inputs and defining parameters for the operation. Once a user creates an operation, a yellow block appears in the formerly empty tile, and each yellow block represents a single primitive mathematical operation. Yellow blocks are connected by black lines when a block takes another block as input. Blocks can also take as input points of the animal's pose[1]. Collectively, the yellow blocks in a tree constitute a behavioral measurement. The output of the measurement is usually the highest block in the tree of primitive operations (which is analogous to the outermost function in a composition of many functions). In figure 2.2, the dotted blue arrows show the direction in which data is passed as it ascends through the composed functions.

The top row of figure 2.2 depicts the behavior "gaze right", which is defined in the following manner. First, two vectors are computed from the underlying pose, one leading from the mouse's shoulder area to the mouse's snout, and a second from the mouse's tail base to its shoulder area. Basin includes a "vector subtraction" operation, which is used to compute these two quantities. These two operations are tagged in red and blue in the top of figure 2.2; the line segments defined are graphically shown over top the image on the left, and the blocks which correspond to the vector computations are shown on the right. These two computations are then fed up into an angle operation, which computes the angle between the two vectors. Finally, the angle operation feeds into a cutoff operation, which determines whether the mouse's head is at a sufficient angle to merit the behavior being called "gaze right".

A second behavioral measurement is shown in the bottom of figure 2.2. Here, Basin measures whether two mice are moving closer to each other. To define this measurement, two averages of the head position are taken, which average the four points (left ear, right ear, snout, and shoulder) at the mouse's head. From these two averages, a vector between the heads of the mice is computed, a norm of this vector is taken, and then a derivative of this vector is taken. The derivative is further fed into a cutoff operation, and if the derivative is sufficiently negative, then a nearing behavior is measured. All of these computations are represented in the tree in the bottom right of figure 2.2. Note that although only these computations are shown for simplicity, a more robust definition of approaching behavior should include additional criteria, such as whether the mice are within a sufficiently close range (a fuller definition is given later in def. 3.2). The visualization capacities of Basin facilitate this progressive refining of Basin definitions.

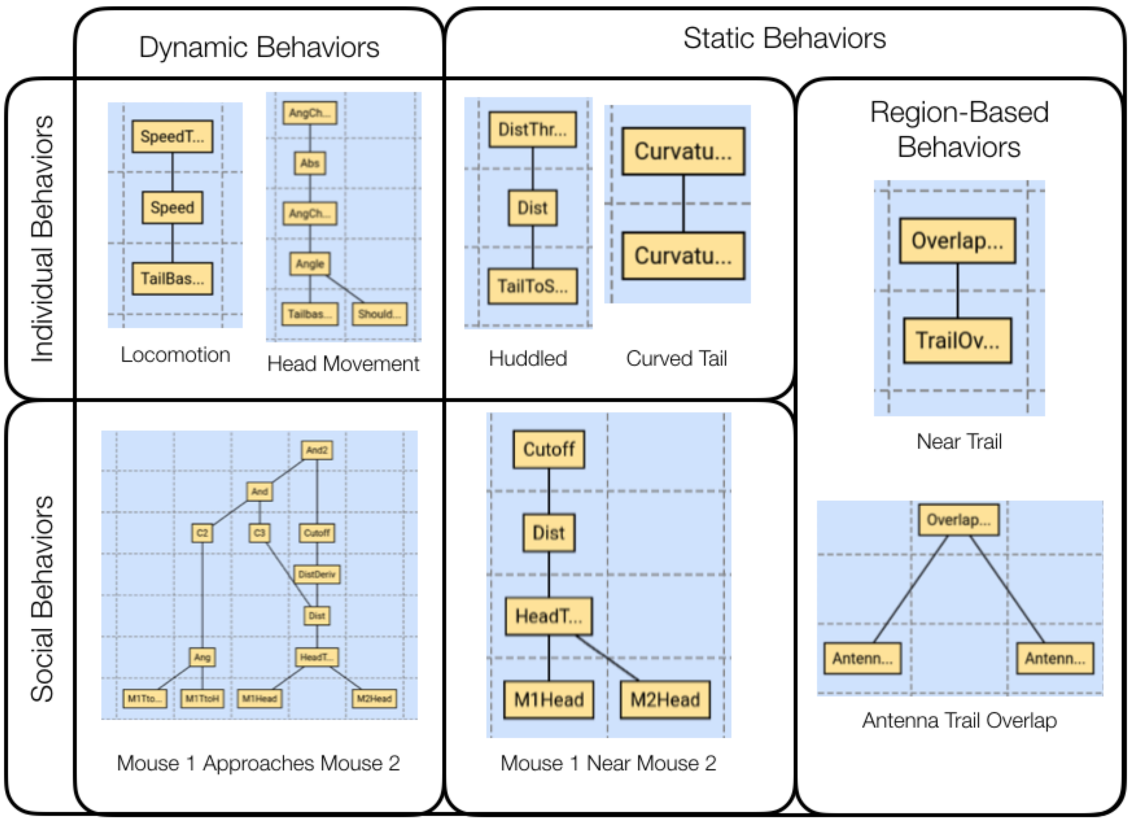

Figure 2.3: A summary of many of the types of behaviors which are definable in Basin.

2.1.3 The Generality of Basin Definitions

The approach described above for measuring behavior yields several benefits, one of which is that the behavioral measurements that Basin supports are quite general. This generality is shown in figure 2.3, which outlines how Basin measurements can fall into several different classes. First, Basin can measure static and dynamic behaviors; i.e., those occurring in a single frame or across multiple frames. Static behaviors, such as huddling, are defined based on geometric configuration of body points, while dynamic behaviors often invoke the use of an numerical derivative, or similar constructs to make use of multi-frame information. As the temporal span of a behavior lengthens, it becomes more difficult for that behavior to be defined in Basin, since the variability in the geometric configuration of a long-time behavior is often quite large.

[1]No lines are depicted for inputs of the animal's pose, since no yellow blocks directly represent the animal's pose. Generally, the lowest blocks on a tree take as input points from the animal's pose.

In addition to defining both static and dynamic behaviors, Basin can also define both individual and social behaviors. Recent advances in deep learning have made it possible track multiple animals simultaneously while maintaining the identities of those animals over time. Thus, in many cases, defining social behaviors as a function of the pose geometry is extremely simple, and a few examples of the types of behaviors which may be defined are shown in fig. 2.3, which includes a "near" behavior as well as a "approach" behavior.

Region-based Behaviors

Finally, Basin can also define so-called "region-based" behaviors. In many experimental setups, animals interact with their environment, and so behaviors are defined relative to the environment with which the animals are interacting. For instance, an example that I will discuss later is an ant following a trail, a region within the video frame. To define these behaviors, information about regions must be input into Basin, in addition to DeepLabCut body point positions. Basin accepts input files which contain polygons for every single frame which define the relevant regions. Per-frame information is necessary, since regions may change over time (for example, a mouse may move its nest over the course of a video). Inputting polygons, rather than masks, dramatically lowers the storage size, usually without diminishing the segmentation quality to too great an extent. A variety of region-based behaviors can be defined in Basin, a few of which are displayed in fig. 2.3. Shown here are two representations of functions for measuring ant trail-tracking, one which measures whether an ant is near a trail by testing the degree to which an ant's head overlaps with the trail, and a second which measures how much an ant's antennae overlap with the trail.

2.2 Basin is Interpretable and Comparable

In this section, I analyze Basin under the framework for behavioral analysis systems discussed in the preceding chapter. I start by arguing that Basin provides an interpretable form of behavioral measurement. The foundation for interpretability in Basin is its reliance on pose. Rather than relying on high-dimensional video frames, which require inherently complex and high-dimensional classification functions, Basin starts with a much simpler representation of an animal, the pose. Next, Basin feeds the animal's pose into a series of intuitively clear, and programatically short operations, which terminate in a measurement of behavior. Thus, Basin measurements may be readily comprehended as a short measurement function over pose.

However, the true standard for successful interpretability is not how well interpretability is argued, but rather the ability of others to interpret the behavioral measurement being used. Thus, the remainder of this thesis provides practical examples of the interpretability of Basin by defining each behavioral definition in the text, in a standardized format. The format is given below for an example behavior of gaze right:

Definition 2.1: Gaze Right (GR)
Gaze right is defined as:

$$\begin{aligned}\mathbf{GR} = \mathbf{Cutoff}(\mathbf{SignedAng}(&\mathbf{VSub}(snout, shoulder),\\ &\mathbf{VSub}(tailbase, shoulder), c, True)\end{aligned}$$

Here, *snout*, *shoulder*, and *tailbase* represent the body points on the mouse. The parameter c represents the cutoff point at which a particular angle between the snout and shoulder, and tailbase and shoulder constitutes "gaze right".

Each constituent function which is not explicitly defined within a definition box (for example, the **VSub**, **SignedAng**, and **Cutoff** operations above) is defined in Appendix A. I note that to create a truly correct definition, each body part should be a function of a time variable (e.g. $snout(t)$, $shoulder(t)$, etc.), and the gaze right variable should also be a function of time. For succinct notation, however, I do not include this time variable, and leave it implicit.

I would like to suggest here that the reader consider the behavioral definitions in the text carefully, rather than assuming that they are "correct" as stated. In the event the reader disagrees with the definition given in the text, I certainly admit that there may be some unforseen limitations to the way these behaviors are defined here. However, I would like to note that such a discussion about how to measure something - an extremely important, and common discussion within the scientific community - would be impossible if one used model-based behavioral definitions for which the classification function is far too complex to explain to a human and debate.

Finally, I claim that researchers can construct comparable measurements within Basin. There are a few simple reasons for this. First of all, because Basin behavioral definitions start at pose, they are invariant to the recording environment in the manner discussed in the preceding chapter. Thus, different researchers can measure the same behaviors within their own experimental paradigms, without any concern about compatibility. In fact, researchers can export behavioral measurements defined in Basin in the format of a JSON file, which can be easily distributed to others interested in analyzing the same behaviors for their own data, within Basin. Even for researchers who do not wish to use Basin itself, the precision of the definitions for the behavioral measurements makes it easy to reproduce the results using one's own analysis tools.

Second, Basin also partially addresses the problem of invariance to irrelevant animal features, mentioned in the comparability chapter above. One of the essential features in Basin is a cutoff operation, which converts scalar arrays to binary arrays by comparing each element of the scalar array to some fixed cutoff value. While a hard cutoff like this may be susceptible to inter-animal size variation, a percentile cutoff, which is true for all values above or below a certain percentile-based threshold for a particular animal, may be more useful (see section 1.3.2 for a more detailed example). By providing both percentile-based and fixed-value-based cutoff operations, Basin gives users the flexibility to tailor their behavioral measurements to overcome inter-animal invariance issues.

Chapter 3

Measuring Behavior With Basin

The aim of this chapter is to present examples of measurements performed in Basin on real datasets. I start by presenting analysis of a simple database of mouse behavior, and then move on to more complex examples in which I replicate results about ant trail-tracking behavior, and also show results about the manifold structure of ant walking behavior.

3.1 Analysis of Mouse Behavior

Due to the preeminence of mice as model organisms in the neuroscience community, a variety of the methods of automated behavioral analysis have been developed in part or in whole for the analysis of mice behavior (33; 20; 37; 14). Since mice are a such an important model organism, I demonstrate here that Basin can measure the behavior of mice. My analysis here is loosely inspired (14), a system for analyzing a variety of individual and social behaviors of mice. The authors of (14) show that their system of behavioral measurement can detect a number of interesting patterns of behavior; for example, they measure differences between genetically altered mice serving as models for autism spectrum disorder and wildtype mice. Here, I measure several behaviors in Basin which closely match the measurements of (14), to suggest that Basin can do similar analyses. Of course, the difference is that in Basin, none of these measurements are hardcoded, and instead are readily created by the user. The dataset I use for measuring mice behavior is a small dataset of video depicting three mice in a cage. These data were collected by Dr. Alexander Mathis. Part of the dataset captures three mice exploring a fixed environment, and in a separate part the floor of the cage continually moves.

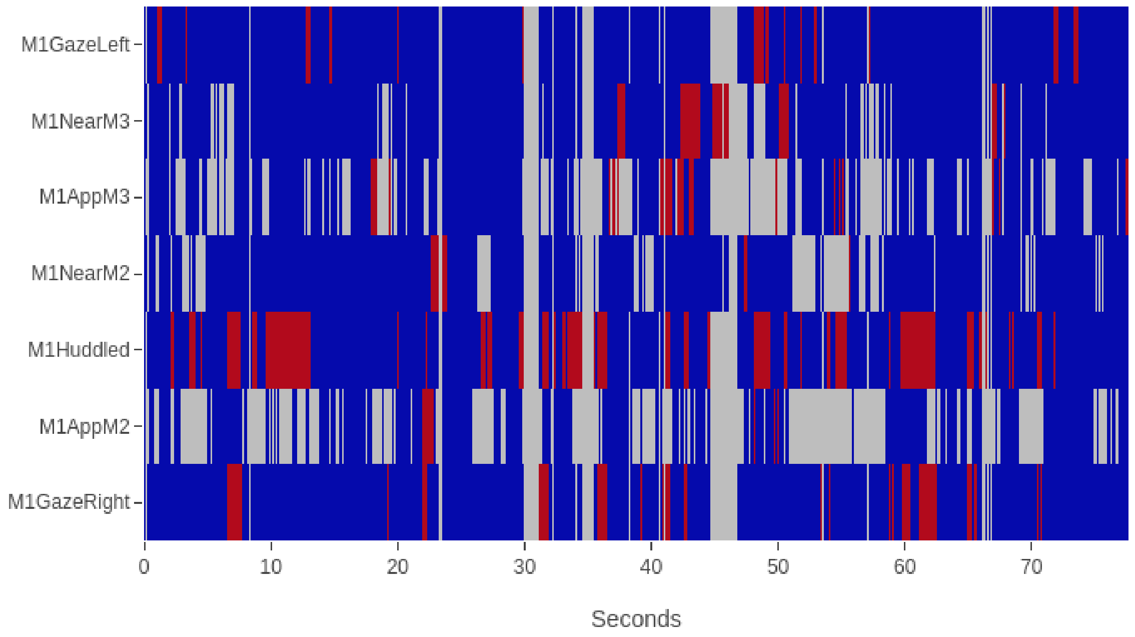

Figure 3.1: Ethogram of mouse behavior. Red bars indicate a behavior is occurring at some moment in time. Blue bars indicate a behavior is not occurring at that moment in time. Grey bars indicate no determination could be made, due to a visual occlusion or failure of the underlying body point tracking system to identify the location of a body point.

During the video, the mice show a variety of behaviors: they walk around, they stop and huddle, they look around, they approach the other mice in the cage, and more. To capture some of this behavior, I defined a number of behaviors for one of the mice, including individual and social behaviors. Figure 3.1 shows the results of this analysis. The display is in the form of an "ethogram", a plot which displays whether or not a behavior is occurring at each point in time. If desired, a variety of analyses can be performed on the data which is used to create the ethogram, including investigations of the transitions between behavioral states, comparisons of the behavioral activity between mice, and more.

Below, I give some of the definitions for the behavioral measurements depicted in figure 3.1. I defined gaze right above in definition 2.1, so I don't restate that here. What I do show are definitions for huddled behavior, mouse 1 approaches mouse 2, and mouse 1 near mouse 2, given as definitions 3.1, 3.2, and 3.3 respectively. The approach behavior is complex, in that it not only detects whether the mice are getting closer together, but also whether the approaching mouse is oriented towards the approached mouse, and whether the two mice are sufficiently close to each other.

Definition 3.1: Mouse 1 Huddled (M1Hud)
I can define huddled as follows:

$$\mathbf{M1Hud} = \mathbf{Cutoff}(\mathbf{Norm}(\mathbf{VSub}(m1\_snout, m1\_tailbase)), c, True)$$

Here, c is a scalar value used to determine the point at which the mouse's posture is sufficiently small to merit being called huddled.

Definition 3.2: Mouse 1 Approaches Mouse 2 (M1AppM2)
I start by defining an distance measurement. To shorten the definitions, I refer to mouse one's snout as $m1sn$, the shoulder as $m1sh$, the tailbase as $m1tb$, a point on the spine as $m1sp$, the left and right ears as $m1le$, $m1re$ respectively, and likewise for mouse 2:

$$\begin{aligned}\mathbf{Dist} = \mathbf{Norm}(\mathbf{VSub}(&\mathbf{Avg}(m1sn, m1tb, m1le, m1re),\\ &\mathbf{Avg}(m2sn, m2tb, m2le, m2re)))\end{aligned}$$

I also measure the angle between the two mice:

$$\mathbf{IsFacing} = \mathbf{Cutoff}(\mathbf{Ang}(\mathbf{VSub}(m1tb, m1sn), \mathbf{VSub}(m1tb, m1sp)), c_1, True)$$

Finally, I combine these to measure the behavior:

$$\begin{aligned}\mathbf{M1AppM2} = \mathbf{And}(&\mathbf{Cutoff}(\mathbf{SDeriv}(\mathbf{Dist}, k), c_2, True),\\ &\mathbf{And}(\mathbf{IsFacing}, \mathbf{Cutoff}(\mathbf{Dist}, c_3, True)))\end{aligned}$$

Here, c_1,c_2, and c_3 are just cutoff parameters for turning scalar values into binary values. k is a parameter controlling the window over which the empirical derivative is computed. Basically, this definition simply checks whether 1) the mice are getting closer, 2) mouse 1 is facing mouse 2 (otherwise, it couldn't be approaching), and 3) the mice are sufficiently close to each other (approaching can't happen when the mice are very far apart, even if they are getting closer).

Definition 3.3: Mouse 1 Near Mouse 2 (M1NearM2)
I use the same **Dist** function as definition 3.2. Then, I can define this function as:

$$\mathbf{M1NearM2} = \mathbf{Cutoff}(\mathbf{Dist}, c, True)$$

3.2 Analysis of Ant Behavior

3.2.1 Replicating Measurements of Trail-Following Behavior

The second dataset with which I demonstrate the efficacy of Basin is a dataset of ant trail-tracking. All data were collected by Dr. Souvik Mandal, a postdoc in the lab of Professor Murthy. Videos depict the behavior of *Camponotus pennsylvanicus*, a species of carpenter ant. Pheromone extracted from ants was drawn on paper to motivate trail-following behavior. Ants were then introduced into a walled-off environment, near the start of the pheromone trail. Ants were then free to roam, and for the videos analyzed, they followed the trail for the majority of each video. The data were collected by mounting a camera on a motorized track, which moved along the track as it detected the ant. Thus, videos appear as 400 by 400 pixel square frames which mostly contain the ant. A DeepLabCut model was then trained by Dr. Mandal to track thirty points on the ant's body in each frame.

Since I am interested in ant-trail tracking behavior, I needed to extract segmentations of the pheromone trail visible in the video frames. Because the video lengths were relatively small, and because the ant trail was in many cases visually similar to the background paper, I elected to hand-segment the trail from video frames, rather than train an algorithm to do the analysis. Thus, I hand-segmented every fourth video frame, and held the segmentations constant for four video frames at a time (the visual change in trail segmentations between frames at the four-frame level was generally small, suggesting that this was a low-error approximation).

To illustrate the efficacy of Basin to measure behavior, I use Basin to reproduce the results of (11), a paper from Prof. Murthy's lab which previously demonstrated various properties of ant trail-tracking. I note here that these measurements are collected in a very different manner than those in (11): first, I use an entirely different dataset, and second, while in (11) code was written specifically to analyze the behaviors desired by the researchers, here the behaviors are defined in the generic framework of Basin.

Figure 3.2 displays the results of my ant analysis. I compare my results with figures 3e and 4d from (11), and to avoid confusion, I reference the figures from (11) by appending the figures with a star; thus, figure 3e becomes 3e*. One of the first analyses performed in (11) was to measure the degree of overlap between the

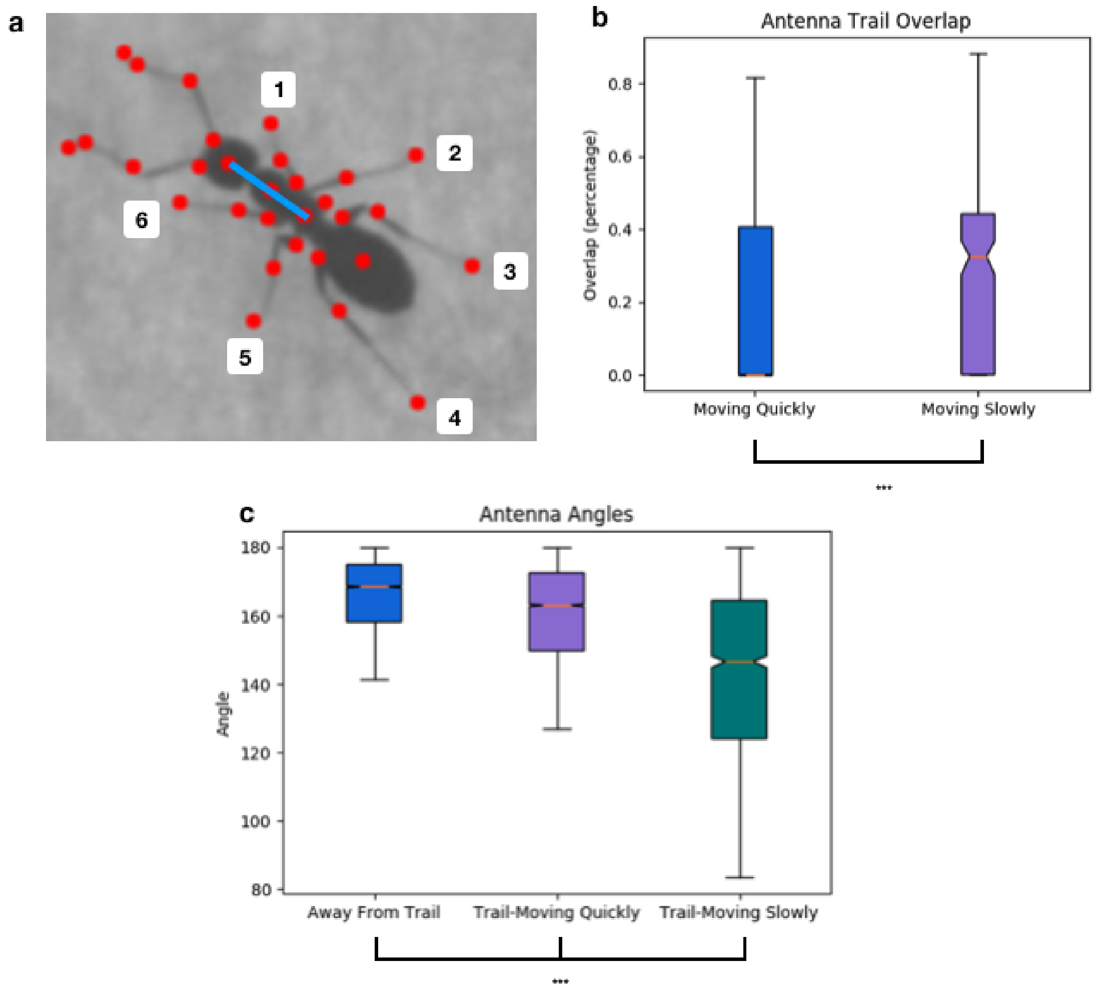

Figure 3.2: Ant analysis in Basin. **(A)** is a single video frame, which shows red dots for all of the points detected by DeepLabCut, along with a number identifying each leg. **(B)** is a boxplot which depicts the distribution of how much a octagon centered at an antenna overlaps with the trail when the ant is moving quickly (measured by leg movement), and when the ant is moving slowly. Distributions are significantly different (Kolmogorov-Smirnov, $p < .0001$). $N = 5$ videos used, with 24216 total frames, of which 1891 occur when the ant is moving slowly, and 11029 occur when the ant is moving quickly (values do not sum to 24216 because only points near the trail are considered). **(C)** gives the distribution of the angle of the major joint in the middle of the antenna, when the ant is away from the trail, when the ant is moving quickly near the trail, and when the ant is moving slowly near the trail. Distributions are significantly different (Kolmogorov-Smirnov, $p < .0001$). Here, 11296 frames occur when the ants are away from the trail (with the same number of frames as **(B)** for the other two conditions).

Definition 3.4: Antenna Trail Overlap (ATO)
The definition of antenna trail overlap is

$$\mathbf{ATO} = \mathbf{SSum}(\mathbf{AreaOverlap}(bp6, trail, w), \mathbf{AreaOverlap}(bp19, trail, w))$$

Here, $bp6$ is a point near the tip of the left antenna, $bp19$ is a point near the tip of the right antenna, and $trail$ is the trail polygon. Parameter w is the apothem of the octagon used to define area overlap.

Definition 3.5: Moving Quickly (MQ)
Let single leg quick movement be defined as follow:

$$\mathbf{MQS}(p1, p2, p3, k, c) = \\ \mathbf{Cutoff}(\mathbf{SAbsValue}(\mathbf{SDeriv}(\mathbf{Ang}(\mathbf{VSub}(p1, p2), \mathbf{VSub}(p1, p3)), k)), c)$$

Then, moving quickly is defined as:

$$\mathbf{MQ} = \mathbf{And}(\mathbf{MQS}(bp3, bp25, bp1, k, c), \mathbf{MQS}(bp3, bp12, bp1, k, c))$$

Here, $bp3$ is the midpoint of the body, $bp25$ is the endpoint of leg 2 (as defined by figure 3.2), and $bp12$ is the endpoint of leg 5. Parameter k is the window over which the derivative is calculated, and c is the cutoff point at which there is considered to be sufficient motion to merit the behavior being called moving quickly. The calculation is essentially determining whether legs two and five are swinging back and forth sufficiently quickly to be called moving quickly.

ant's antenna and the trail when the ant is walking and not walking. Ants often move more slowly when they are sampling the trail with their antennae to test for pheromone; corresponding with this intuition, the authors of (11) find greater antenna overlap when the ants are moving slowly, as opposed to quickly. I replicate this analysis here (the definition for antenna-trail overlap is given in def. 3.4).

In (11), the authors have a setup which enables them to have access to the global position of the ant within the arena - that is, the position of the ant relative to a fixed, constant point in space. In contrast, as I described above, in the setup of the new dataset the video camera is constantly moving with the ant and no similar global position was recorded. This means that I cannot define movement speed simply by taking the derivative of the global position. To circumvent this problem, I defined movement by looking at the movement of the ant's leg relative to the spine of the ant (def. 3.5).

Figure 3.2(b) shows the results of the analysis, which measures the percentage of overlap which an octagon centered at the ant's antenna has with the trail. In this figure, the median - depicted by orange lines in both plots - is at the bottom of the box plot titled "moving quickly", showing a strong difference in distributions between the two plots (tests to measure this difference are also discussed in the figure caption). In particular, results indicate less trail-sampling occurring in the moving condition. The results here are quite comparable to figure 3e*. In 3e*, the authors define a "probing" state, which occurs when the ant is not moving quickly, and the "sinusoidal" and "trail-following" states are those in which the ant is moving quickly. The authors of (11) find a quite similar difference in antenna-trail overlap, with ants sampling the trail more when they move more slowly.

Definition 3.6: Head Trail Overlap (HTO)
The definition of head trail overlap is

$$\mathbf{HTO} = \mathbf{Cutoff}(\mathbf{AreaOverlap}(bp1, trail, w), c, False)$$

Here, $bp1$ is the head body point, and $trail$ is the trail polygon. Parameter w is the apothem of the octagon used to define area overlap, and c is the cutoff point at which there is considered to be an overlap.

Definition 3.7: Antenna Angle (AA)
The definition of antenna angle is

$$\mathbf{AA} = \mathbf{Ang}(\mathbf{VecSub}(bp7, bp6), \mathbf{VecSub}(bp7, bp1))$$

Here, $bp1$ is the head body point, $bp6$ is a point near the tip of the left antenna, and $bp7$ is a point at the middle of the antenna. The angle is thus measuring the angle of the antenna.

A second analysis was performed to test the relationship between moving speed and antenna angle in three conditions; when the ant is away from the trail (measured with def 3.6), when the ant is near the trail and moving quickly, and finally when the ant is near the trail and moving slowly. The antenna angle was measured with the function given in def. 3.7. The results indicate that this angle varies most when the ant is near the trail and moving slowly, which is consistent with the results above, suggesting that during these periods the ant is sampling the trail, which requires a bend of the antennae. Further, when the ant is moving more quickly near the trail or is away from the trail, the antennae angles are much closer to straight. Again, this analysis closely matches figure 4d* from (11), in which the authors find a closely

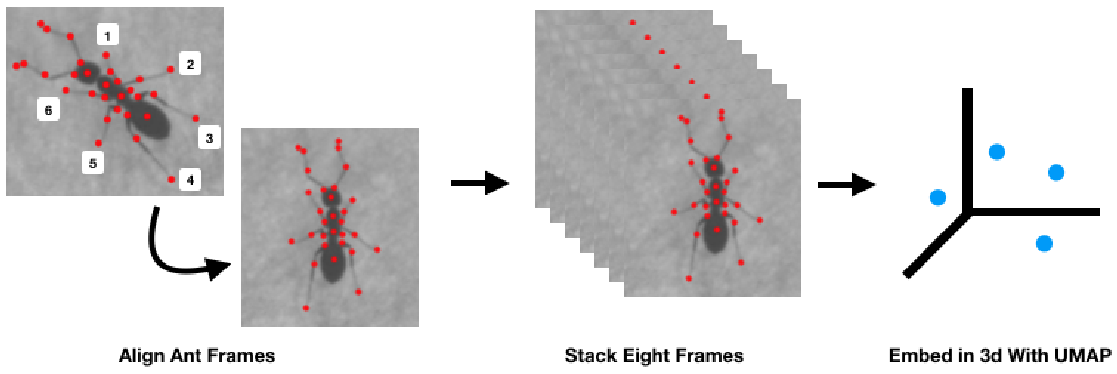

Figure 3.3: Embedding process with UMAP. Six body points, each with two coordinates, are centered and rotated to ensure constant alignment on a per-frame basis. Then, eight frames of these coordinate sets are stacked together to form a feature vector, which is then embedded into three-dimensional space with UMAP.

matched widening of the antenna distribution during the periods of the time when the ants are moving more slowly and are closer to the trail. Overall, both this result and the above result suggest that Basin is effective at measuring behavior, in that it can replicate results found by previous groups.

3.2.2 Manifold Structure of Ant Locomotion

In the first chapter, I discussed how dimensionality reduction has been used to visualize the intrinsic structure present in high-dimensional data of behaving animals. In particular, I emphasized that one must rigorously characterize the common features of data within specific regions of the lower-dimensional structure to provide an accurate characterization of the manifold. The authors of (9) provide an excellent example of how this analysis should be done, and here I show how Basin can be leveraged to perform a similar analysis in an expedient manner. For this chapter, when I use term "manifold", I am simply referring to a structure of points in a space, rather than the technical definition of a collection of points for which each neighborhood can be embedded in a lower-dimensional euclidean space.

In (9), the authors record fly walking behavior, extract fly feet position, stack thirty frames of fly walking data together as a single feature vector, and then embed these vectors into three-dimensional space. The embedding is performed using the Uniform Manifold Approximation and Projection (UMAP) algorithm introduced in (23). The authors of (9) discover a remarkably well-defined manifold which resembles a bell; this manifold encodes gait phase along its circular axis, and further encodes different gait types (tripod, tetrapod, etc) along its central axis. Here, I do a very similar analysis on ants, which is illustrated in figure 3.3.

Remarkably, when I perform the UMAP embeddding, I find a strikingly similar structure to that found in (9). The projected data is depicted in figure 3.4a. The primary substantive difference in the visual appearance of the manifold here is that the manifold expands in width and then contracts along its middle axis, making it resemble a vase somewhat more than a bell. It is important to note that this is a totally different organism (ant vs. fly), recorded under entirely different conditions, and tracked using completely different methods. The fact that the manifolds appears similar despite these differences in setup strengthens the argument that the manifold's structure is more due to the intrinsic walking patterns of insects, rather than some idiosyncrasy of either experiment. In what follows, I refer to two axes of 3.4; the *central axis* is the linear axis around which the manifold in 3.4a is nearly rotationally symmetric; the *circular axis* is the angular axis which measures angles in the plane orthogonal to the central axis.

To investigate the structure of the manifold, I defined Basin behavioral measurements and then used these measurements to color the manifold. The result of this analysis is shown in figures 3.4b and 3.5a,b. To compare my results to (9), I use the convention introduced above, in which I append an asterisk to figure numbers which refer to figures in (9). In figure 3.4b, I show the results of a very similar analysis displayed in figure 4b*. The only difference is that in 4b*, the position of the mid legs is analyzed, whereas here, the norm of the front-left leg is used (measurement defined in def. 3.8). The coloring of the manifold in 3.4b is quite similar to that in 4b*; the coloring suggests the phase of the walking gait (for which these leg position measurements are a proxy) is encoded in the manifold along its circular axis.

I then aimed to understand the central axis. The authors of (9) used multiple methods to characterize the structure of the central axis. First, they trained a footfall-detection algorithm to measure the number of ant feet "in stance," or touching the ground at a particular point in time. Second, they embedded synthetic walking data which perfectly models various gait patterns into the manifold, to understand which sections of the manifold model which gaits. The authors first found strong segregation of the manifold by number of feet in stance, with the majority of the manifold showing three feet in stance, and the number of feet in stance increasing along the central axis (see fig. 4d*). They further found that points on the central axis which lay closer to the opening or lip of the manifold

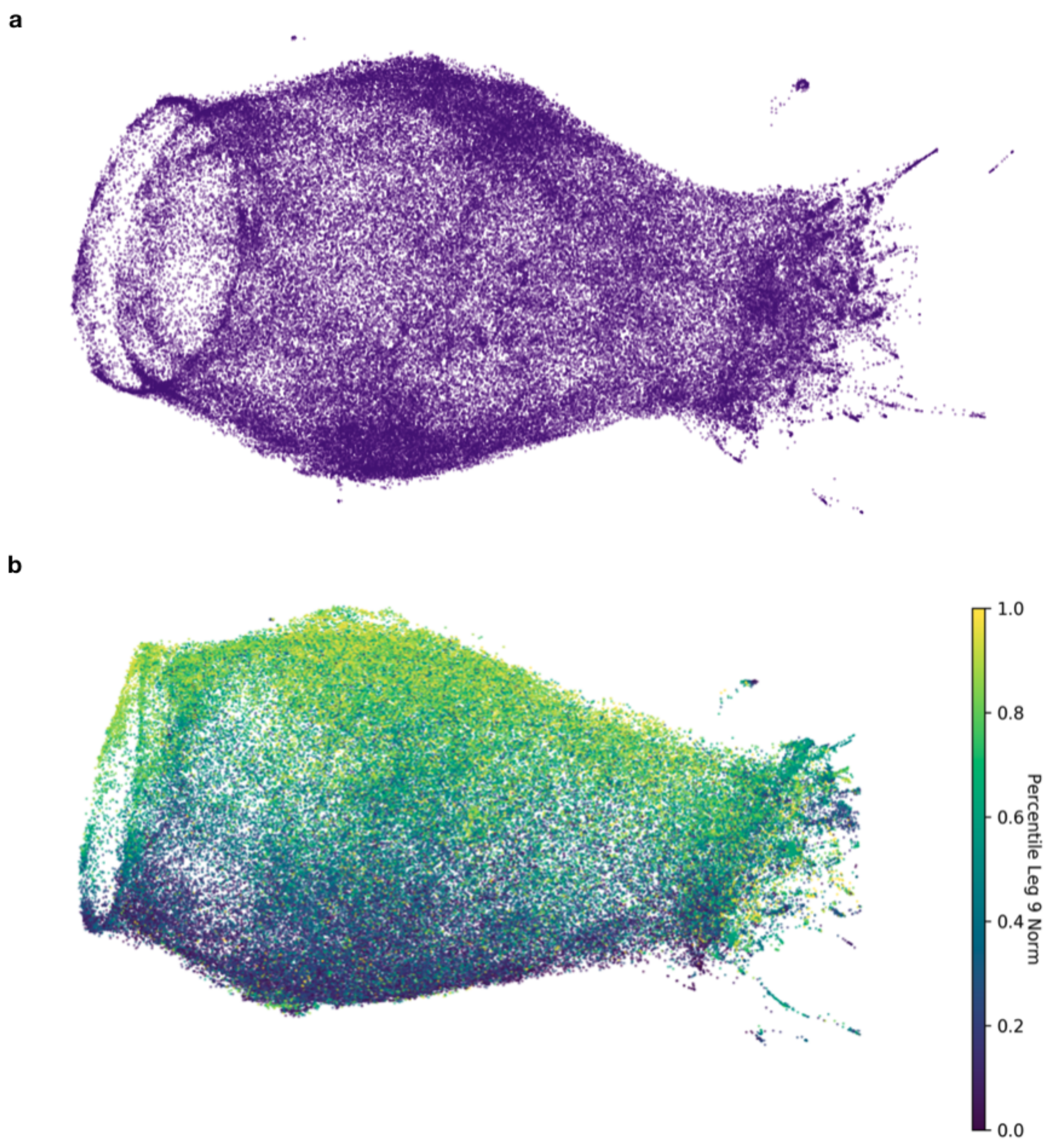

Figure 3.4: Results of ant UMAP embedding. **(A)** is an illustration of the UMAP manifold, which consists of over 100k frames of ant walking data. In some videos, the ants are trail-tracking, but most are simply of generic walking behavior. **(B)** is the same manifold, colored by the percentile of the norm of the front-left leg. Loosely, this corresponds to the phase of the ant's gait as it walks.

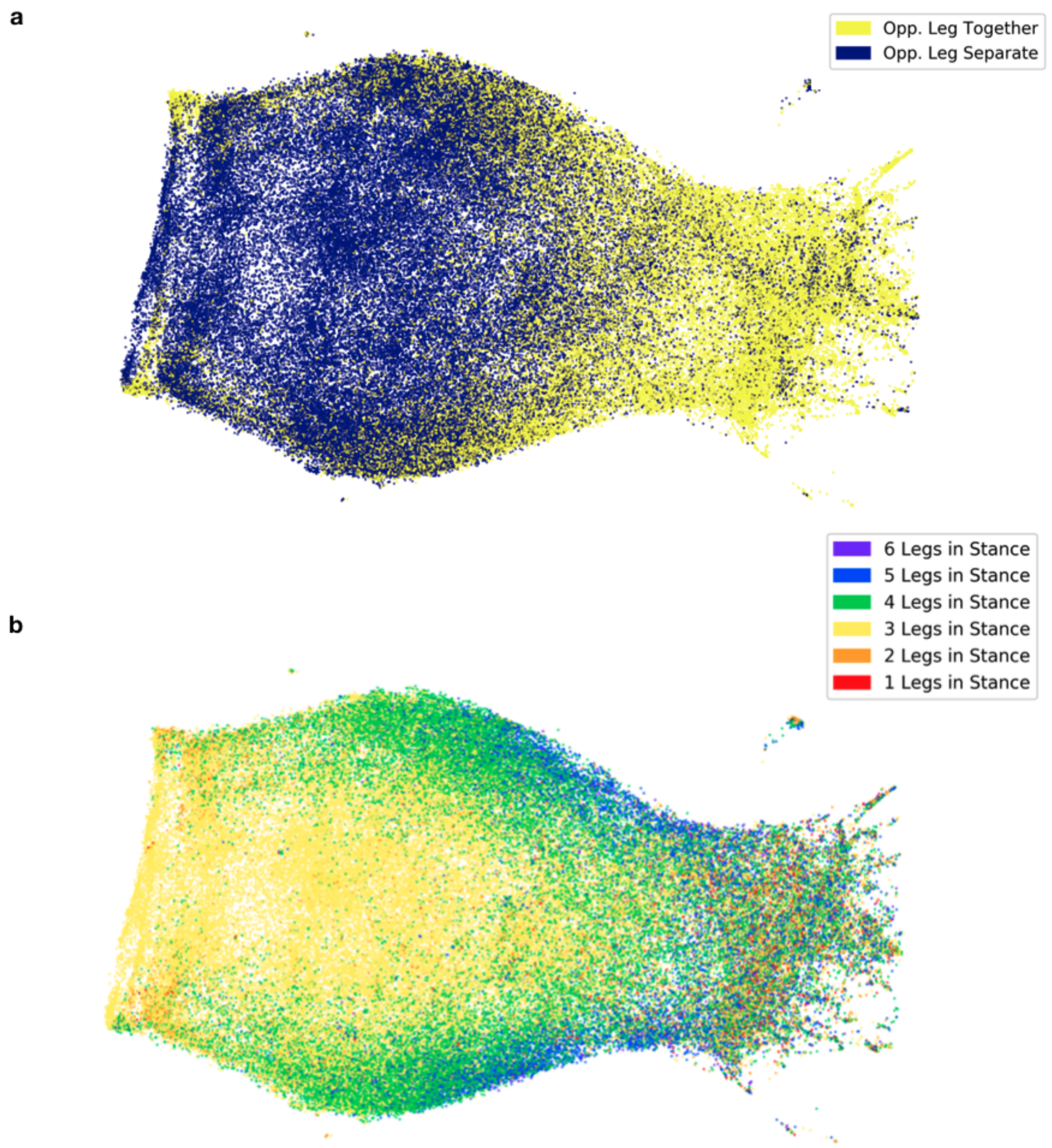

Figure 3.5: More colorings of ant UMAP embedding. **(A)** is colored in a binary fashion based on whether either pair of opposite legs is moving together (pairs of opposite legs are front-left and back-right, as well as front-right and back-left). **(B)** shows the number of legs "in stance", or touching the ground, at any point in time.

Definition 3.8: Leg Length (LL)
Leg length is measured with

$$\mathbf{LL} = \mathbf{Norm}(\mathbf{VecSub}(bp3, bp9))$$

Here, $bp3$ is the middle of the ant's body, and $bp9$ is the tip of the ant's top left leg. Obviously, three-dimensional leg length does not change, but since the video is recorded form above, the length of the leg appears to change from the perspective of the camera as it contracts, which corresponds to the phase of the gait.

corresponded to the fly performing an alternating-tripod gait, a gait characterized by the fly placing three feet on the ground at at time, and moving the other three. The gait then shifted down the central axis to become a tripod gait (four legs on the ground, two legs moving) and then further down the central axis were other "non-canonical" gaits involving four of five feet placed on the ground.

Definition 3.9: Opposite Legs Together (OLT)
Let slow leg movement be defined as

$$\mathbf{LM}(p, k, c) = \mathbf{PCutoff}(\mathbf{Norm}(\mathbf{VDeriv}(p, k)), c, True)$$

Where p is a leg point, k is the window used for calculating the empirical derivative, and c is a cutoff. Then, opposite legs are moving together if the following is true:

$$\begin{aligned}&\mathbf{OLT} = \\ &\qquad \mathbf{Or}(\mathbf{And}(\mathbf{LM}(p28, k, c), \mathbf{LM}(p9, k, c)), \mathbf{And}(\mathbf{LM}(p15, k, c), \mathbf{LM}(p22, k, c)))\end{aligned}$$

Body parts $p28, p9, p15$, and $p22$ are at the tips of legs $3, 6, 4$, and 1 respectively. If either pair of opposite legs are both moving slowly, then this measurement function returns true, and the gait is not a tripod gait. This is true because in a tripod gait either legs 1, 3, and 6 or legs 2, 4, and 5 are always moving together (numbering from figure 3.2). Thus, there is never a situation in which legs 3 and 6 or 1 and 4 are both moving slowly. In contrast, in tetrapod gait, at least one of the pairs of legs is always moving slowly.

Here, my intention is to perform all analyses with Basin. Some of the analyses described above are either incongruent with our experimental setup, or difficult to perform within Basin, so I construct alternative analyses which approximate the analyses performed in (9). The first such analysis which I perform tests whether either pair of opposite legs (pair 1 and 4 or pair 3 and 6 from figure 3.2) are both moving slowly. For reasons explained in definition 3.9, if this test is true, then an alternating tripod gait is not being performed. The result of this analysis is given in figure 3.4c. What I observe is that the opposite legs move separately near the lip of the manifold, whereas further down the manifold they move together. This result is exactly as one would expect if the points closer to the lip are in a tripod gait, and the points further down the central axis are in a tetrapod or higher-order gait.

Definition 3.10: Feet Not In Stance (FNIS)
I start by defining a function which measures whether a leg is moving up the ant's body:

$$\begin{aligned}&\mathbf{MUB}(p,k,c) = \\ &\qquad \mathbf{Cutoff}(\mathbf{SDeriv}(\mathbf{Proj}(\mathbf{VSub}(bp1,bp3),\mathbf{VSub}(p,bp3)),k),c,False)\end{aligned}$$

The **MUB** operator measures whether a foot is not in stance by measuring the derivative of a projection of the ant's leg on the ant's backbone. The vector from $bp3$ (a point on the middle of ant's spine) to p (a point at the tip of an ant's leg) is projected on to the vector from $bp3$ to $bp1$, a point on the ant's head. Thus, this measures whether the ant's leg is moving forward in the axis along the ant's body. Then I define the primary function:

$$\begin{aligned}\mathbf{FNIS} = \mathbf{BSum}(\mathbf{BSum}(&\mathbf{BSum}(\mathbf{MUB}(p28,k,c),\mathbf{MUB}(p9,k,c)),\\ &\mathbf{BSum}(\mathbf{MUB}(p15,k,c),\mathbf{MUB}(p22,k,c))),\\ &\mathbf{BSum}(\mathbf{MUB}(p25,k,c),\mathbf{MUB}(p12,k,c)))\end{aligned}$$

The binary sum operator only takes two arguments, which necessitates a number of different sum operations to sum all six values of **MUB**, but the core operation is simple: simply add all the binary values of **MUB** together. Here all body part arguments to **FNIS** are points on the tips of ant legs. k, c are parameters controlling the window of the empirical derivative, and the cutoff parameter, respectively.

This suggestion of gait type represented along the central axis is strengthened by the next analysis of number feet in stance. Since the recording conditions for the experiment are not conducive to directly measuring the number of feet in stance, I adopt a proxy measurement. In essence, I note that when an ant's foot is not on the ground (i.e. not in stance), it is moving up the axis of the spine. Thus, I can measure the total number of legs moving up an ant's spine using definition 3.10.

When I count the number of feet in stance and color the manifold with this number, I find the manifold is colored quite similarly to 4d* from (9). My result is shown in figure 3.5b. As one moves down the central axis, the number of feet in stance increases, and the majority of the manifold is dominated by three feet in stance. This plot also matches up quite well to 3.5a. When three legs are in stance, the ant is likely performing an alternating tripod gait, and these regions appear to correspond to the regions in 3.5a in which the opposite legs are not moving together, which I suggested above should occur during an alternating tripod gait.

In sum, the multiple convergent analyses of the central axis of the manifold suggest that this axis encodes gait information, a result which parallels that found by (9) in the fly manifold. These results suggest that Basin can be a valuable tool for making intuitions about the structure of lower-dimensional representations of behavioral data more concrete and rigorous.

Chapter 4

The Architecture of Basin

4.1 The Software Systems of Basin

Basin is composed of two software components: a front-end and a back-end, which communicate over multiple channels, and which both interact with databases. These components are all illustrated in figure 4.1. The following section will briefly overview the functionality of each sub-system.

4.1.1 Front-End

The front-end of Basin leverages the Electron environment (12), a software framework for writing desktop apps using web graphics technologies such as HTML, CSS, and Javascript. Electron incorporates the Chromium browser to render the graphics. File I/O and other system-specific commands are implemented using the Node JS libraries. In addition, Basin uses React (29) libraries for programatically creating HTML elements, along with the Redux (8) framework, which enables vastly simpler updates of React elements triggered by updating program state.

4.1.2 Inter-Process Communication

As mentioned below, the back-end of Basin is written in Python, which presents a natural challenge concerning Inter-Process Communication (IPC) with the front-end. Specifically, the Python back-end runs in a different process (which itself can spawn child processes) than the the front-end, requiring inter-process communication between processes in two different languages.

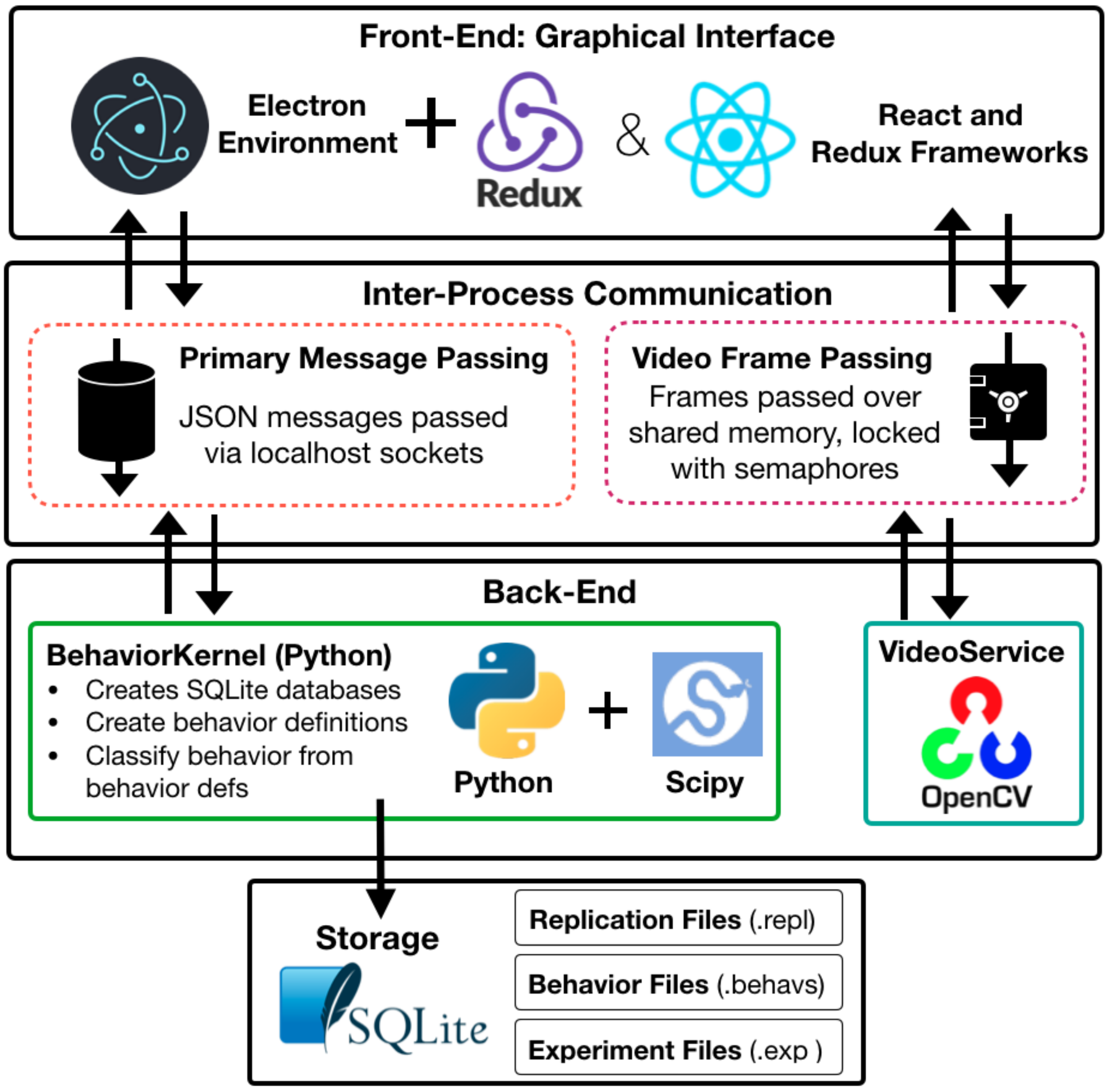
Front-End: Graphical Interface
Electron Environment
Redux
&
React and Redux Frameworks
Inter-Process Communication
Primary Message Passing
JSON messages passed via localhost sockets
Video Frame Passing
Frames passed over shared memory, locked with semaphores
Back-End
BehaviorKernel (Python)
• Creates SQLite databases
• Create behavior definitions
• Classify behavior from behavior defs
Python
Scipy
VideoService
OpenCV
Storage
SQLite
Replication Files (.repl)
Behavior Files (.behavs)
Experiment Files (.exp)

Figure 4.1: Overview of Basin architecture. Citations for logo images can be found, from top left to bottom right in (13; 1; 30; 27; 32; 2; 7).

While there are many potential ways to do IPC between these two components of the application, the one which worked the best is the following. The Javascript code running the front-end calls a Python script which is independent of the back-end, using the child process API provided by Node JS (24). Javascript communicates with this script using I/O streams. IPC is then performed between the front-end Python script and the Python back-end by passing JSON messages over localhost sockets provided by the multiprocessing module's Manager functionality (28).

Basin also features another form of IPC. Video playback in Javascript is generally done through the HTML video tag. Unfortunately, there is no way to currently access frame-level information using this tag (to determine exactly which frame is currently being displayed, or to move to a specific frame within a video). The precision required by Basin playback requires frame-based granularity, making using this tag untenable. Thus, an alternative method to access video had to be identified, and since no other widely used methods for video decoding in Javascript exist, the decoding must be done in another language. The most widely used method for performing video decoding is OpenCV (25), a library written in C++ for video decoding. While Javascript can interface with C++ code through bindings, due to technical reasons, OpenCV functions could not be directly called through this Javascript - C++ interface. Thus, video frames have to be passed from another process to the front-end process. Further, this form of IPC must be fast, because video frames are large, and must be displayed sequentially extremely quickly. Therefore, using the existing IPC detailed above was not feasible, because socket-based transport is too slow.

The solution, then, was to used shared memory. Shared memory is a way of allocating regions of RAM which are simultaneously accessible (readable and writable) to multiple processes, both of which have pointers to the memory location. Of course, shared memory must be partnered with inter-process locking mechanisms to avoid simultaneous reading/writing from both processes, and other similar concurrency issues. The standard approach here is to use semaphores, OS-provided locks which can be accessed from any process with sufficient privileges.

One final point to note here is that because OpenCV is generally dynamically linked to the C++ libraries which call OpenCV library functions, users would have to install OpenCV for Basin to work, if C++/OpenCV code were used directly (static linkage of OpenCV is possible, but in practice was difficult due to the number of dependencies of OpenCV itself). Hence, rather than using C++ code directly, a Python/Open-CV library is used, which can be easily installed using Python package management.

Thus, the IPC for reading and writing video files works as follows. A Python script which is separate from the primary back-end code runs, and decodes video frames from the video file, using OpenCV bindings available in Python. Then, because the shared memory/semaphore locking system calls are not generally available in Python[1] (because these system calls are only available to C), Basin passes video frames decoded in the Python/OpenCV library to Cython code, which is a separate language which allows for mixing of C and Python code in the same file. Thus, C system calls to set up and interface with semaphores and shared memory can be combined with the video frame decoding performed in pure Python.

4.1.3 Back-End and Storage

Here, I discuss the architecture of the back-end, along with a discussion of how the back-end interfaces with the storage system. The back-end of the system, as mentioned above, is written in Python. A controller process in the back-end can receive input passed over the localhost socket IPC described above. Then, depending on the command sent by the front-end, the back-end may spawn a child process to complete the command. This was done to increase speed and avoid situations in which multiple commands sent from the front-end had to wait in a queue to be completed.

Basin creates and stores data primarily in two types of files, experiment (.exp) and replication (.repl) files, which serve as databases to store the measurements computed by Basin. Both files are simply SQLite databases with custom file extensions. Experiment files correspond to the analysis mode of the system described in chapter two, and replication files correspond to the replication mode of the system. Each file can be loaded into the system, and all state from the previous usage of the system will be loaded.

The back-end can perform a variety of operations on the storage files, including construct new experiment and replication files, construct new behavior measurements, edit previously created behavior measurements, perform replications of measurements over multiple input files, and much more. Since multiple processes may be spawned for different jobs in the back-end, a concurrency system exists in which, during crucial sections of operation, child processes can set a locking parameter within the database to prevent other processes from creating inconsistent

[1]Later versions of Python 3 are starting to provide interfaces to shared memory/semaphore system calls, but due to the opacity of the underlying implementation, and given that Basin needs to work cross-platform, using this functionality was untenable.

state.

References

[1] AAMINE1965. The logo of Redux. `https://upload.wikimedia.org/wikipedia/commons/4/49/Redux.png`, 2018. Accessed: 2020-4-3.

[2] Adi Shavit. OpenCV Logo with text. `https://upload.wikimedia.org/wikipedia/commons/thumb/3/32/OpenCV_Logo_with_text_svg_version.svg/831px-OpenCV_Logo_with_text_svg_version.svg.png`, 2006. Accessed: 2020-4-3.

[3] Anderson, D. J., and Perona, P. Toward a science of computational ethology. *Neuron 84*, 1 (2014), 18–31.

[4] Berman, G. J. Measuring behavior across scales. *BMC biology 16*, 1 (2018), 23.

[5] Berman, G. J., Bialek, W., and Shaevitz, J. W. Predictability and hierarchy in Drosophila behavior. *Proceedings of the National Academy of Sciences 113*, 42 (2016), 11943–11948.

[6] Berman, G. J., Choi, D. M., Bialek, W., and Shaevitz, J. W. Mapping the stereotyped behaviour of freely moving fruit flies. *Journal of The Royal Society Interface 11*, 99 (2014), 20140672.

[7] D. Richard Hipp. SQLite Logo. `https://upload.wikimedia.org/wikipedia/commons/thumb/3/38/SQLite370.svg/1024px-SQLite370.svg.png`, 2010. Accessed: 2020-4-3.

[8] Dan Abramov. Getting Started with Redux. `https://reactjs.org/docs/getting-started.html`. Accessed: 2020-2-25.

[9] DeAngelis, B. D., Zavatone-Veth, J. A., and Clark, D. A. The manifold structure of limb coordination in walking drosophila. *eLife 8* (2019).

[10] Doshi-Velez, F., and Kim, B. Towards a rigorous science of interpretable machine learning. *arXiv preprint arXiv:1702.08608* (2017).

[11] Draft, R. W., McGill, M. R., Kapoor, V., and Murthy, V. N. Carpenter ants use diverse antennae sampling strategies to track odor trails. *Journal of Experimental Biology 221*, 22 (2018), jeb185124.

[12] Electron Team. Electron documentation. `https://www.electronjs.org/docs`. Accessed: 2020-2-25.

[13] Electron Team. Electron software framework logo. `https://upload.wikimedia.org/wikipedia/commons/thumb/9/91/Electron_Software_Framework_Logo.svg/1024px-Electron_Software_Framework_Logo.svg.png`, 2019. Accessed: 2020-4-3.

[14] Ey, E., Torquet, N., Lagache, T., Dallongeville, S., Imbert, A., Legou, T., Le, A. S., Faure, P., Bourgeron, T., Olivo-Marin, J., et al. Live Mouse Tracker: real-time behavioral analysis of groups of mice. *BioRxiv preprint* (2018).

[15] Eyjolfsdottir, E., Branson, K., Yue, Y., and Perona, P. Learning recurrent representations for hierarchical behavior modeling. *arXiv preprint arXiv:1611.00094* (2016).

[16] Friedman, J., Hastie, T., and Tibshirani, R. *The elements of statistical learning*, vol. 1. Springer series in statistics New York, 2001.

[17] Garcia, V. A., Junior, C. F. C., and Marino-Neto, J. Assessment of observers' stability and reliability—a tool for evaluation of intra-and inter-concordance in animal behavioral recordings. In *2010 Annual International Conference of the IEEE Engineering in Medicine and Biology* (2010), IEEE, pp. 6603–6606.

[18] Grünwald, P., and Roos, T. Minimum description length revisited. *arXiv preprint arXiv:1908.08484* (2019).

[19] Hsu, A. I., and Yttri, E. A. B-soid: An open source unsupervised algorithm for discovery of spontaneous behaviors. *bioRxiv* (2019), 770271.

[20] Kabra, M., Robie, A. A., Rivera-Alba, M., Branson, S., and Branson, K. JAABA: Interactive machine learning for automatic annotation of animal behavior. *Nature methods 10*, 1 (2013), 64.

[21] Maaten, L. v. d., and Hinton, G. Visualizing data using t-SNE. *Journal of machine learning research 9*, Nov (2008), 2579–2605.

[22] Mathis, A., Mamidanna, P., Cury, K. M., Abe, T., Murthy, V. N., Mathis, M. W., and Bethge, M. DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. *Nature neuroscience 21*, 9 (2018), 1281.

[23] McInnes, L., Healy, J., and Melville, J. Umap: Uniform manifold approximation and projection for dimension reduction. *arXiv preprint arXiv:1802.03426* (2018).

[24] Node JS Team. Node.js v13.9.0 Documentation - Child Process. `https://nodejs.org/api/child_process.html`.

[25] OpenCV Team. Opencv. `https://opencv.org`. Accessed: 2020-2-25.

[26] Pereira, T. D., Aldarondo, D. E., Willmore, L., Kislin, M., Wang, S. S.-H., Murthy, M., and Shaevitz, J. W. Fast animal pose estimation using deep neural networks. *Nature methods 16*, 1 (2019), 117–125.

[27] Python Software Foundation. Python logo. `https://upload.wikimedia.org/wikipedia/commons/thumb/c/c3/Python-logo-notext.svg/1024px-Python-logo-notext.svg.png`, 2008. Accessed: 2020-4-3.

[28] Python Team. multiprocessing — Process-based parallelism. `https://docs.python.org/3.8/library/multiprocessing.html`.

[29] React Team. Getting started. `https://reactjs.org/docs/getting-started.html`. Accessed: 2020-2-25.

[30] React Team. The logo for the React open source JavaScript library. `https://upload.wikimedia.org/wikipedia/commons/thumb/a/a7/React-icon.svg/1280px-React-icon.svg.png`, 2017. Accessed: 2020-4-3.

[31] Rissanen, J. *Information and complexity in statistical modeling.* Springer Science & Business Media, 2007.

[32] SciPy Developers. This is a logo for SciPy. `https://upload.wikimedia.org/wikipedia/en/5/58/Scipylogo.png`, 2008. Accessed: 2020-4-3.

[33] Spink, A., Tegelenbosch, R., Buma, M., and Noldus, L. The EthoVision video tracking system—a tool for behavioral phenotyping of transgenic mice. *Physiology & behavior 73*, 5 (2001), 731–744.

[34] Stephens, G. J., Johnson-Kerner, B., Bialek, W., and Ryu, W. S. Dimensionality and dynamics in the behavior of C. elegans. *PLoS computational biology 4*, 4 (2008).

[35] Tinbergen, N. *The study of instinct.* Clarendon Press/Oxford University Press, 1951.

[36] Todd, J. G., Kain, J. S., and de Bivort, B. L. Systematic exploration of unsupervised methods for mapping behavior. *Physical biology 14*, 1 (2017), 015002.

[37] Wiltschko, A. B., Johnson, M. J., Iurilli, G., Peterson, R. E., Katon, J. M., Pashkovski, S. L., Abraira, V. E., Adams, R. P., and Datta, S. R. Mapping sub-second structure in mouse behavior. *Neuron 88*, 6 (2015), 1121–1135.

Appendix A

Definitions of Primitive Operations

Here, I state mathematical definitions of the Basin primitives. Note that in some cases, the definitions given are only approximated in Basin (e.g. for vector and scalar derivatives). These approximations are noted on the side. The abbreviations given under each primitive's name are used in the main text to define behaviors.

To define Basin primitives, let $\vec{v}_1(t), ..., \vec{v}_K(t)$ denote vector-valued functions with p components, defined over discrete timepoints t, where t is an integer representing the current frame. Let $v_{ij}(t)$ denote the jth component of vector $\vec{v}_i(t)$. Let $a_1(t), ..., a_K(t)$ denote real-valued functions with similar t-indexing, and finally let $b_1(t), ..., b_K(t)$ denote binary-valued functions with range in $\{0, 1\}$. Let $P(t) = ((x_1(t), y_1(t)), ..., (x_K(t), y_K(t)))$ denote a polygon, given by its vertices, real valued two-dimensional coordinates, arranged in counter-clockwise order. Further, let $R(P(t))$ denote the set of points in $\mathbb{R}^2$ bounded by the polygon's points. Let $I(\phi)$ denote the indicator function which is true if the logical expression ϕ is true. Additionally, let $\psi(\vec{v}_i(t), x, y, w, P(t)) = I(((x, y) \in \theta(v_i(t), w)) \wedge ((x, y) \in R(P(t))))$ where $\theta(v_i(t), w)$ denotes the set of real-valued points contained within the octagon with apothem w centered at $v_i(t)$. Let $\kappa(\vec{c}(s)) = \frac{|c'_x(x)c''_y(y) - c''_x(x)c'_y(y)|}{(c'_x(x) + c'_y(y)^{3/2}}$ where $\vec{c}$ is a two-dimensional curve parameterized by s, $(x, y) = \vec{c}(s)$, and c_x, c_y are the x and y coordinates of the curve. Let $\text{atan2}(\vec{a}, \vec{b})$ be the arctangent function which can return values in the range $(-180, 180)$ using two coordinates to determine the quadrant of the angle from vector $\vec{a}$ to vector $\vec{b}$. Let $\gamma(a_i(t), s)$ be the value of the sth percentile of $a_i(t)$, over all t. In each definition box, let the definition of the function be denoted by f.

Table A.1:: Definitions of Primitive Operations

Primitive Name	Definition	Notes
Average (Avg)	$f(\vec{v}_1(t),...,\vec{v}_k(t)) = \frac{1}{K}\sum_{i=1}^{K}\vec{v}_i(t)\text{K}$	Can be defined for any K
Vector Derivative (VDeriv)	$f(\vec{v}_i(t),k) \approx \left(\frac{dv_{i1}(t)}{dt},...,\frac{dv_{ip}(t)}{dt}\right)$	Approximated by a Savitzky - Golay filter, where k is the window size of the polynomial fit.
Scalar Derivative (SDeriv)	$f(a(t),k) \approx \frac{da(t)}{dt}$	Approximated by a Savitzky - Golay filter, where k is the window size of the polynomial fit.
Norm (Norm)	$f(\vec{v}_i(t)) = \sqrt{\sum_{j=1}^{p} v_{ij}(t)^2}$	
Vector Subtraction (VSub)	$f(\vec{v}_i(t),\vec{v}_j(t)) = (v_{i1}(t)-v_{j1}(t),...,v_{ip}(t)-v_{jp}(t))$	
Scalar Subtraction (SSub)	$f(a_i(t),a_j(t)) = a_i(t) - a_j(t)$	
Scalar Sum (SSum)	$f(a_i(t),a_j(t)) = a_i(t) + a_j(t)$	
Scalar Abs. Value (SAbs)	$f(a(t)) = \|a(t)\|$	
Curvature (Curvature)	$f(\vec{v}_1(t),\vec{v}_2(t),\vec{v}_3(t),\vec{v}_4(t)) = \frac{\kappa(\vec{c}(s_1))+\kappa(\vec{c}(s_2))}{2}$	Where $\vec{c}$ is the curve found by fitting a polynomial through $\vec{v}_1,\vec{v}_2,\vec{v}_3,\vec{v}_4$, and s_1, s_2 are the parameters of the curve which yield $\vec{c}(s_1) = \vec{v}_2(t), \vec{c}(s_2) = \vec{v}_3(t)$ The derivatives are calculated by a Savitzky-Golay filter as above.
Angle (Ang)	$f(\vec{v}_i(t),\vec{v}_j(t)) = \arccos\frac{\vec{v}_i(t)\cdot\vec{v}_j(t)}{\|\|\vec{v}_i(t)\|\|\|\|\vec{v}_j(t)\|\|}$	

Table A.1:: Definitions of Primitive Operations

Primitive Name	Definition	Notes
Signed Angle (SignedAng)	$f(\vec{v}_i(t), \vec{v}_j(t)) = \text{atan2}(\vec{v}_{i1}(t)\vec{v}_{j2}(t) - \vec{v}_{i2}(t)\vec{v}_{j1}(t), \vec{v}_{i1}(t)\vec{v}_{j1}(t) + \vec{v}_{i2}(t)\vec{v}_{j2}(t))$	Gives a signed angle in the range of $(-180, 180)$. The formula for this expression can be derived by examining formulas for the cross and dot product.
Projection (Proj)	$f(\vec{v}_i(t), \vec{v}_j(t)) = \frac{\vec{v}_i(t) \cdot \vec{v}_j(t)}{\lvert\lvert\vec{v}_i(t)\rvert\rvert}$	
Cutoff (Cutoff)	$f(a_i(t), c, \phi) = I((a_i(t) > c) \oplus (\phi = 1))$	Where $\oplus$ is the XOR operator, c is a parameter controlling the cutoff used for the function, and ϕ is a parameter controlling whether or not the cutoff should be defined as greater or less than the scalar value.
Percentile Cutoff (PCutoff)	$f(a_i(t), s, \phi) = I((a_i(t) > \gamma(a_i(t), s) \oplus (\phi = 1))$	Everything same as Cutoff, except now comparing to the sth percentile of $a_i(t)$ for all t.
In Region (InRegion)	$f(\vec{v}_i(t), P(t)) = I(\vec{v}_i(t) \in R(P(t)))$	Basin computes this by using a ray-casting algorithm to determine the parity of the number of intersections between the line segment connecting the origin and the vector and the line segments connecting adjacent vertices of the polygon.

Table A.1:: Definitions of Primitive Operations

Primitive Name	Definition	Notes
Area Overlap (AreaOverlap)	$f(\vec{v}_i(t), P(t), w) = \int_x \int_y \psi(\vec{v}_i(t), x, y, w, P(t))dxdy$	Achieved in Basin by computing the number of pixels in the overlap of two masks.
Distance To Region (DistToRegion)	$f(\vec{v}_i(t), P(t)) = \text{argmin}_{(x,y)\in R(P(t))} \|\|\vec{v}_i(t) - (x, y)\|\|_2$	
And (And)	$f(b_i(t), b_j(t)) = I((b_i(t) = 1) \wedge (b_j(t) = 1))$	
Or (Or)	$f(b_i(t), b_j(t)) = I((b_i(t) = 1) \vee (b_j(t) = 1))$	
Not (Not)	$f(b_i(t)) = I(\neg b_i(t))$	
Binary Sum (BSum)	$f(b_i(t), b_j(t)) = b_i(t) + b_j(t)$	The plus operation adds the numbers, so a value of two is possible for this operation.