Ali Osman Berk Şapcı

June 20, 2022

Contents

List of Figures

List of Tables

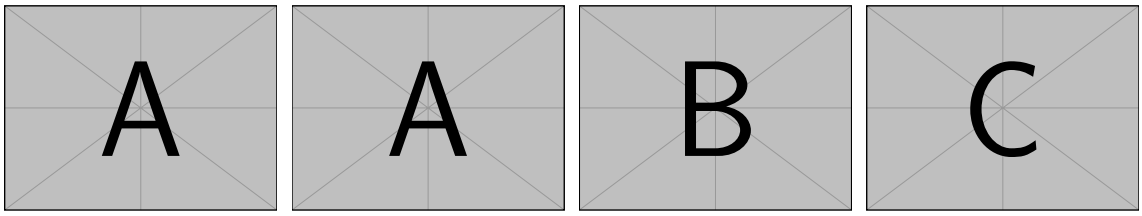
Abstract

A *Bastı* (English: *Basty*, Azerbaijani Turkish: *Basdı*, Anatolian Turkish: *Basdırık*) is an evil spirit in Turkic mythology which rides on people's chests while they sleep, bringing on nightmares. Bastı sits astride a sleeper's chest and becomes heavier until the crushing weight awakens the terrified and breathless dreamer. The victim awakes, unable to move under the Bastıs weight. It may also include lucid dreams.

Chapter 1

Introduction

1.1 Challenges



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

Figure 1.1: Orientations.

1.2 Contributions

Chapter 2

Biological Experiments and Data Collection

Chapter 3

Feature Extraction

3.1 Overview of Feature Extraction

For a single experiment data, i.e., a single fruit fly recorded between ZT10 and ZT2 (zeitgeber time 10 and 2), feature extraction consists of four consecutive steps, where the input in one stage is the output of the previous one. The input of the first step is the raw output signal of the tracking and pose estimation model; in our case, the output of DeepLabCut, a toolbox for markerless pose estimation. The feature extraction steps are as follows:

1. Constructing pose values and preprocessing; dealing with occluded body parts, alignment of different orientations, filtering, and imputation.
2. Computing spatio-temporal features, such as distances between body parts, velocity, and angular velocity from body part positions.
3. Computing dynamic postural features by extending spatio-temporal features to multiple time scales using wavelet transformation and sliding window statistics.
4. Computing normalized high-dimensional behavioral representations.

Matrices $\mathbf{X} \in \mathbb{R}^{T \times N}$ and $\mathbf{Y} \in \mathbb{R}^{T \times N}$ denotes a multivariate time series for x and y cartesian components of two-dimensional video recordings that are collected for N tracked body parts of the fly on T consecutive time stamps by a pose estimation model. This multivariate time series matrices $\mathbf{X}$ and $\mathbf{Y}$ are the raw input data that goes into the first step of the feature extraction. Note that the number of body parts, N, must be the same among all experiments conducted with different fruit flies, but the number of time stamps, T, might differ. Each column of the $\mathbf{X} = \left[\mathbf{x}_1, \cdots, \mathbf{x}_N\right]^\top$ and $\mathbf{Y} = \left[\mathbf{y}_1, \cdots, \mathbf{y}_N\right]^\top$ can be written respectively as follows:

$$\mathbf{y}_i = (y_{i,1}, y_{i,2}, \dots, y_{i,t-1}, y_{i,t}, y_{i,t+1}, \dots, y_{i,T}), \tag{3.1}$$

$$\mathbf{x}_i = (x_{i,1}, x_{i,2}, \dots, x_{i,t-1}, x_{i,t}, x_{i,t+1}, \dots, x_{i,T}). \tag{3.2}$$

Here i denotes the index of the body part, e.g., leg tip or proboscis.

In addition to $\mathbf{X}$ and $\mathbf{Y}$, a pose estimation model may report prediction scores for each tracked body part at each time step, which is the case for DeepLabCut as

well. $L \in \mathbb{R}^{N \times T}$ denotes the time series of prediction scores, each column of the $L = [\mathbf{l_1}, \cdots, \mathbf{l_N}]^\top$ can be written as follows:

$$\mathbf{l}_i = (l_{i,1}, l_{i,2}, \dots, l_{i,t-1}, l_{i,t}, l_{i,t+1}, \dots, l_{i,T}). \tag{3.3}$$

The prediction scores tend to be very low when the body part is not visible. Thus, L provides valuable information about the occluded body parts. In the Section 3.2.1, how L is incorporated into construction of pose values is described in detail.

3.2 Preprocessing

This step involves preprocessing the signal by filtering and imputation of certain video frames. But in addition to this, there are a couple of optional procedures that can be beneficial for our task of learning stereotypical behaviors. These additional procedures deal with occluded body parts of the fly, alignment of the fly orientations and defining new points of interest.

3.2.1 Occluded Body Parts

As mentioned in the Section 1.1, the two-dimensionality of the video recordings introduces several significant challenges, one of which is the occluded body parts. There are many types of occlusions. One type is short occlusions that often results from postural changes. Imputation of the time series $\mathbf{X}$ and $\mathbf{Y}$ for such short occlusions is relatively easy since the number of consecutive missing data points is few. However, this is not the case for long occlusions, which usually occur when the fly is dormant for an extended period of time. Especially for the body parts with left and right counterparts, fly's orientation can result in one of the counterparts being occluded for long dormant epochs. We use imputation and elimination of corresponding data-points to deal with the occlusions. Before describing those approaches, we define a criterion for being occluded.

Oriented Pose Values for Body Parts with Left & Right Counterparts

If the fly is oriented perpendicular to the camera perspective, as in Figure 1.1d, then one of the left and right body parts is often occluded. In other orientations (e.g., Figure 1.1a and Figure 1.1c), both of them or none of them might be occluded as well. However, in the conducted experiments, flies usually choose to stay dormant perpendicular to the camera perspective in long dormant epochs, as mentioned in Chapter 2. In such cases, one can concede using only one of the left and right counterparts to construct pose values. Therefore, this optional step is included in the behavioral mapping pipeline to reduce pose values of body parts with left and right counterparts to a single value.

We use prediction scores to determine which body part should be used to compute oriented pose values. Let i and j be a pair of body parts which are left and right counterparts of each other, e.g., left haltere and right haltere. Then, one can use the $\mathbf{l}_i$ and $\mathbf{l}_j$ to predict if one of them is occluded at a particular time step t. Let $\text{orient}(\mathbf{x}_i, \mathbf{x}_j)$ ($\text{orient}(\mathbf{y}_i, \mathbf{y}_j)$) be a new pose vector that will be computed based

on $\mathbf{x}_i$ and $\mathbf{x}_j$ ($\mathbf{y}_i$ and $\mathbf{y}_j$), e.g., a vector of oriented haltere pose values, composed of left haltere and right haltere pose values. The following conditional procedures are proposed to compute oriented pose values from the left and right pose values by deciding the orientation of the fly for a counterpart body pair. The procedures below can be used separately, or successively.

- If $l_{i,t} - l_{j,t} \geq \epsilon$, then, without loss of generality, $\text{orient}(\mathbf{x}_i, \mathbf{x}_j)_t = x_{i,t}$ and $\text{orient}(\mathbf{y}_i, \mathbf{y}_j)_t = y_{i,t}$ for a threshold ϵ, typically $\epsilon > 0.5$.
- If $\left|\left\{\, t' \,:\, l_{i,t'} > l_{j,t'},\, t' \in [t-\tau \,..\, t+\tau] \,\right\}\right| > \tau$, then, without loss of generality, $\text{orient}(\mathbf{x}_i, \mathbf{x}_j)_t = x_{i,t}$ and $\text{orient}(\mathbf{y}_i, \mathbf{y}_j)_t = y_{i,t}$, for a window of size $2w+1$.
- If $l_{i,t} > l_{j,t}$ and if the nearest confident left orientation is closer than the nearest confident right orientation, i.e.,

$$\arg\min_{t'} \left\{\, |t-t'| \,:\, l_{i,t'} - l_{j,t'} \geq \epsilon \,\right\} > \arg\min_{t'} \left\{\, |t-t'| \,:\, l_{j,t'} - l_{i,t'} \geq \epsilon \,\right\},$$

  then, without loss of generality, $\text{orient}(\mathbf{x}_i, \mathbf{x}_j)_t = x_{i,t}$ and $\text{orient}(\mathbf{y}_i, \mathbf{y}_j)_t = y_{i,t}$.
- If simply $l_{i,t} > l_{j,t}$, then without loss of generality, $\text{orient}(\mathbf{x}_i, \mathbf{x}_j)_t = x_{i,t}$ and $\text{orient}(\mathbf{y}_i, \mathbf{y}_j)_t = y_{i,t}$.

Except directly comparing the prediction confidence scores as in the last procedure, some of the time points might be left with undecided orientations. If the number of such time points is acceptable, then directly comparing the prediction scores for those time points is convenient and handy.

After applying the above procedures for a left and right counterpart pair i and j, we can define oriented multivariate time series as

$$\mathbf{X}^o = \left(\left[(\mathbf{x}_k)_{k \notin \bigcup_{\{i,j\} \in \mathcal{O}} \{i,j\}} \right]^\top \,\middle|\, \left[\big(\text{orient}(\mathbf{x}_i, \mathbf{x}_j)\big)_{\{i,j\} \in \mathcal{O}} \right]^\top \right),$$
$$\mathbf{Y}^o = \left(\left[(\mathbf{y}_k)_{k \notin \bigcup_{\{i,j\} \in \mathcal{O}} \{i,j\}} \right]^\top \,\middle|\, \left[\big(\text{orient}(\mathbf{y}_i, \mathbf{y}_j)\big)_{\{i,j\} \in \mathcal{O}} \right]^\top \right),$$

where $\mathcal{O}$ is the set of index pairs of left and right counterparts and $\bigcup_{\{i,j\} \in \mathcal{O}} \{i,j\}$ is the union of all indexes of such body parts pairs. Applying described procedures for each left and right counterparts results in computing $\mathbf{X}^o$ and $\mathbf{Y}^o$. This oriented versions of original data matrices can be used instead of $\mathbf{X}$ and $\mathbf{Y}$ in the rest of the pipeline, if desired.

Detecting Occlusions Using Prediction Scores & Anomalous Pose Values

Major postural changes and overlapping body parts during movements result in some body parts being occluded for a short interval of time. These types of occlusions may span several frames to several seconds. Since the pose estimation model makes predictions for such data points, it is necessary to detect and process such data points. Our pipeline exploits two indicators for detection: prediction scores and anomalous changes in pose values. We mark the body parts estimated to be occluded and corresponding time intervals, and then the desired imputation method is used to fill those time intervals appropriately.

The following conditions are considered to detect occluded body parts, note that they can be used separately or in combination with each other.

- If the prediction confidence score at time step t is lower than a given threshold ϵ, then the t is marked to be imputed.
- If z-score of the prediction confidence score $l_{i,t}$ computed within a window with size τ, and centered at t, is lower than a given threshold ϵ_z, then the t is marked to be imputed.
- If the second-order gradient of the estimated pose value exceeds a given threshold δ, then the t is marked to be imputed.
- If the difference between estimated pose value at time step t and median of pose values within the window of size τ, centered at time step t, exceeds the threshold δ, then the t is marked to be imputed.

Here, the window sizes and threshold parameters are determined separately for each condition. The conditions described above are useful not only for occluded body parts, but are also beneficial for tackling unnatural and abnormal predictions made by the pose estimation model.

3.2.2 Aligning Different Orientations

It may be desired to align different orientations and postures in some cases. For instance, it is not possible to interpret vertical and horizontal replacements of body parts separately without aligning them into a reference frame. Because flies can position themselves in different orientations while exhibiting similar actions, as it can be seen from the Figure 1.1a and Figure 1.1b.

To rotationally align each frame, we transformed body part coordinate values into an egocentric reference frame centered in the middle of the fly's spine, e.g., a line along the thorax. Then, we performed a linear transformation to all pose values in order to place the spine at the origin, oriented along the y-axis.

Alignment of the frames is an optional step in the behavioral mapping pipeline. It is potentially beneficial and reasonable to perform alignment when the Cartesian pose values of the body parts are included as spatio-temporal features.

3.2.3 Filtering and Imputation

Estimated pose values are highly noisy because of the reasons explained in Section 1.1. Thus, filtering the pose values and imputation of the data points marked as occluded or abnormal is an essential step before proceeding with the rest of the pipeline, which contains stages that are sensitive to noise.

One of the most practical and effective approaches for time series imputation is applying interpolation. We also benefit from univariate interpolation to replace values at marked time points, where the exact algorithm is chosen to be one of the followings; linear interpolation, spline interpolation, forward filling, and backward filling.

After imputation, we apply a median filter with appropriate window size and smooth each $\mathbf{x_i}$ and $\mathbf{y_i}$ separately. Finally, a boxcar filter (moving average) with a relatively small window size is used to filter out rapidly changing signals by averaging. We observed that large window sizes may result in smoothing out some critical signals, which potentially define short-duration low-amplitude behaviors, e.g., switch-like haltere movement behavior.

A Rauch-Tung-Striebel Kalman smoother [Rauch et al., 1965] is employed in the development stage. However, no significant performance improvement is observed, and hence later abandoned due to its computational cost and requirement of setting the various state parameters to reasonable values.

The configuration and actual parameters of the imputation methods and the filters are given in Chapter 6.1, together with some representative examples demonstrating the effect of preprocessing with different configurations.

3.3 Computation of Spatio-temporal Features

After preprocessing of pose values, it is now feasible to go one step further towards learning stereotypical behaviors. Although tracking of relevant body parts and processing corresponding pose values is an essential step for quantifying behavior, a set of coordinate values is not sufficient to represent and capture complex spatio-temporal dynamics of animal behavior. There are thousands of unique postures, and behaviors are not even exhibited by some static set of postures. Instead, they are defined by expressive and meaningful spatio-temporal features such as distances, velocities, angles, and angular velocities. Therefore, one needs to compute such features from the coordinate values of body parts in two-dimensional space.

The second stage of the feature extraction is computation of spatio-temporal features from pose values. Two type of features are computed in this stage, as listed below.

1. **Snapshot features:** Spatio-temporal feature values computed at a snapshot of time, listed as follows:
    - distances,
    - angles,
    - cartesian pose values (i.e., per body part features).
2. **Gradient features:** Spatio-temporal feature values computed based on how snapshot features change over time, listed as follows:
    - change of distances,
    - change of angles (i.e., angular velocities),
    - change of cartesian pose values (i.e., body part velocities).

The gradients of snapshot features are computed using second-order accurate central differences in the interior points. The resulting gradient features have the same shape, i.e., the number of features and the number of time-stamps, as the snapshot features.

3.3.1 Distances Between Body Parts

Given a body part pair (i, j), the distance between them at a time step t is calculated as a usual Euclidean distance, given below,

$$d_t^{i,j} = \sqrt{(x_{i,t} - x_{j,t})^2 + (y_{i,t} - y_{j,t})^2}. \tag{3.4}$$

The corresponding gradient feature, namely the change of distance between body part i and j, is computed using the second-order gradient approximation,

$$\dot{d}_t^{i,j} = \begin{cases} \frac{|d_{t+1}^{i,j} - d_t^{i,j}|}{\Delta t} & \text{if t=0 or t=T,} \\ \frac{|d_{t+1}^{i,j} - d_{t-1}^{i,j}|}{2\Delta t} & \text{otherwise,} \end{cases} \tag{3.5}$$

where Δt is the sampling period, and it is equal to $1/\text{FPS}$ seconds.

3.3.2 Joint Angles Between Body Parts

Given a triplet of body parts (i, j, k), the angle between i and k around j is calculated using the two-argument arctangent function as given below,

$$\omega_t^{i,j,k} = \text{atan2}\left(\det \begin{bmatrix} x_{i,t} - x_{j,t}, x_{k,t} - x_{j,t} \\ y_{i,t} - y_{j,t}, y_{k,t} - y_{j,t} \end{bmatrix}, \begin{bmatrix} x_{i,t} - x_{j,t} \\ y_{i,t} - y_{j,t} \end{bmatrix} \cdot \begin{bmatrix} x_{k,t} - x_{j,t} \\ y_{k,t} - y_{j,t} \end{bmatrix}\right) + \pi. \tag{3.6}$$

Then, similar to the change of distance features, the angular velocities are approximated by

$$\dot{\omega}_t^{i,j,k} = \begin{cases} \frac{|\omega_{t+1}^{i,j,k} - \omega_t^{i,j,k}|}{\Delta t} & \text{if t=0 or t=T,} \\ \frac{|\omega_{t+1}^{i,j,k} - \omega_{t-1}^{i,j,k}|}{2\Delta t} & \text{otherwise.} \end{cases} \tag{3.7}$$

3.3.3 Cartesian Pose Values of Body Parts

Cartesian pose values of a body part i are straightforwardly given by the x and y coordinate values as follows:

$$x_t^i = x_{i,t}, \tag{3.8}$$

$$y_t^i = y_{i,t}. \tag{3.9}$$

Note that for a single body part, two feature values are generated. Corresponding gradient features, namely the body part velocities along each cartesian component, are computed by

$$\dot{x}_t^i = \begin{cases} \frac{x_{t+1}^i - x_t^i}{\Delta t} & \text{if t=0 or t=T,} \\ \frac{x_{t+1}^i - x_{t-1}^i}{2\Delta t} & \text{otherwise,} \end{cases} \tag{3.10}$$

$$\dot{y}_t^i = \begin{cases} \frac{y_{t+1}^i - y_t^i}{\Delta t} & \text{if t=0 or t=T,} \\ \frac{y_{t+1}^i - y_{t-1}^i}{2\Delta t} & \text{otherwise.} \end{cases} \tag{3.11}$$

In order to compute overall two-dimensional velocity of a body part, one can always use the distance between the origin and corresponding body part.

3.3.4 Constructing Spatio-temporal Feature Matrices

Let $\mathcal{C}, \mathcal{D}$, and $\mathcal{A}$ denote the sets of body parts, body part pairs, and body part triplets; respectively defining cartesian pose values, distances, and angles. Similarly, let $\mathcal{C}', \mathcal{D}'$, and $\mathcal{A}'$ denote sets that define sets of respective gradient features. Then snapshot feature matrix S constructed as follows;

$$\mathbf{S} = \Big(\big[(\mathbf{x}^i)_{i \in \mathcal{C}} \big] \;\Big|\; \big[(\mathbf{y}^i)_{i \in \mathcal{C}} \big] \;\Big|\; \Big[(\mathbf{d}^{i,j})_{\{i,j\} \in \mathcal{D}} \Big] \;\Big|\; \Big[(\boldsymbol{\omega}^{i,j,k})_{\{i,j,k\} \in \mathcal{A}} \Big] \Big), \tag{3.12}$$

where the vectors are defined as $\mathbf{x}^i = [x_1^i, \cdots, x_T^i]$, $\mathbf{y}^i = [y_1^i, \cdots, y_T^i]$, $\mathbf{d}^{i,j} = \big[d^{i,j}, \cdots, d_T^{i,j}\big]$, and $\boldsymbol{\omega}^{i,j,k} = \big[\omega_1^{i,j,k}, \cdots, \omega_T^{i,j,k}\big]$.

Similarly, for gradient features, the feature matrix is constructed by concatenating change of distances, angular velocities and body part velocities; given by

$$\mathbf{G} = \Big(\big[(\dot{\mathbf{x}}^i)_{i \in \mathcal{C}'} \big] \;\Big|\; \big[(\dot{\mathbf{y}}^i)_{i \in \mathcal{C}'} \big] \;\Big|\; \Big[\big(\dot{\mathbf{d}}^{i,j}\big)_{\{i,j\} \in \mathcal{D}'} \Big] \;\Big|\; \Big[(\dot{\boldsymbol{\omega}}^{i,j,k})_{\{i,j,k\} \in \mathcal{A}'} \Big] \Big), \tag{3.13}$$

where the vectors are defined as $\dot{\mathbf{x}}^i = [\dot{x}_1^i, \cdots, \dot{x}_T^i]$, $\dot{\mathbf{y}}^i = [\dot{y}_1^i, \cdots, \dot{y}_T^i]$, $\dot{\mathbf{d}}^{i,j} = \big[\dot{d}^{i,j}, \cdots, \dot{d}_T^{i,j}\big]$, and $\dot{\boldsymbol{\omega}}^{i,j,k} = \big[\dot{\omega}_1^{i,j,k}, \cdots, \dot{\omega}_T^{i,j,k}\big]$.

The resulting two feature matrices are $\mathbf{S} \in \mathbb{R}^{T \times (2|\mathcal{C}| + |\mathcal{D}| + |\mathcal{A}|)}$, namely snapshot feature matrix, and $\mathbf{G} \in \mathbb{R}^{T \times (2|\mathcal{C}'| + |\mathcal{D}'| + |\mathcal{A}'|)}$, namely gradient feature matrix. Let N_S denote the number of snapshot features, being equal to $2|\mathcal{C}| + |\mathcal{D}| + |\mathcal{A}|$ and let N_G denote the number of gradient features, which is equal to $2|\mathcal{C}'| + |\mathcal{D}'| + |\mathcal{A}'|$.

3.4 Computation of Dynamic Postural Features

Instantaneous values of spatio-temporal features do not provide a sufficient description of complex postural dynamics of behaviors. Understanding the output of a complex biological system, in our case behavior, can only be achieved by studying multiple time-scales together. Previous studies attempt to search behavioral motifs, e.g., repeated sub-sequences of actions with finite length, within the behavioral time series [Ye and Keogh, 2011, Brown et al., 2013]. However, as Berman et al. [2014] states, this paradigm usually requires problems of temporal alignment and relative phasing between different scales. Alternatively, extending spatio-temporal features to capture postural dynamics at different time-scales eliminate requirements of temporal alignment and motif based analysis. In order to extend the spatio-temporal features to dynamic postural features, we applied wavelet transformation (similar to Berman et al. [2014]) and computed moving statistics at different time-scales (similar to Kabra et al. [2013]), respectively for the snapshot feature set ($\mathbf{S}$) and the gradient feature set ($\mathbf{G}$).

3.4.1 Moving Statistics of Gradient Features

Gradient features only reflect the instantaneous values of velocities with respect to the sampling rate. In order to capture how these values change within a given interval, the moving statistics, e.g., mean and standard deviation, of gradient features are computed with a sliding window approach. Let τ be the window size parameter,

i.e., the timescale of interest, then the moving mean of the corresponding gradient feature $\mathbf{g}_i$ is given by the function μ_τ:

$$\mu_\tau(g_{i,t}) = {}^{1}\!/_{\min\{t+\tau,\, T\} - \max\{t-\tau,\, 1\} + 1} \sum_{t'=\max\{t-\tau,\, 1\}}^{\min\{t+\tau,\, T\}} g_{i,t'}. \tag{3.14}$$

Similarly, the moving standard deviation of a gradient feature $g_{i,t}$ by is computed by σ_τ, as in the below equation.

$$\sigma_\tau(g_{i,t}) = \left({}^{1}\!/_{\min\{t+\tau,\, T\} - \max\{t-\tau,\, 1\} + 1} \sum_{t'=\max\{t-\tau,\, 1\}}^{\min\{t+\tau,\, T\}} \left(\mu_\tau(g_{i,t}) - g_{i,t'}\right)^2 \right)^{1/2} \tag{3.15}$$

Moving statistics feature generation approach is used to learn animal behavior by Kabra et al. [2013], and Marshall et al. [2021] is also included such features into the analysis.

3.4.2 Wavelet Transformation of Snapshot Features

The wavelet domain is a useful representation of postural dynamics due to the following reasons given by Berman et al. [2014], and the proposed spectrogram generation is used by others as well [Marshall et al., 2021, Todd et al., 2017].

- It describes dynamics over multiple time-scales simultaneously by possessing a multi-resolution time-frequency trade-off.
- It eliminates the requirement of precise temporal alignment for capturing periodic behaviors by taking amplitudes of the continuous wavelet transform of each snapshot feature at different scales.

Given a function $s(t)$, continuous wavelet transformation at a frequency $f > 0$ is expressed as the following integral:

$$W_{f,t'}\left[s\left(t\right)\right] = \frac{1}{\sqrt{a\left(f\right)}} \int_{-\infty}^{\infty} s\left(t\right) \, {}^{*} \left(\frac{t - t'}{a(f)} \right) \mathrm{d}t \,, \tag{3.16}$$

where Ψ is the wavelet function and a is a function for converting frequencies to wavelet scale factor. The Morlet wavelet is suitable for describing postural dynamics which is closely related to human perception, both hearing and vision [Daugman, 1985], and is used in the pipeline. The corresponding wavelet function is given by

$$\Psi(t) = \exp\left\{ \frac{t^2}{2} \right\} \cos(w_0 t), \tag{3.17}$$

where w_0 is a non-dimensional parameter. The frequency-to-scale conversion function a for the Morlet wavelet is as follows:

$$a\left(f\right) = \frac{w_0 + \sqrt{2 + w_0^2}}{4\pi f}. \tag{3.18}$$

For the discrete sequence of snapshot feature $\mathbf{s}_i$ with sampling period Δt, $W_{f,t'}[s(t)]$ translates into

$$W_f(\mathbf{s}_i, t') = \frac{1}{\sqrt{a(f)}} \sum_{t=1}^{T} \Delta t s_{i,t}{}^{*} \left(\frac{t - t'}{af} \right), \tag{3.19}$$

where $t', t \in \mathbb{Z}$ and $1 \leq t' \leq T$ [Torrence and Compo, 1998].

Normalization of Wavelet Power Spectrum

In order to ensure that wavelet transforms (Equation 3.19) at each frequency f are directly comparable to each other and to the other transformed time series, the transformation W_f has to be normalized at each frequency f to have unit energy. This normalization for the Morlet wavelet at frequency f is as follows:

$$C(f) = \frac{\pi^{-\frac{1}{4}}}{\sqrt{2a(f)}} \exp\left\{ 1/4 \left(w_0 - \sqrt{w_0^2 + 2} \right)^2 \right\}. \tag{3.20}$$

So resulting normalized transformation, which is also used to generate the spectrogram in Berman et al. [2014], is given by

$$W_f^0(\mathbf{s}_i, t') = \frac{1}{C(f)} |W_f(\mathbf{s}_i, t')|. \tag{3.21}$$

In addition to the above conventionally used normalization, we alternatively adopted the normalization proposed by Liu et al. [2007]. According to this alternative adjustment, the wavelet power spectrum should be equal to the transform coefficient squared divided by the scale it associates.

$$W_f^0(\mathbf{s}_i, t') = \frac{W_f(\mathbf{s}_i, t')^2}{a(f)} \tag{3.22}$$

We observed substantial improvements using this power spectrum (see Section 6.1).

Determining Spectrum Frequencies

We investigate two different approaches for computing a set of frequencies and, we include both of them in the behavior mapping pipeline. One set is dyadically spaced frequencies between $f_{\min}$ and $f_{\max}$ via

$$f_i = f_{\max} 2^{-\frac{i-1}{N_f - 1} \log \frac{f_{\max}}{f_{\min}}}, \tag{3.23}$$

where $f_{\max} = {}^{FPS}/_2$ Hz is the Nyquist frequency.

The other alternative set of frequencies is linearly spaced between $f_{\min}$ and $f_{\max}$ by

$$f_i = f_{\min} + \frac{f_{\max} - f_{\min}}{N_f - 1} i, \tag{3.24}$$

for $i = 1, 2, \ldots, N_f$, and their corresponding wavelet scales are computed by $a(f_i)$.

3.4.3 Constructing Dynamic Postural Feature Tensors

Let $\mathcal{T}_S = \{ {}^1/_{f_{\min}}, \cdots {}^1/_{f_{\max}} \}$ and $\mathcal{T}_G = \{ \tau_{\min}, \cdots, \tau_{\max} \}$ denote the time-scale sets respectively for wavelet transforms of snapshot features and moving statistics of gradient features. Then corresponding feature tensors are given as follows:

$$\mathbf{W} = \left(W_f^0 \left(\mathbf{s}_i, t \right) \right)_{1 \leq t \leq T,\, {}^1/_f \in \mathcal{T}_S,\, 1 \leq i \leq N_S}, \tag{3.25}$$

$$\mathbf{M}_\mu = \left(\mu_\tau \left(g_{i,t} \right) \right)_{1 \leq t \leq T,\, \tau \in \mathcal{T}_G,\, 1 \leq i \leq N_G}, \tag{3.26}$$

$$\mathbf{M}_\sigma = \left(\sigma_\tau \left(g_{i,t} \right) \right)_{1 \leq t \leq T,\, \tau \in \mathcal{T}_G, 1 \leq i \leq N_G}. \tag{3.27}$$

The resulting extended feature tensors of dynamic postural representations are $\mathbf{W} \in \mathbb{R}^{T \times |\mathcal{T}_s| \times N_S}$, $\mathbf{M}_\mu \in \mathbb{R}^{T \times |\mathcal{T}_v| \times N_G}$ and $\mathbf{M}_\sigma \in \mathbb{R}^{T \times |\mathcal{T}_v| \times N_G}$.

3.5 Computation of Behavioral Representations

After applying wavelet transformation or computing moving statistics to extend extracted spatio-temporal features to dynamic postural features, a couple of additional operations are required to continue in the behavior mapping pipeline.

3.5.1 Flattening Dynamic Postural Feature Tensors

As constructed in Section 3.4, dynamic postural feature tensors are $\mathbf{W} \in \mathbb{R}^{T \times |\mathcal{T}_S| \times N_S}$, $\mathbf{M}_\mu \in \mathbb{R}^{T \times |\mathcal{T}_G| \times N_G}$, and $\mathbf{M}_\sigma \in \mathbb{R}^{T \times |\mathcal{T}_G| \times N_G}$. In order to apply manifold learning-based dimensionality reduction algorithms or traditional machine learning algorithms such as decision trees, the last two dimensions of these feature tensors need to be flattened. As a result, feature matrices are $\mathbf{W}^* \in \mathbb{R}^{T \times (N_S |\mathcal{T}_S|)}$, $\mathbf{M}_\mu^* \in \mathbb{R}^{T \times (N_G |\mathcal{T}_G|)}$ and $\mathbf{M}_\sigma^* \in \mathbb{R}^{T \times (N_G |\mathcal{T}_G|)}$ are obtained.

3.5.2 L_1 Normalization of Features

Dynamic postural feature distributions of similar behaviors may differ among flies due to different characteristics such as gender, and being sleep-deprived. Due to the two-dimensional nature of the video recordings, different orientations may cause observing different feature values for the same behavior. In order to have a homogeneous feature space among flies and along the time, at each time step t, L_1 normalization is applied as follows:

$$\hat{\mathbf{w}}_i = \left(\frac{w_{t,i}^*}{\sum_{j=1}^{N_S |\mathcal{T}_s|} w_{t,j}^*} \right)_{1 \leq t \leq T} \quad \text{and} \quad \hat{\mathbf{W}} = \left[(\hat{\mathbf{w}}_i)_{1 \leq i \leq N_S |\mathcal{T}_S|} \right]. \tag{3.28}$$

Similarly, L_1 normalized versions of $\mathbf{M}_\mu^*$ and $\mathbf{M}_\sigma^*$, namely $\hat{\mathbf{M}}_\mu$ and $\hat{\mathbf{M}}_\sigma$ are obtained. Here $\hat{\mathbf{W}}$, $\hat{\mathbf{M}}_\mu$ and $\hat{\mathbf{M}}_\sigma$ are the final multivariate time series of normalized high dimensional behavioral representation of a single experiment data, i.e., single fruit fly recorded between ZT10 and ZT2. Notice that we may treat each time step, that is, the frame, as a discrete probability distribution after L_1 normalization.

Chapter 4

Experiment Outlining

Chapter 5

Behavior Mapping

5.1 Behavioral Embeddings

The dynamic postural features are able to capture many different timescales, however its high-dimensional structure makes it challenging to directly exploit behavioral representations in analysis, learning and visualization. For example, behavioral representation matrix ($\hat{\mathbf{W}}$) computed from dynamic postural features ($\mathbf{W}$) with 20 spatio-temporal features and 25 timescales (i.e., frequency channels) has $20\times25{=}500$ columns. Since the correlation between different spatio-temporal features (e.g., distance between head and proboscis and cartesian pose values of proboscis) and different timescales are often strong, one may expect that the topological structure of the high dimensional behavioral representations ($\hat{\mathbf{W}}$) can be accurately represented in a lower dimensional space. Therefore, we would like to find a low-dimensional embedding that captures the important features of the dataset. The embedding we compute should minimize local distortions, since trajectories pause near a repeatable position whenever a particular stereotyped behaviour is observed [Berman et al., 2014, DeAngelis et al., 2019, Ali et al., 2019].

Many dimensionality algorithms seek to preserve the pairwise distance structure among all the data samples, the well-known examples of such algorithms are PCA [Hotelling, 1933], and MDS [Kruskal, 1964]. Alternatively, some algorithms favor the preservation of local distances over global distance, such as UMAP (Uniform Manifold Approximation & Projection) [McInnes et al., 2020] t-SNE [Maaten and Hinton, 2008], Laplacian Eigenmaps [Belkin and Niyogi, 2003] and LargeVis [Tang et al., 2016]. The latter category of algorithms achieves to preserve important local structures, and helps to improve classification performance when used in combination with learning algorithms where the function is only approximated locally, e.g., k-nearest neighbor classifier [McInnes et al., 2020]. Similarly, it has been shown that manifold learning based dimensionality reduction algorithms can improve the clustering performance [Sainburg et al., 2021]. Moreover, the trade-off of preserving local distances over global distances does not introduce any significant disadvantages in the latter stages of the behavioral pipeline.

We use UMAP for its superior performance in many aspects. McInnes et al. [2020] demonstrates that a k-nearest neighbor classifier trained on UMAP embeddings achieves higher accuracy for large k values, compared to PCA, t-SNE, LargeVis

and Laplacian Eigenmaps since it captures non-local scales markedly better and local scales comparably or better. Thus, it can be argued that UMAP has captured more of the global and topological structure of the benchmark datasets than its counterparts, t-SNE and LargeVis. Another reason for choosing UMAP over other alternatives is that it tends to produce more stable embeddings. It is capable of producing sub-sample embeddings which are very close to the full embedding even for sub-samples of 5% of the dataset, outperforming the results of t-SNE and LargeVis. Finally, computational performance of UMAP scales well with the number of samples, the embedding dimensionality and the data dimensionality, compared to t-SNE LargeVis, and Eigenmaps, resulting significantly lower run-times on various datasets such as COIL-20 [Nene et al., 1996], Fashion-MNIST [Xiao et al., 2017], and GoogleNews [Mikolov et al., 2013]. Run-time performance of the dimensionality reduction algorithm is an important concern in the behavioral mapping pipeline, as the number of samples and the data dimensionality is often of the order of 10^6 and 10^2, respectively.

UMAP and other non-linear dimensionality reduction algorithms that attempt to use a mathematical structure akin to a k-nearest neighbor graph to approximate a manifold, follow a similar basic structure as below [McInnes et al., 2020].

- Graph Construction
    1. Construct a weighted k-nearest neighbor graph.
    2. Apply some transformation on the edge weights to envelop local distances.
    3. Deal with the incompatibility of the local metrics, i.e., disagreeing weights of the edges.
- Graph Layout
    1. Define an objective function that preservers desired characteristics of the k-nearest neighbor graph.
    2. Compute a low dimensional representation by optimizing the defined objective function.

A distance metric is needed to construct a k-nearest neighbor (k-NN) graph. In our case, we normalized the feature matrices as described in Section 3.5.2, feature vectors at each time step sum up to 1, and hence feature vectors can be considered as discrete probability distributions. We use the Hellinger distance [Hellinger, 1909] to quantify the similarity between "discrete probability distributions" of features, which is given by

$$\text{Hellinger}(P, Q) = \frac{1}{\sqrt{2}} \sqrt{\sum_{i=1}^{k} (\sqrt{p_i} - \sqrt{q_i})^2}. \tag{5.1}$$

Then, based on the Hellinger distances, UMAP constructs a weighted k-NN graph using nearest neighbor descent [Dong et al., 2011]. Then, the k-NN graph is modified by making edges directed and defining the new weights using local Riemannian metric of each data point. After this step, there exist two edges with disagreeing weights between nearest neighbors as the local metric differ. UMAP combines those weights

by using t-conorm and the resulting weight can be interpreted as the probability of at least one of the two directed edges exists.

Finally, UMAP uses a force directed graph layout algorithm in low dimensional space where attractive and repulsive forces are defined based on the gradients, optimizing the edge-wise cross-entropy between the original weighted graph and equivalent weighted graph induced by the embeddings. The algorithm proceeds by iteratively applying attractive and repulsive forces at each edge or vertex. Since the "true" graph captures the topology of the source data, the approximated graph also matches the overall topology of the data, and thus corresponds to a good low dimensional representation. A detailed mathematical and algorithmic description of the UMAP algorithm can be found in [McInnes et al., 2020], and skipped here as it goes beyond the scope of this work.

The algorithm described above is an unsupervised method, but can be easily extended to work in a supervised or semi-supervised manner. While we use a proper metric, e.g., Hellinger distance, defining the distance between a set of points, one can also define a simple metric for categorical values to extend UMAP further for supervised and semi-supervised cases. We can obtain a second view on the source data by using a metric where distances for points in the same category, points in different categories, points without a category (for the semi-supervised case) are defined appropriately. A straightforward and simple example would be defining distances as follows: 1 if the points are in the same category, 0 if the points are in different categories, 0.5 if either of the points is uncategorized. One can combine weighted graphs constructed using the two distance metrics (local metrics and defined metric for categorical values), and arrive at a shared view on the data. In the behavioral mapping pipeline, we benefit from unsupervised UMAP, and its supervised and semi-supervised extensions for different purposes.

The behavioral embeddings exploited fall into three different categories, namely "disparate embeddings", "joint embeddings" and "pair embeddings", detailed descriptions and their applications are described respectively in Section 5.1.1, Section 5.1.2, and Section 5.1.3 respectively.

5.1.1 Disparate Embeddings

Supervised

[NOTE: We compute Supervised-UMAP embeddings separately for each experiment. Computing supervised embeddings is only possible for annotated experiments. This is useful for exploring behavioral sub-categories. For instance, one annotation category, e.g. "grooming", can be consisted of two different clusters in the behavioral embedding space. We can further investigate how high-level behavioral annotations contain different but similar behaviors.]

Unsupervised

[NOTE: For each experiment, we compute a usual unsupervised embedding separately.]

5.1.2 Joint Embeddings

Supervised

[NOTE: We compute Supervised-UMAP embeddings of all annotated experiments together. There is no specific benefit or use case (except visualization purposes) for computing this. The resulting embedding is usually not homogeneous in terms of fly experiments. Different flies do not mix well in the behavioral space.]

Unsupervised

[NOTE: We compute a single behavioral embedding using all experiments. The problem with this approach is that embedding space is too crowded, and thus we can not find meaningful and homogeneous clusters right away. This embedding might be useful for visualization purposes.]

5.1.3 Pair Embeddings

[NOTE: This is the novel and most useful embedding approach that we finally utilize to label unannotated experiments using annotated ones. We compute an embedding for each annotated and unannotated pair. For example, if there are N_A annotated experiments and N_U unannotated experiments, we compute $N_A \cdot N_U$ embeddings in total. There are number of advantageous of this approach. Especially when the behavioral repertoire of the annotated and the unannotated are similar, resulting embedding is easy to interpret and use for clustering, etc.]

5.2 Soft Clustering

McInnes et al. [2017] Campello et al. [2013] [NOTE: We always use soft clustering feature of HDBSCAN, since it is very beneficial to have a composite assignment for a data-point. For example, 0.7 grooming, 0.3 proboscis pumping may indicate that those two behaviors are simultaneously exhibited or a combination of both is exhibited etc. We can always take arg max if a categorical label is needed.]

5.2.1 Disparate Clustering

[NOTE: If embedding is a disparate embedding, then we directly cluster each of them separately. If a joint embedding or pair embedding is clustered, then experiments need to be extracted from the embedding space first, and then they need to be clustered separately.]

5.2.2 Joint Clustering

[NOTE: This is only applicable to joint and pair embeddings. We cluster all the experiments in the behavioral embedding together.]

5.2.3 Crosswise Clustering

[NOTE: This is again only applicable to joint and pair embeddings. For joint embeddings, we exclude a subgroup of experiments. For pair embeddings, we exclude one of the pair experiments. Then rest of the embedding is clustered and clusters are formed. Finally, for each excluded embedding, soft cluster membership vectors are computed based on the formed clusters.]

5.2.4 Mapping Clusters to Behavioral Categories

[NOTE: If a clustering contains annotated experiments, we map that clusters in that clustering to a behavioral composition as follows (*subject to change, there are couple of alternatives*)

$$m_\alpha = \frac{\text{number of frames with annotation } \alpha \text{ in the cluster}}{\text{total number of frames with annotation } \alpha}. \tag{5.2}$$

So for each cluster, we end up with a vector $\mathbf{m} = (m_\alpha)$, represent it behavioral composition.]

5.2.5 Computing Behavioral Scores

[NOTE: Behavioral score of unannotated experiment will be computed using clustering membership score and behavioral composition mapping of those clusters. For example, using semi-supervised pair embeddings and crosswise clustering; one can compute behavioral scores for a frame as follows;

$$y_\alpha = \sum_{c=0}^{C} m_\alpha^c \tag{5.3}$$

where C is the number of clusters, $\mathbf{m}^c$ is the behavioral composition of the cluster c. As a result, for each frame, we end up with a behavioral score vector $\mathbf{y} = (y_\alpha)$.]

5.3 Nearest Neighbor Analysis and Classification

One of our ultimate goals is to annotate an experiment using already annotated ones. In previous sections and chapters, we described feature extraction, experiment outlining, and computation of behavioral embeddings. In this section, we describe a novel nearest neighbor based classification approach, which has to tackle following challenges:

- sparsity of behavioral expressions,
- imbalanced distribution of behavioral categories,
- scarcity of annotated experiments, which are laborious to produce,
- variation between experiments.

Our approach consists of two steps, briefly stated as follows:

1. computing behavioral weights of an unannotated experiment, using its semi-supervised pair embeddings for each annotated experiment,

2. combining behavioral weights of the unannotated experiment with an annotated experiment committee by voting.

We compute the behavioral weights with our nearest neighbor based approach by considering sparsity of behavioral expressions and imbalanced distribution of behavioral categories, as described in Section 5.3.1. In the Section 5.3.2, we detail the committee by voting approach, which helps us to deal with variation between experiments by providing multiple "views" on observed behavioral expression that we attempt to annotate. Finally, the post-processing of resulting predictions is described in the Section 5.3.3.

5.3.1 Behavioral Weights

Consider two experiments, an annotated one expt$^+$, and an unannotated one expt$^-$, and their semi-supervised pair embeddings, respectively $\mathbf{E}^+ = \left[\mathbf{e}^+_1, \cdots \mathbf{e}^+_{F^+}\right]$ and $\mathbf{E}^- = \left[\mathbf{e}^-_1, \cdots \mathbf{e}^-_{F^-}\right]$, where F^+ and F^- are the total numbers of data points, for example, frames, estimated as dormant and active. Given the true annotations $\mathbf{y}^+$ of expt$^+$, and K behavioral categories, the goal is to compute $\widehat{\mathbf{b}}_f = \left[\widehat{b}_{f,1}, \cdots \widehat{b}_{f,K}\right]$, representing the weights (in other words, the similarity score) of each behavioral category for expt$^-$, using $\mathbf{E}^+, \mathbf{E}^-$ and $\mathbf{y}^+$.

The procedure start by querying k-nearest neighbors of expt$^-$'s each frame f in the joint behavioral embedding space consisting of $\mathbf{E}^+$ and $\mathbf{E}^-$, using the k-d trees for efficiency [Bentley, 1975]. k-NN (f) denotes the set of indices of $\mathbf{e}^-_f$'s k nearest neighbors, and the k-NN weight $b_{f,i}$ for each query point (i.e., frame) f of expt$^-$, and behavioral category i, is computed by

$$b_{f,i} = \begin{cases} \displaystyle\sum_{f' \in k\text{-NN}_i(f)} \frac{1}{\mathrm{d}\left(\mathbf{e}^-_f, \mathbf{e}^+_{f'}\right)^p + \epsilon} & \text{if } |k\text{-NN}_i(f)| \neq 0, \\ 0 & \text{if otherwise,} \end{cases} \qquad \text{where} \quad p \in \{0, 1, 2\}. \tag{5.4}$$

Here, k-NN$_i(f) = \left\{ f' \,:\, y^+_{f'} = i, \text{ and } f' \in k\text{-NN}(f) \right\}$, is the set of indices of data points of expt$^+$ whose annotation is the behavior category i and is one of the k nearest neighbors of $\mathbf{e}^-_f$. $\mathrm{d}\left(\mathbf{e}^-_f, \mathbf{e}^+_{f'}\right)$ is the euclidean distance between $\mathbf{e}^-_f$ and $\mathbf{e}^+_{f'}$, and p parameterizes the relation between distance and weight $b_{f,i}$. We add a small number ϵ (10^{-6}) to the denominator to avoid numerical errors. The resulting vector $\mathbf{b}_f = \left[b_{f,1}, \cdots, b_{f,K}\right]$ weights the similarity of the frame f to each behavioral category in the shared embedding space based on nearest neighbors.

Naturally, the number of occurrences or durations of the behavior bouts are different for each behavioral category, and therefore, $\mathbf{y}^+$ is highly imbalanced. As a result, number of nearest neighbors and $\mathbf{b}_f$ are biased in favor of frequently occurring and long-bout behaviors. For instance, pumping-like movements of the proboscis occur more frequently in longer bouts than switch-like movements of the haltere. Especially when k is large, it becomes crucial to consider the imbalanced distribution of behavior occurrences, since the embedding space will be dominated by frequent behaviors. Thus, incorporating this imbalance into the formulation may help to improve the recall of rarely occurring short-bout behaviors and precision of frequently occurring long-bout behaviors. To achieve this, we normalize the scores previously

computed, $b_{f,i}$, as a function of the number of occurrences of the behavioral category i as follows:

$$b'_{f,i} = \frac{b_{f,i}}{(1+N_i^+)^p} \quad \text{or} \quad \frac{b_{f,i}}{\log_k(1+N_i^+)} \quad \text{where} \quad p \in \{\,0, 1/2, 1\,\}, k \in \{\,2, 10\,\}, \tag{5.5}$$

where $N_i^+ = \left|\left\{\, f \,:\, y_f^+ = i \,\right\}\right|$ is the number of frames annotated as behavioral category i. In the above equation, two different alternatives are given for this normalization step; a polynomial one and a logarithmic one, where p and k parameterize the relation between N_i^+ and $b'_{f,i}$. For instance, if one is mostly interested in achieving high recall for frequently occurring behaviors, low p values or using the logarithmic alternative might be more appropriate. It may be even desired to set $p = 0$, and not considering the number of occurrences in some cases, see Section 6.1 for more details.

The resulting vector $\mathbf{b}'_f \in \mathbb{R}^K$ is dependent on the annotated experiment expt^+, and the vectors computed based on different annotated experiments are not comparable with each other. Hence, we map the values of $b'_{f,i}$ to $[0, 1]$ using either the softmax function or L_1 normalization as follows:

$$\hat{b}_{f,i} = \frac{\exp\left\{b'_{f,i}\right\}}{\sum_{j=1}^{K} \exp\left\{b'_{f,j}\right\}} \quad \text{or} \quad \frac{b'_{f,i}}{\sum_{j=1}^{K} b'_{f,j}}. \tag{5.6}$$

The resulting behavioral weight vector $\hat{\mathbf{b}}_f \in [0,1]^K$ can be considered as a probability distribution. Here, the vector $\hat{\mathbf{b}}_f$ represents the behavioral characteristics of the frame f of expt^- based on the behavioral repertoire of expt^+. The voting-like scheme, as described in Section 5.3.2, incorporates the behavioral weight vectors of all annotated experiments to finalize the classification for expt^-.

5.3.2 Experiment Committee by Voting

Consider all experiments: unannotated experiments $\text{expt}_1^-, \cdots, \text{expt}_{R^-}^-$, and annotated experiments $\text{expt}_1^+, \cdots, \text{expt}_{R^+}^+$, where R^- and R^+ are the number of experiments, respectively. The goal is to combine the behavioral weights of an unannotated experiment expt_k^-, calculated separately for each annotated experiment.

Let $\hat{\mathbf{b}}_f^{k,l}$ denote the behavioral weights of expt_k^- computed with expt_l^+. Each annotated experiment contributes to the overall behavioral score of expt_k^-; in other words, a committee consisting of annotated experiments vote according to $\hat{\mathbf{b}}_f^{k,l}$. Before describing hard voting and soft voting approaches, there is one more step to discuss.

There exists a significant variation among the exhibited behavioral repertoires in the experiments. An annotated experiment might lack some behavioral expressions or manifest some behaviors excessively. In such cases, if the behavioral weight vector $\hat{\mathbf{b}}_f^{k,l}$ is not confident, i.e., weights of behavioral categories are close to each other, one may want to decrease its contribution to the voting. To achieve this, we propose two optional approaches; penalizing the behavioral weights with the entropy of the

"probability distribution" $\widehat{\mathbf{b}}_f^{k,l}$, or with the uncertainty. The contribution of votes of expt_l^+ to the committee formed for expt_k^- is $\mathbf{v}_f^{k,l} = \left[v_{f,1}^{k,l}, \cdots, v_{f,K}^{k,l}\right]$, and is given by

$$v_{f,i}^{k,l} = \left(\log_2(K) - \mathrm{H}\left(\widehat{\mathbf{b}}_f^{k,l}\right)\right) \widehat{b}_{f,i}^{k,l} \quad \text{or} \quad \left(1 - \max_{1 \le j \le K} \widehat{b}_{f,j}^{k,l}\right) \widehat{b}_{f,i}^{k,l} \quad \text{or} \quad \widehat{b}_{f,i}^{k,l}. \tag{5.7}$$

If $\max_i \widehat{b}_{f,i}$ is close to $1/K$, which means that the computed vector weights the behaviors uniformly, then the factors $\left(\log_2(K) - \mathrm{H}\left(\widehat{\mathbf{b}}_f^{k,l}\right)\right)$ or $\left(1 - \max_{1 \le j \le K} \widehat{b}_{f,j}^{k,l}\right)$ may be used to decrease the "importance" of the vote.

Now, after computing behavioral votes for each annotated and unannotated experiment pair, we can combine those votes for a single unannotated experiment expt_k^- by forming a committee of annotated experiments $\text{expt}_1^+, \cdots, \text{expt}_{R^+}^+$. The predicted behavioral category at frame k of expt_k^-, denoted by $\widehat{y}_f^k$, is given by combined votes of the committee. To achieve this, we can follow two alternative voting schemes, namely hard voting (i.e., majority rule voting) or soft voting.

Hard Voting

In the hard voting scheme, $\widehat{y}_f^k$ is given by the majority behavioral category of the arg max of the each annotated experiment's votes at frame f, and computed by the following formula:

$$\widehat{y}_f^k = \underset{1 \le i \le K}{\arg\max} \left\{ \underset{1 \le j \le K}{\arg\max}\, v_{f,j}^{k,l} \;:\; j = i,\, 1 \le l \le R^+ \right\}. \tag{5.8}$$

Soft Voting

In contrast to majority voting (hard voting), the soft voting scheme assigns the behavioral category as the arg max of the sum of each annotated experiment's vote vector, and $\widehat{y}_f^k$ is given by

$$\widehat{y}_f^k = \underset{1 \le i \le K}{\arg\max} \sum_{l=1}^{R^+} v_{f,i}^{k,l}. \tag{5.9}$$

5.3.3 Post-processing

Chapter 6

Results

6.1 Employing Proposed Pipeline for Collected Data

6.2 Analyzing Behavioral Repertoire of Asleep Fruit Fly

Chapter 7

`basty`: A Software Package for Automated Behavioral Analysis of Asleep Fruit Fly

Chapter 8

Conclusion

Bibliography

H. E. Rauch, C. T. Striebel, and F. Tung. Maximum likelihood estimates of linear dynamic systems. *AIAA Journal*, 3(8):1445–1450, August 1965. ISSN 0001-1452. doi: 10.2514/3.3166. URL https://ui.adsabs.harvard.edu/abs/1965AIAAJ...3.1445R/abstract.

Lexiang Ye and Eamonn Keogh. Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. *Data Mining and Knowledge Discovery*, 22(1):149–182, January 2011. ISSN 1573-756X. doi: 10.1007/s10618-010-0179-5. URL https://doi.org/10.1007/s10618-010-0179-5.

André E. X. Brown, Eviatar I. Yemini, Laura J. Grundy, Tadas Jucikas, and William R. Schafer. A dictionary of behavioral motifs reveals clusters of genes affecting Caenorhabditis elegans locomotion. *Proceedings of the National Academy of Sciences*, 110(2):791–796, January 2013. doi: 10.1073/pnas.1211447110. URL https://www.pnas.org/doi/abs/10.1073/pnas.1211447110. Publisher: Proceedings of the National Academy of Sciences.

Gordon J. Berman, Daniel M. Choi, William Bialek, and Joshua W. Shaevitz. Mapping the stereotyped behaviour of freely moving fruit flies. *Journal of The Royal Society Interface*, 11(99):20140672, October 2014. doi: 10.1098/rsif.2014.0672. URL https://royalsocietypublishing.org/doi/full/10.1098/rsif.2014.0672. Publisher: Royal Society.

Mayank Kabra, Alice A. Robie, Marta Rivera-Alba, Steven Branson, and Kristin Branson. JAABA: interactive machine learning for automatic annotation of animal behavior. *Nature Methods*, 10(1):64–67, January 2013. ISSN 1548-7105. doi: 10.1038/nmeth.2281. URL https://www.nature.com/articles/nmeth.2281. Number: 1 Publisher: Nature Publishing Group.

Jesse D. Marshall, Diego E. Aldarondo, Timothy W. Dunn, William L. Wang, Gordon J. Berman, and Bence P. Ölveczky. Continuous Whole-Body 3D Kinematic Recordings across the Rodent Behavioral Repertoire - SI. *Neuron*, 109(3):420–437.e8, February 2021. ISSN 08966273. doi: 10.1016/j.neuron.2020.11.016. URL https://linkinghub.elsevier.com/retrieve/pii/S0896627320308941.

Jeremy G. Todd, Jamey S. Kain, and Benjamin L. de Bivort. Systematic exploration of unsupervised methods for mapping behavior. *Physical Biology*, 14(1):015002, February 2017. ISSN 1478-3975. doi: 10.1088/1478-3975/14/1/015002. URL https://doi.org/10.1088/1478-3975/14/1/015002. Publisher: IOP Publishing.

John G. Daugman. Uncertainty relation for resolution in space, spatial frequency,

and orientation optimized by two-dimensional visual cortical filters. *JOSA A*, 2 (7):1160–1169, July 1985. ISSN 1520-8532. doi: 10.1364/JOSAA.2.001160. URL https://opg.optica.org/josaa/abstract.cfm?uri=josaa-2-7-1160. Publisher: Optica Publishing Group.

Christopher Torrence and Gilbert P. Compo. A Practical Guide to Wavelet Analysis. *Bulletin of the American Meteorological Society*, 79(1):61–78, January 1998. ISSN 0003-0007, 1520-0477. doi: 10.1175/1520-0477(1998)079<0061:APGTWA>2.0.CO;2. URL http://journals.ametsoc.org/doi/10.1175/1520-0477(1998)079<0061:APGTWA>2.0.CO;2.

Yonggang Liu, X. San Liang, and Robert H. Weisberg. Rectification of the Bias in the Wavelet Power Spectrum. *Journal of Atmospheric and Oceanic Technology*, 24(12):2093–2102, December 2007. ISSN 1520-0426, 0739-0572. doi: 10.1175/2007JTECHO511.1. URL http://journals.ametsoc.org/doi/10.1175/2007JTECHO511.1.

Brian D DeAngelis, Jacob A Zavatone-Veth, and Damon A Clark. The manifold structure of limb coordination in walking Drosophila. *eLife*, 8:e46409, June 2019. ISSN 2050-084X. doi: 10.7554/eLife.46409. URL https://elifesciences.org/articles/46409.

Mohammed Ali, Mark W. Jones, Xianghua Xie, and Mark Williams. TimeCluster: dimension reduction applied to temporal data for visual analytics. *The Visual Computer*, 35(6-8):1013–1026, June 2019. ISSN 0178-2789, 1432-2315. doi: 10.1007/s00371-019-01673-y. URL http://link.springer.com/10.1007/s00371-019-01673-y.

H. Hotelling. Analysis of a complex of statistical variables into principal components. *Journal of Educational Psychology*, 24(6):417–441, September 1933. ISSN 0022-0663. doi: 10.1037/h0071325. URL https://search.ebscohost.com/login.aspx?direct=true&db=pdh&AN=1934-00645-001&site=eds-live. Publisher: Warwick & York.

J. B. Kruskal. Multidimensional scaling by optimizing goodness of fit to a nonmetric hypothesis. *Psychometrika*, 29(1):1–27, March 1964. ISSN 0033-3123, 1860-0980. doi: 10.1007/BF02289565. URL http://link.springer.com/10.1007/BF02289565.

Leland McInnes, John Healy, and James Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *arXiv:1802.03426 [cs, stat]*, September 2020. URL http://arxiv.org/abs/1802.03426. arXiv: 1802.03426.

Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008. ISSN 1533-7928. URL http://jmlr.org/papers/v9/vandermaaten08a.html.

Mikhail Belkin and Partha Niyogi. Laplacian Eigenmaps for Dimensionality Reduction and Data Representation. *Neural Computation*, 15(6):1373–1396, June 2003. ISSN 0899-7667. doi: 10.1162/089976603321780317. Conference Name: Neural Computation.

Jian Tang, Jingzhou Liu, Ming Zhang, and Qiaozhu Mei. Visualizing Large-scale

and High-dimensional Data. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 287–297, Republic and Canton of Geneva, CHE, April 2016. International World Wide Web Conferences Steering Committee. ISBN 978-1-4503-4143-1. doi: 10.1145/2872427.2883041. URL https://doi.org/10.1145/2872427.2883041.

Tim Sainburg, Leland McInnes, and Timothy Q. Gentner. Parametric UMAP Embeddings for Representation and Semisupervised Learning. *Neural Computation*, 33(11):2881–2907, October 2021. ISSN 0899-7667. doi: 10.1162/neco_a_01434. URL https://doi.org/10.1162/neco_a_01434.

Sameer A Nene, Shree K Nayar, and Hiroshi Murase. Columbia Object Image Library (COIL-20). Technical Report CUCS-005-96, Columbia University, February 1996. URL http://www.cs.columbia.edu/CAVE/software/softlib/coil-20.php.

Han Xiao, Kashif Rasul, and Roland Vollgraf. Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms, September 2017. URL http://arxiv.org/abs/1708.07747. Number: arXiv:1708.07747 arXiv:1708.07747 [cs, stat].

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, Red Hook, NY, USA, December 2013. Curran Associates Inc.

E. Hellinger. Neue Begründung der Theorie quadratischer Formen von unendlichvielen Veränderlichen. *Journal für die reine und angewandte Mathematik*, 1909(136):210–271, July 1909. ISSN 1435-5345. doi: 10.1515/crll.1909.136.210. URL https://www.degruyter.com/document/doi/10.1515/crll.1909.136.210/html?lang=en. Publisher: De Gruyter.

Wei Dong, Charikar Moses, and Kai Li. Efficient k-nearest neighbor graph construction for generic similarity measures. In *Proceedings of the 20th international conference on World wide web - WWW '11*, page 577, Hyderabad, India, 2011. ACM Press. ISBN 978-1-4503-0632-4. doi: 10.1145/1963405.1963487. URL http://portal.acm.org/citation.cfm?doid=1963405.1963487.

Leland McInnes, John Healy, and Steve Astels. hdbscan: Hierarchical density based clustering. *Journal of Open Source Software*, 2(11):205, March 2017. ISSN 2475-9066. doi: 10.21105/joss.00205. URL https://joss.theoj.org/papers/10.21105/joss.00205.

Ricardo J. G. B. Campello, Davoud Moulavi, and Joerg Sander. Density-Based Clustering Based on Hierarchical Density Estimates. In Jian Pei, Vincent S. Tseng, Longbing Cao, Hiroshi Motoda, and Guandong Xu, editors, *Advances in Knowledge Discovery and Data Mining*, Lecture Notes in Computer Science, pages 160–172, Berlin, Heidelberg, 2013. Springer. ISBN 978-3-642-37456-2. doi: 10.1007/978-3-642-37456-2_14.

Jon Louis Bentley. Multidimensional binary search trees used for associative searching. *Communications of the ACM*, 18(9):509–517, September 1975. ISSN 0001-

0782. doi: 10.1145/361002.361007. URL https://doi.org/10.1145/361002.361007.