Segmenting behaviour of free roaming mice with Autoregressive Hidden Markov Models

Frans Norden **Valter Lundegårdh** **Magnus Svensson Pierrau** **Jake Kelly**

Abstract

In this project behaviour of free roaming mice is analysed through feature tracking with a Deep Neural Network and segmentation into states using an Autoregressive Hidden Markov Model (ARHMM). By analysing the time series of mouse behavior, we found that the mouse movements can be segmented in epochs of duration 200-400 ms. We therefore forced the ARHMM model, through regularisation, to find states on that time scale. This lead us to finding between 9 and 11 discrete states in an 8 minute long video of a free roaming mouse. We proved the legitimacy in this through analysing the cross likelihood between states, visually inspecting the videos with assigned states and the state trajectories in parameter space. We furthermore compared the states from two different animals and found that animals show similar states.

Acknowledgement

We would like to thank our fantastic supervisors Arvind Kumar and Andreas Kardamakis for fruitful discussions and guidance, as well as to our contact at KI and friend for life Dimitrios who worked hard to help us in producing the videos and providing interesting ideas! Furthermore, we would like to thank Axel Dellenby for technical support with the Arduino hardware and Espen Teigen for the software implementation of the system interrupt script.

Project report for DD2430

1 Introduction

The way animals behave are in many ways essential to our own existence. If we are not able to understand how invasive species work, our supplies could be scavenged or diseases may spread more easily. If we overlook the capabilities of predatory animals we could fall victim to it. If our cattle are not well taken care of we could lose an indispensable source of nutrition. Our own well-being, in many ways, depends on how the creatures we share this planet with, are understood. Even further, studying the behaviour in other animals can help us in understanding how we ourselves function. Studying mice behaviour in particular, can be favourable in many ways including, but not limited to, medicine and neuroscience. But what actually constitutes animal behaviour?

According to one of the first books published on the subject by Niko Tinbergen, one of the founding fathers of modern ethology, animal behaviour is essentially the movement which animals make, or in other words "[...] a symphony of muscle contractions.". This itself means that the study of behaviour, according to Tinbergen, includes "[...] how the behaviour machinery works, how it develops during the life of the individual, and how animals have evolved their behaviour machinery through generations." [1].

At the Kardamakis group in Karolinska Institutet (KI) a current goal is to better understand how the behaviour machinery works of behaviour. More precisely they want to understand how the brain incorporates additional information during decision making. An experiment, which can repeatedly expose a mouse to external stimulus and record the changes in behaviour, is one way of accomplishing this. However, to accomplish this experiment it is necessary to create an objective model for the behaviour in mice to be able to capture the potential changes in behaviour. Using various machine learning techniques, Wiltschko et al. [2] were able to show that mouse behaviour exhibits sub-second structure, and can be reliably modelled as a time series of discrete behavioural states or motifs.

In this project we found that mouse behaviour exists on a time scale with mean duration of 200-400 ms. Furthermore, we found that we could identify 9 clear states in an 8 minute long video sequence of a free roaming mouse. That model could then be used to detect similar behaviour in another mouse with high accuracy.

1.1 Purpose

The purpose of this project is to get an insight of how mice decision-making process, or in other words their behaviour, can be modelled.

1.2 Goals

The goal of this project is to create a pipeline to detect and track the mouse in real time and classify the mouse's behaviour into distinct motor motifs offline.

1.3 Problem description

To achieve this high-level goal presented in 1.2 we break it down into smaller low-level problems.

To detect and track a mouse we require an algorithm that can be applied to video data of the mouse. To fulfill the requirement of real time analysis the algorithm must be computationally efficient.

Furthermore, to fully capture the behaviour of a mouse we are required to extract a meaningful kinematic profile from the extracted data. This includes choosing variables which carry sufficient information about behaviour.

From the chosen variables above a method to extract them is needed. In addition, we require a model which is able to interpret a time series of the kinematic profile and segment the series into meaningful behavioural states.

Last but not least a method to evaluate the reliability of our findings to verify that our model accurately describes mouse behaviour is required.

2 Related Work

Automatic analysis of behavior in freely behaving animals has only become possible recently. The research field of animal behaviour classification have in many papers been divided into two subtasks

[2, 3, 4, 5, 6]. The first subtask has been pose estimation from video footage, which, with the aid of deep learning methods, is possible to perform non-invasively. The second is behaviour classification where both supervised and unsupervised learning methods can be used to segment behaviour in a way that a human cannot reliably do while also avoiding confirmation bias.

2.1 Pose estimation

To better understand animal behaviours a typical idea has been to record videos of creatures moving around with reflective markers attached to the animals. However, markers are invasive and may affect the behaviour. It would be preferable to use a non-invasive pose estimation. This is exactly what DeepLabCut (DLC) has enabled [7].

DLC is a software toolbox designed to assist researchers in tracking body parts of animals. It is based on transfer learning that utilities deep convolutional networks, pre-trained on ImageNet [8, 9]. This minimises the amount of required labelled training data and, given only a few hundred labelled frames, it can predict marker location, given an unmarked frame, up to a few pixels.

An alternative approach to markerless pose estimation was taken in sub second behaviour paper where the authours utilised the extracted latent variables of a variational autoencoder (VAE) applied to the entire video frame [2]. This proved to be an efficient way to reduce dimensionality in complex data, such as videos of animals. A drawback is however that it can be hard to deduce what the latent variables actually represent, compared to DLC in which we get specific coordinates for chosen markers.

2.2 Segmentation of animal behaviour into discrete states

To further understand the behaviour of animals many efforts have been taken to identify discrete and stereotypical states in time series containing animal behaviour.

In the paper by Wiltschko et al. an in-depth analysis of how to segment behaviour was done. After applying a Variational Autoencoder (VAE) the authors used the generated low dimensionality latent data to conduct a comparison study between different models ability to segment the data into behaviours [2, 3].

In their study they found that an Auto Regressive Hidden Markov Model (ARHMM), followed by the Autoregressive Mixture Model best modelled the captured behavioural data. They confirmed their findings by recreating a video of an artificial mouse from the ARHMM states which was close to indistinguishable from a real mouse. This was accomplished by analysing the cross likelihood and examining how well the model generalised to other mice. Batty et al. provided further evidence to support the accuracy of this method by correlating the results with neural activity of the mouse [5]. In the paper by Luxem et al. the authors also employed a VAE, but instead of applying it on their image data, as in the sub second behaviour and behavenet paper, they extracted the pose of the mouse with DeepLabCut and then applied a VAE on that data [4, 2, 5]. Furthermore, instead of using Markov Model or a Gaussian Model on the extracted latent variables, Luxem et al. used a Recurrent Neural Network together with Gated Recurrent Units on the latent variables to segment the behaviour as a function of time. They showed that they had found behaviours by projecting the extracted variables down to a 2D space with Unifold Manifold Approximation where their variables create distinguishable clusters. In this project we utilised the ARHMM as in Wiltschko et al. and the DLC tracking as in Mathis A. et al. but instead of applying a VAE we performed manual variable extraction [2, 7].

Yet another approach was taken in Hong et al. where they were able to accurately classify some behaviours in mice [6]. They began with fitting ellipses to the mice to extract pose information. This together with labelled videos, roughly 500.000 frames, starting when the mice were in a certain behavioural state, was then sent into a random decision forest of 200 random trees [10]. The investigated behaviours were accurately identified by the classifier with two of the three behaviours having a 99 percent accuracy and the third a 92 percent accuracy. In this project we will instead of labeling 500.000 frames utilise transfer learning via DLC [7].

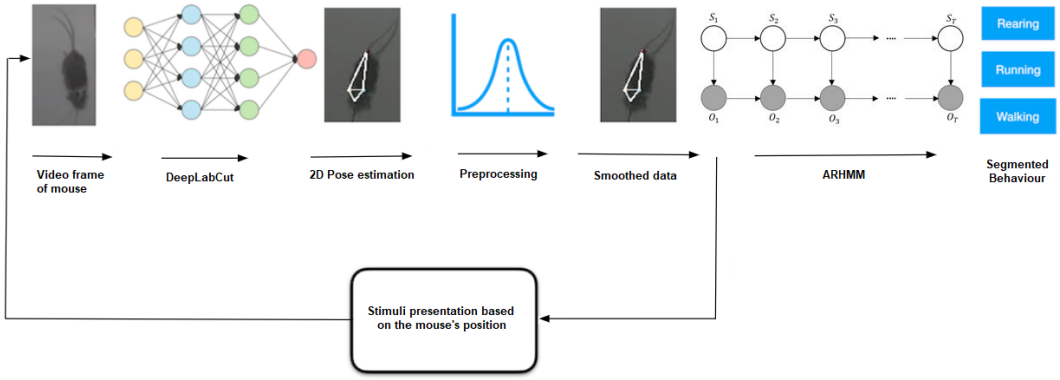


Figure 1: Flowchart over the data processing pipeline

3 Methods

The workflow used in this project, to segment and classify behavioural states, is illustrated in figure 1. The method was a combination of the approaches taken in two recent publications [2, 4]. By extracting pose information using DLC and smoothing this data with a Kalman filter in addition to further pre-processing of the data described in 5.1, we obtained three translationally invariant variables; speed, body length and yaw, i.e the head angle relative to the body. The reason for why these were picked is described in section 3.3. The behaviour was then segmented into states with an ARHMM.

3.1 Pose estimation with DeepLabCut

We used DLC to track the mouse and estimate it's pose [7]. We tracked the base of the mouse's tail, both ears and the snout. It would not be possible to extract such coordinates if we used the approach taken by Wiltschko et al. due to the difficulty in interpreting the latent variables of the VAE [2].

In addition, as we wanted an accurate estimation of where these body parts were, while not using invasive methods that could affect the mouse, DLC was a natural fit. Furthermore, the amount of labelled frames required to achieve good performance was also within a reasonable boundary compared to the 500k in [6].

The deep neural network that DLC is built upon previous algorithm i.e. DeepCut [11] and DeeperCut [12]. It uses deconvolutional layers as its output layer which creates probability densities for each marker. That is, for each new frame it is given as training input, it adjusts its weights iteratively, based on a cross-entropy loss function between the predicted score-map and the ground truth score. When it is given a new frame it chooses the position where the generated probability is the highest for its estimate, as long as that probability is higher than some cut off value p_{cutoff}. Thus, when the algorithm is not confident enough it will simply avoid registering a certain marker.

Both DeepCut [11] and DeeperCut build upon the ResNet architecture which is a convolutional neural network architecture that utilises residual blocks in order to avoid the vanishing gradient problem [11, 12, 13, 14]. This makes it possible to train a network that is much deeper. Training such a network from scratch is however computationally expensive and DLC therefore utilises transfer learning with models pretrained on ImageNet [9].

In order to present stimuli relative to the mouse, its position was required in real time, which was achievable due to a recent update to DLC, namely DLC-live [15]. We did this in anticipation of reaching a second goal which was not achieved, more can be read about it in section 8.6.

3.2 Kalman filtering

In order to smooth the coordinates generated by DLC a Kalman filter [16] was applied to the data. This was done to make the data less noisy. More details about the effect the filtering had can be found in section 5.1. The filter contains a state space motion model of the mouse which it compares through the Kalman gain with the measured coordinates. The state space model of an object in

Cartesian coordinates are given by equation 1 where T governs the allowed speed of the mouse in the model. This matrix is multiplied with the vector $\mathbf{a} = [x, y, \dot{x}, \dot{y}]^T$ which represent the location in x,y coordinates and its corresponding speed.

$$A = \begin{bmatrix} 1 & 0 & T & 0 \\ 0 & 1 & 0 & T \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{1}$$

3.3 Extracting kinematic profile

To obtain pose dynamics of behaviour over time, certain pose features were extracted from the smoothed coordinate data to create a kinematic profile.

Firstly, since behaviour can be expressed as sequences of short, repeated modules of movement, it was important that we chose features that would allow for repeated trajectories in pose space [2, 5, 4]. In particular, this meant choosing features that were invariant to both translation and rotation, so that a repeated behavioural motif would be modelled with consistency, regardless of the mouse's position or orientation. However, this must be regarded as a simplification, as in fact animal behaviour is dependent on the environment, and in particular the animal's location within that environment, but for the purposes of this research we chose to disregard this fact.

Secondly, as it is inevitable when processing high dimensional raw video data that some information is lost we had to ensure that the chosen pose dynamics could best capture mouse behaviour. We therefore chose features that were not entirely correlated with each other, in order to capture different aspects of behaviour. See the plots in figure 14 for more information.

Having taken these requirements into consideration, we chose the angle of the mouse's head with respect to the axis of its body, the magnitude of the speed of the center of gravity of the mouse, and the elongation of the mouse's body as the three features best suited to comprise the kinematic profile used for modelling. Specifics on feature calculations and analysis are presented in section A.1.

3.4 Behaviour segmentation with ARHMM

To extract discrete behaviour from the mouse an ARHMM model was utilised. As has been shown it is possible to generate mouse-like behaviour from the states found with an ARHMM according to Wiltschko et al.[2]. It was further proven that this model is able to find well-separated behavioural modules in a real mouse. When applied to synthetic mouse data it is, however unable to find such modules.

This model was also chosen because it segments states as a time series which is how behaviour should look like [2]. Each such sequence will be contained inside a state of a Hidden Markov Model which then will represent a specific behaviour. These states can then be combined to represent a more complete behaviour.

The main parameters when fitting an ARHMM to time series data are the number of states K, the number of auto-regressive lags l and the stickiness parameter κ. For this project we decided not to investigate the effect of changing the lags due to the limited time frame of this project. The optimal value for K, the number of states, was chosen to be the one that generated the the highest log-likelihood of the fit to data. The optimal κ variable however was determined by using the calculated cross-likelihood matrix of the found states, described in detail in section 3.5 and shown in figure 8. To find the best performing model we used multiple values during experimentation, but this report only presents the findings for $\kappa = 1, 2500, 5000$ and 10000, lags $l = 1$ and the number of states K which maximise the log-likelihood.

When the optimal parameters were chosen we also investigated if merging states increased the likelihood even further. If so, merging was done. More details about how the merging was performed can be found in section A.4.

In this project the ARHMM model from the `ssm` package[1] was used. We used the behavenet branch which is used in the behavenet paper [5].

[1] https://github.com/lindermanlab/ssm

3.5 Evaluating of detected behaviours

To evaluate the behaviours identified by our model we considered three aspects; how well separated, i.e. distinct, the behaviours were, which is necessary in order to evaluate whether the states the model finds are meaningful; to which degree the states created by the model describe the same behaviour in a different mouse; to which extent the states are stereotypical.

For the first two a cross-likelihood matrix, CL, was used as it provided us with a quantitative way to conduct our analysis. Intuitively, an entry $CL(i, j)$ in a cross-likelihood matrix is a measure of how much better or worse state i can explain the state trajectories (figure 19) currently classified as state j (details can be found in Section A.3). This is important to evaluate as previous research has shown that mouse behaviours is organised into distinct sub-second modules [2]. Thus, for us to say that we have captured true behaviour we had to show that the states the ARHMM identifies are distinct states, that is each of them describe their own series of particular movements.

Ideally the matrix should have all negative values with a zero-diagonal, which would indicate that each state uniquely describes a unique behaviour that cannot be explained better by any other state.

The same logic was used for the generalisation aspect of our model. If the cross likelihood matrix showed distinct states when the model was applied to a mouse it had not seen before it would be a sign that it had found general mouse behavioural states to some extent. The extent in this case would be for mice under the same laboratory conditions displaying roughly the same type of behaviour.

Lastly, to show that our model found stereotypical behavioural modules we observed the video sequences displaying a particular behaviour. If the mouse during one state conducted the same series of movements this would mean that it was a stereotypical state. We argued this was an acceptable measurement, even if qualitative, as this was one method used in the sub second behaviour paper to determine if the states were stereotypical [2]. Additional qualitative measures are suggested in section 8.6.

4 Implementation

4.1 Experimental setup

To conduct the project, access to one of labs at the KM-B (Comparative Medicine Biomedicum) facility at Karolinska Institute was granted. There an $0.6 \times 0.6\ m^2$ enclosed arena was constructed where lab mice could roam free. At two locations in the arena, at the top, the reward box (RB), and the bottom, the trial initiation box (TB), there are holes which carry photo-resistors with a beam that once broken triggers the next phase in some trial setting, see figure 2. Above each hole are QLED lights indicating the phase of the trial. The photo-resistors, QLEDS and pump for the water dispenser were controlled by a Arduino Mega 2560 Microcontroller board.

To encourage mice into a goal-orientated behaviour they were food restricted and then trained to activate TB to receive a dose of sugar water at RB. However if the mice desired more sugar water it had to return to the bottom trigger and repeat the process, once a trial cooldown time has passed. This was done in accordance with the guidelines and experimental protocols approved by KI. More information can be found in the appendix in section A.6.

By having the mouse in a goal-orientated mode it would make it possible to reproduce results as one could with greater certainty say that the background variables in these cases were the same. However, this scheme was also part of the second goal which was not reached and thus left for future work, see 8.6 for more information.

In addition, visual stimuli was presented using a projector placed under the arena, projecting onto the glass floor of the arena. Multiple types of stimulus were presented; moving dots of fixed size, still dots of fixed size and fixed dots of growing or shrinking size. Like the goal orientated behaviour this was also for future work described in section 8.6.

An infrared high speed camera (FLIR BlackFly S USB3), was placed approximately 1.5 meters above the arena to capture the experiments. By setting the frame size to 448×448, exposure time to 1500 μs, disabling various camera auto-adjustments and synchronizing the camera exposures to a precise 2 ms interrupt procedure on the Arduino board, we were able to capture video in 500 FPS, giving a temporal resolution of 2 ms. The synchronization with the Arduino board also allowed us to perfectly

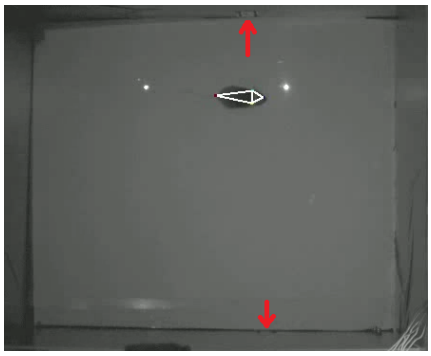


Figure 2: Experimental arena setup. Mouse is overlayed with DeepLabCut pose estimates and the red arrows indicate the TB (bottom) and RB (top) respectively. Image taken by BlackFly during mouse conditioning

synchronize the events in the arena, such as poking RB or TB, with the frames captured by the BlackFly.

4.2 Position extraction

To train DLC we labelled roughly 700 frames of four different mice in different light settings. The marker size for each body part was set to 2 pixels as we anticipated it was sufficient to mark the area of interest. Furthermore, the batch size, the augmenter and learning rate were all set to the DLC default values. The probability cut-off value (p_{cutoff}) was set to 0.99 to ensure that we only extract high confidence positions.

We evaluated the ResNet50, ResNet101 and MobileNetV2 networks and found that ResNet101 showed the lowest, albeit marginally, mean pixel error, compared to ResNet50, while MobileNet performed significantly worse [17, 18]. We decided on using ResNet50 due to its superior inference speed compared to ResNet101 which allowed for live tracking, while still maintaining a low mean pixel error.

We trained the ResNet50 architecture on a Nvidia Titan RTX GPU, with DLC for 400k iterations as the loss started to plateau at that point, see figure 3. The average pixel error on the test data was 1.97 with $p_{\text{cutoff}} = 0.99$.

Finally, it is important to note that for certain videos cropping was used to avoid including parts outside of the arena, which decreased erroneous markers.

5 Data

For our ARHMM experiments we used two videos of two different mice, none of which had been used for training DLC, mouse #2050 and mouse #2053. Visual stimuli occurs in both videos, but this is disregarded in the analysis as the mice responses were negligible. The reason for having visual stimuli present is described more in detail in section 8.6.

The data for mouse #2050 and #2053 were two videos taken of the two mice when they are roaming free. They consist of roughly 260000 frames (≈ 9.4 minutes) and 230000 frames (≈ 8.3 minutes) respectively.

To make the data comparable between the two mice sequences of 220000 frames, roughly 8 minutes of footage were extracted from each of the videos. These 220k frames were sequential frames after the first 2000 frames. The first 2000 frames were ignored due to typical initial inaccurate estimates from DLC that to us seemed to occur at the start of any video.

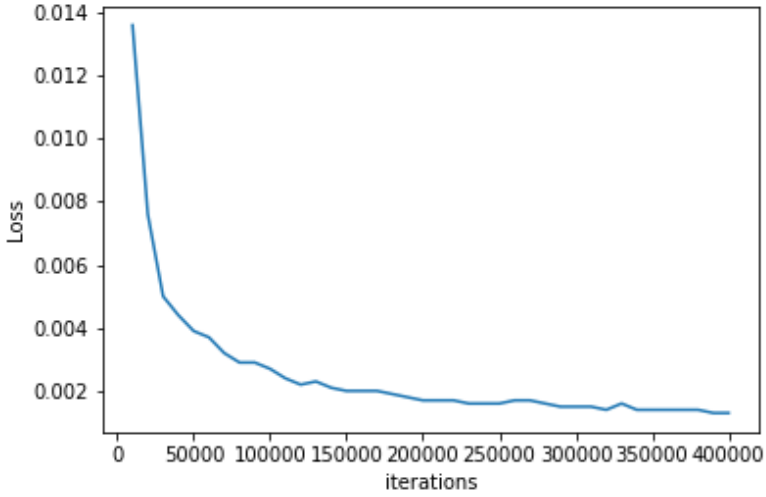
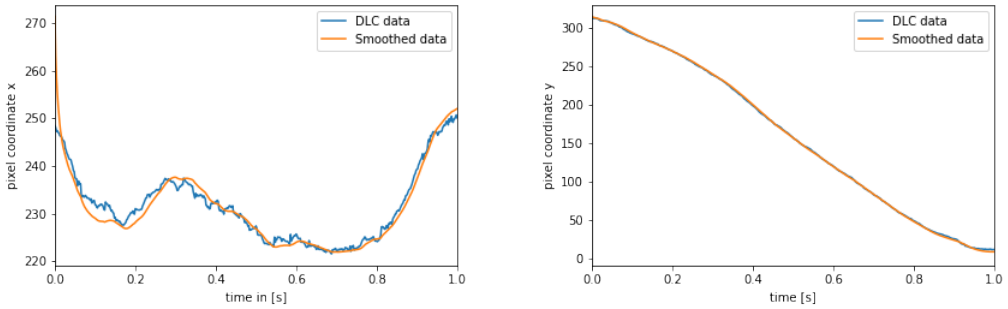


Figure 3: The calculated DLC loss per 10k iterations

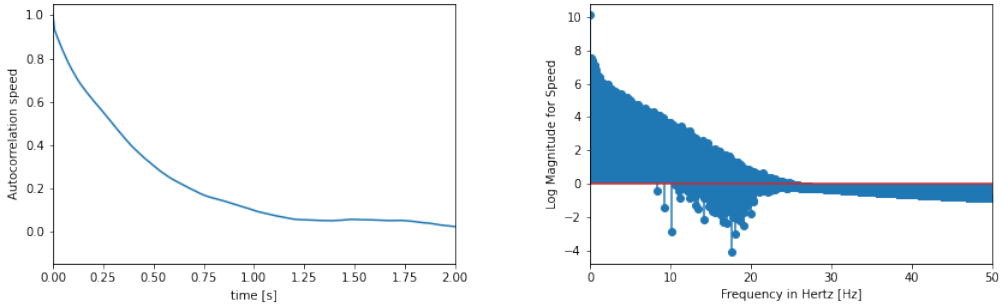
5.1 Data pre-processing for behaviour segmentation

The first pre-processing step was the application of a Kalman filter. As can be seen in figure 4, the noisy original trajectory was smoothed by the Kalman filter without any substantial lag. The time constant T was set to $T = \frac{1}{10}$ based on the observed top speed of the mouse, which we estimated to 0.4 m/s, in discussions with the researchers at KI. When applied on data that is correctly classified by DLC the mean pixel error between the coordinate given by DLC and the Kalman Filter was below 0.1 pixels.



(a) x-coordinate **(b)** y-coordinate

Figure 4: Kalman filter applied to the snout coordinates of mouse #2050 data. The blue curve shows the coordinate before smoothing and the orange curve the same coordinate after smoothing



(a) Autocorrelation function for speed from mouse data $\tau = 0.4$

(b) Logarithm of Fourier spectrum for the speed variable

Figure 5: Autocorrelation function and Fourier spectrum for the speed variable

In order to determine on what time scale behaviour exists we analysed the autocorrelation function (ACF), the Fourier spectrum and performed a change point analysis of the three variables of the kinematic profile. From the ACF we extracted the mean time, τ, of the correlating sequences which was 0.4, 0.43 and 0.18 seconds for speed, head body angle and body length respectively. The changepoint analysis gave a mean time between changes in the data of 0.26, 0.244 and 0.304 seconds between the three variables. This indicated that behaviour motifs are expected to be found on a time scale with a mean of 200-400 ms. Furthermore the Fourier spectrum analysis found that values above 5 Hz could be considered noise. The ACF and Fourier spectrum analysis for speed can be seen in figure 5. Figures for the other variables and the changepoint analysis can be found in section A.2.

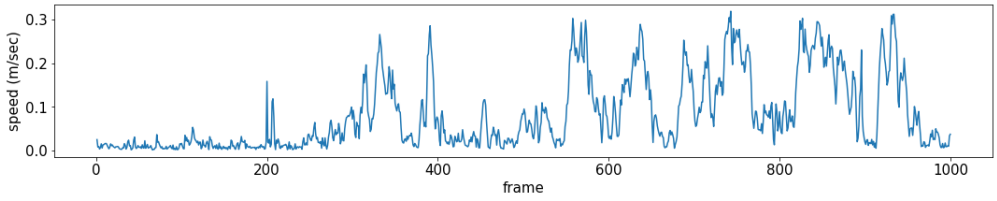
In order to verify that the found timescale is indeed the relevant one, one ought to perform the changepoint, spectrum and autocorrelation on multiple animals. However, since the timescales were similar to those found by Wiltschko et al. we refrained from doing so, but for rigour this ought to be studied in future work.

Furthermore, since the suggested time scales were similar to those found by Wiltschko et al., this implied that the three variables were reasonable choices and that the actual behaviours did not exist on the millisecond time scale [2]. This motivated that a Gaussian filter with a cut off frequency of 5 Hz could be applied on the data without information loss. Before the Gaussian filter was applied, thresholds were applied to the three variables in order to truncate outliers generated by DLC. The respective thresholds where motivated by analysis where the mouse was not seen to approach speeds higher than 0.4 m/s and to not have a head body angle greater than 90 degrees.

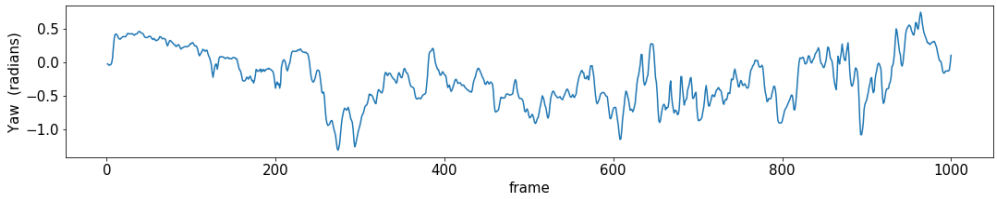
For mouse #2050 this resulted in a 1% loss of total data and for mouse #2053 2%. We argued that this was reasonable even though there was a possibility that it could create a gap in the data. This could impact the ARHMM model's performance due to the prerequisite that the data is sequential. However not removing the erroneous data could add additional faulty states which was also undesirable. Thus we decided to investigate whether our model created additional states for the transitions over these gaps by analysing the videos.

Furthermore the data was down-sampled to 50 FPS. This was motivated by the time scale found by the ACF, change point and Fourier analysis. This resolved the problem of noise from DLC being on the same order of magnitude as the movement of the mouse in 500 FPS.

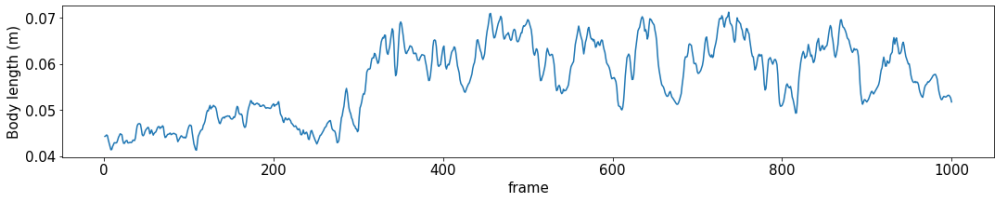
The data that was analysed with the ARHMM was divided into a training and validation (70/30) dataset in order to avoid overfitting.



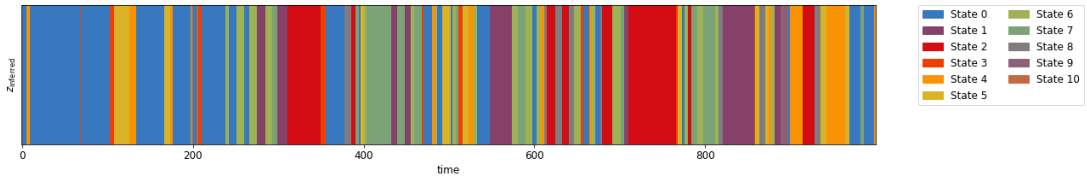
(a) The speed of the mouse over time



(b) The yaw angle of the mouse over time



(c) The body length of the mouse over time



(d) ARHMM applied to mouse #2050 after being trained on mouse #2050

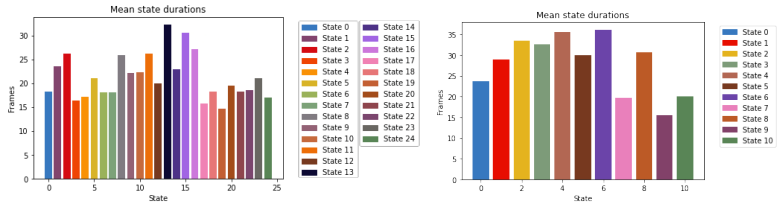
Figure 6: The data from mouse #2050 inputted into the ARHMM for frames 9000 to 10000 with values for speed, yaw and body length as well as the state transitions provided by the ARHMM

6 Results and discussion

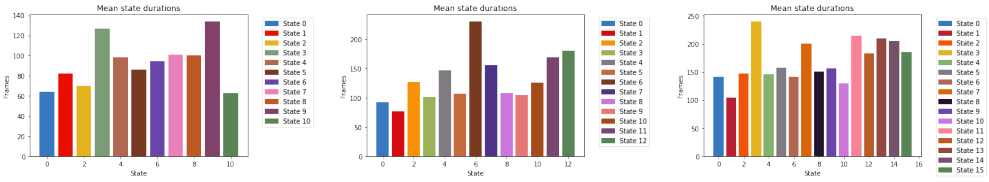
6.1 Model selection

After applying the ARHMM on the pre-processed data from mouse #2050 with $\kappa = 2500$, where a small segment of the data can be found in figure 6, we got the state sequence for frames 9000 to 10000 that can be seen in figure 6d. Here the number of states that gave the highest likelihood was 11. For the other values for κ we got similar looking likelihood graphs but the number of optimal states varied. For $\kappa = 1$ it was 25, for $\kappa = 5000$ it was 13 and for $\kappa = 10000$ it was 16.

Figure 7 illustrates that fitting the data with a low value on κ did not produce states that lasted as long as the predicted time series analysis. For $\kappa = 1$ we achieved a mean state duration of 60 ms while our prediction was between 200-400 ms. Figures 8a and 8b also illustrate the poor state distinctiveness produced by choosing $\kappa = 1$. By increasing κ we found three models that suited the time scale better. By considering the evolution of their log-likelihood over number of states for mouse #2050 (figure 9) we found that they all had a decay in likelihood in the same region as where the $\kappa = 1$ model planes out. This indicated that the model with $\kappa = 1$ overfitted, and suggests that κ acts as a regulariser. The mean state time for the fit with these 3 larger κ values are presented in figure 7 and we see a mean of 200 ms for $\kappa = 2500$, 270 ms for $\kappa = 5000$ and 400 ms for $\kappa = 10000$, all three within our predicted timescale. One drawback with applying $\kappa > 1$ was that the likelihood of the fit decreased somewhat with large κ. This was interpreted as an effect of applying regularisation and does not necessarily mean that the fit was worse at explaining the actual behaviours.

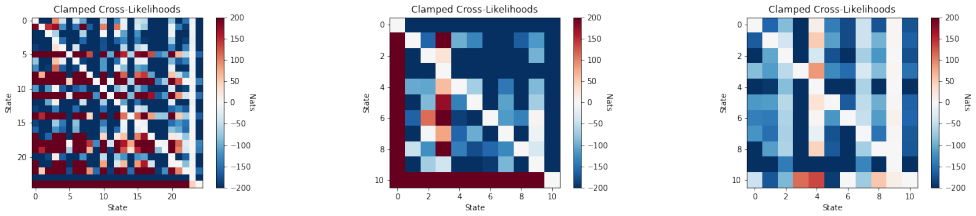


(a) $\kappa = 1$, $K = 25$ **(b)** $\kappa = 1$, $K = 11$

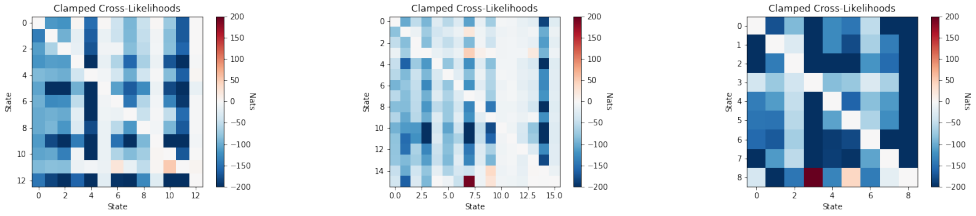


(c) $\kappa = 2500$, $K = 11$ **(d)** $\kappa = 5000$, $K = 13$ **(e)** $\kappa = 10000$, $K = 16$

Figure 7: Mean state durations for mouse #2050 fitted with various κ and K observed states. The y-axis has unit frames where one frame is 2 ms.

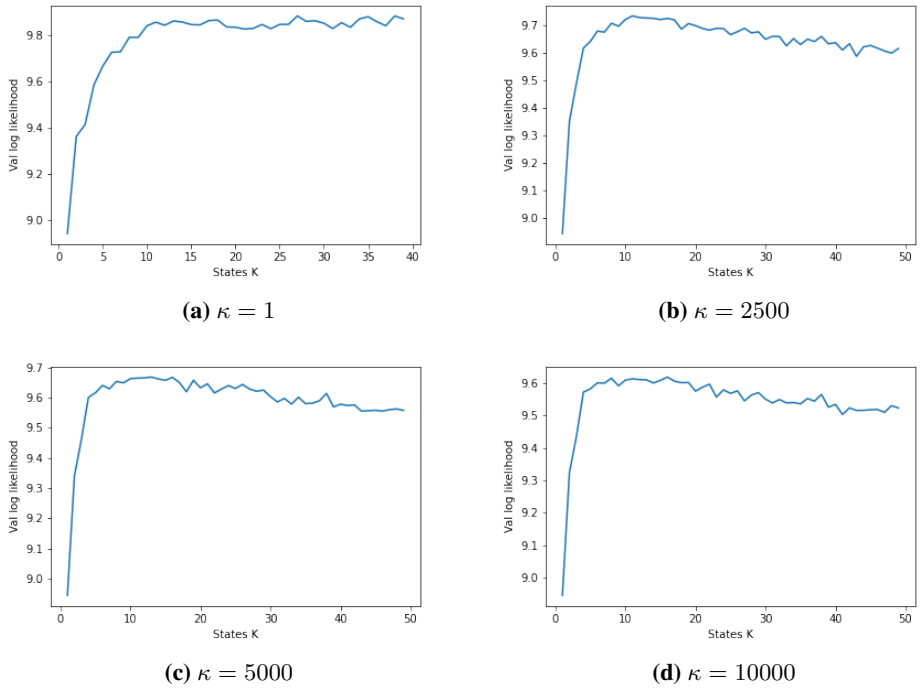


(a) $\kappa = 1$, $K = 25$, mean$=-34$ **(b)** $\kappa = 1$, $K = 11$, mean$= 0$ **(c)** $\kappa = 2500$, $K = 11$, mean$=-119$



(d) $\kappa = 5000$, $K = 13$, mean$= -92$ **(e)** $\kappa = 10000$, $K = 16$, mean$= -50$ **(f)** $\kappa = 2500$, $K = 9$ with two merged states and mean$= -146$

Figure 8: Cross-Likelihood matrices for mouse #2050 with various κ and K. The values are clamped to $[-200, 200]$ for clearer visualization



(a) $\kappa = 1$ **(b)** $\kappa = 2500$

(c) $\kappa = 5000$ **(d)** $\kappa = 10000$

Figure 9: Log likelihood of fitting the model to mouse #2050 video data with varying κ

6.2 State distinctiveness

Using the cross-likelihood matrix, described in section 3.5, many states were found to be non-distinct from visual inspection of figure 8 where κ was set to 1. However, decreasing K and increasing κ yielded more distinct states. With the optimal number of states suggested by figure 9 most entries were negative (blue), with a few exceptions. Notably, state 4 (column 5) is seen to be relatively non-distinct, as compared to the other states. When inspecting the video sequences where state 4 occurs, we found that it was indeed hard to interpret this as any stereotypical behaviour, whereas other states were showing sequences of stereotypical behaviour, such as sitting still (state 0), walking (state 2), left yaw (state 6) and right yaw (state 8). The videos can be found on our GitHub[2]. We interpreted this as evidence that our time scale analysis has guided us to a time scale where behaviour is expressed, and that the correct choice of K is important to extract distinct behaviours.

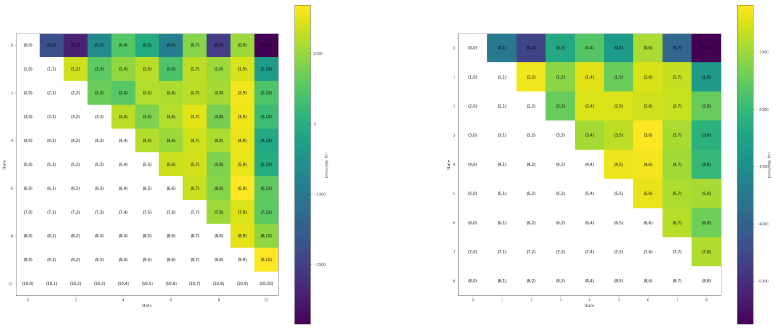
Furthermore, we note that we yield the lowest average cross-likelihood with $\kappa = 2500$ (figure 8c). Increasing κ further increased the average cross-likelihood, which indicated that the states were then less distinct. This can also be seen by the lighter shade of blue apparent in figures 8d and 8e. We therefore decided to choose the model with $\kappa = 2500$ as our main model due to it having a higher likelihood of the fit.

By visual inspection of state videos for our model, we found that the state 10 captured sequences containing outliers. These were created by removed frames, in which DLC failed to register a label or labelled it inaccurately, which in turn caused large leaps in the variables. On one hand this is positive, since it means that these outliers do not distort the other states, but on the other hand we also observe that some high speed sequences are classified into this state. If outliers are masked, or a more robust tracking model is used, these high speed movements may constitute a state on their own and not end up in an outlier state, as currently observed.

6.3 State merging

Since state 4 appeared non-distinct, we chose to merge it with another state. By considering figure 8c we note that states 1, 3 and 10 would explain these trajectories better than state 4. Furthermore, the

[2]https://github.com/mpierrau/DD2430_Project_Course_in_Data_Science



(a) How much merging two states would affect the log-likelihood before merging on mouse #2050 with $\kappa = 2500$

(b) How much merging two states would affect the log-likelihood after merging on mouse #2050 with $\kappa = 2500$

Figure 10: How much merging two states would affect the log-likelihood for mouse #2050 with $\kappa = 2500$

merge was supported by the merge matrix in figure 10a, where each cell represents how much the log-likelihood would be affected by merging the two states in each cell. This information combined with the information from the cross-likelihood plot led us to merge state 4 with state 3, and state 9 with state 6 by the same argument. After merging we arrived at a model with 9 resulting states, with a cross-likelihood matrix shown in figure 8f. This model was then used when evaluating generalisability between mice. Figure 10b shows that a subsequent merge would only affect the log-likelihood very marginally.

6.4 State stereotypicality and generalisability

To further evaluate that the trajectories through pose space were meaningful we also created videos for the different states sequences which indicated that most states for the presented models with $\kappa > 1$ were meaningful.

Figures 6d and 11 illustrates inferred state transitions on a 1000 frame subsample of the data used to train the model, and for mouse #2053 data respectively.

From these figures we find that there are some overlapping states between models, but many more mismatches between states. However, since the states have different ID's (as seen by the varying colors in the figures) in the different models, it is hard to compare them, which is why we use the cross-likelihood matrices for comparison.

The state trajectories (figure 19) gave an indication that each state seemed to capture different behaviours, but it is hard to quantitatively determine whether these behaviours are meaningful and distinct from this. In order to quantify their distinctiveness we consider the cross-likelihood matrices (figure 12). Comparing this figure with figure 8f we can see that there is a difference in cross-likelihood when we just use the mouse #2050 data and when we generalise with mouse #2053 data. The difference is however not much different when we compare the best model with the models with different κ in figure 8. It even achieves a lower cross-likelihood than our model, before the states were merged, which should be seen as very good generalisability capability.

This all further suggested that the chosen model with $K = 9$ and $\kappa = 2500$ is a representative model for mouse behaviour.

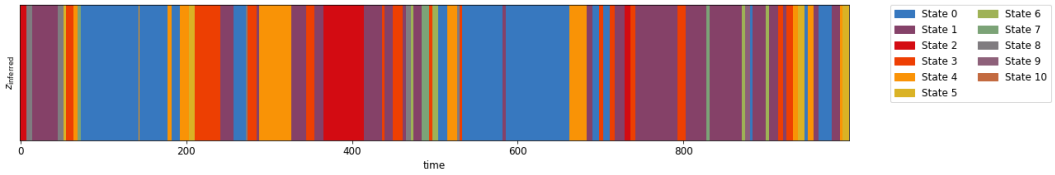


Figure 11: ARHMM applied to mouse #2053 after being trained on mouse #2050

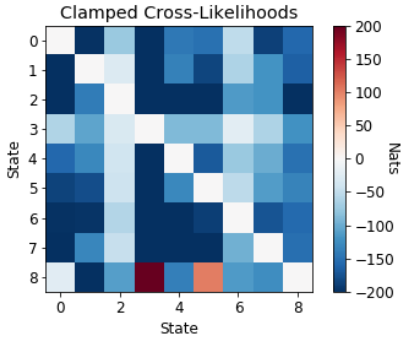


Figure 12: The clamped cross likelihood matrix for the merged ARHMM applied to the data of mouse #2053 with a mean$= -138$

7 Conclusions

The goal of this project was to:
"*Create a pipeline which detects and tracks the mouse in real time and classifies the mouse's behaviour into distinct motifs offline.*"

We were in this project able to detect points on the mouse in real time and track it when it moves through the arena. Our pipeline then analyses the data offline and segments it into distinct behaviours.

Applying an ARHMM-model with 9 states and $\kappa = 2500$ yielded us the most distinct states according to our cross likelihood evaluation. Applying that model to another mouse further confirmed that the model was able to generalise and therefore was able to capture general behavioural patterns for mice.

Assigning a meaningful value to κ takes us back to the question in the introduction regarding what a behaviour actually is. Is behaviour as suggested a "[...] symphony of muscle contractions [...]" and what are we then actually looking for? If we are looking for the muscle contractions themselves then the model should be fitted with a low κ. Are we instead interested in the actual symphony of them, i.e behaviours, then, as shown in this project, applying a higher value of κ is justifiable.

8 Future work

8.1 Improving the model

Expand dataset. We trained our final model using only one video. As there does however exist plenty of video data, there are significant improvements to be made using this data. This ought to result in a more robust and generalisable model.

All of the models used in this research were each trained using a single observation sequence consisting of <10 minutes of video footage. Such a model is invariably dependent on the specifics of the mouse behaviour presented in each video. To train a fully robust model, the ARHMM could be trained over multiple observation sequences, with the Expectation Maximization algorithm adopted to maximise over several independent time series. If we denote the K independent observation sequences as:

$$\mathbf{O} = [\mathbf{O^1}, \mathbf{O^2}, \ldots, \mathbf{O^K}]$$

where $\mathbf{O^k} = [O_1^k, \ldots, O_{n_k}^k]$ is the k:th observation sequence. Knowing that each sequence is independent, we now have to iteratively adjust our model parameters λ to maximise:

$$\mathbf{P}(\mathbf{O}|\lambda) = \prod_{k=1}^{K} \mathbf{P}(\mathbf{O^k}|\lambda)$$

Masking outliers. In the current dataset there exist outliers. This appear specifically in corners and when the mouse is poking its snout into the TB or RB. In this project we have smoothed data and thresholded out outliers, but this results in sequences which may display large leaps in the variable values, which affects the ARHMM negatively. The used package (`ssm`) allows for masking of unobserved values, which we recommend be applied to outliers in the future.

Model selection. To better motivate model selection, regularizers like AIC or BIC can be applied. It is important to understand how many parameters a model has in order to properly apply these techniques.

Computing speed. For calculating the speed of the mouse we used its center of gravity. This however means that stretching of the body implies movement of the mouse. Perhaps a more accurate descriptor of speed, which is uncorrelated to body length, would be the distance the tail label has travelled as that does usually not move when the mouse is just stretching or rearing.

Extract variables with a VAE. One of the reasons to why we in this project found less states than Wiltschko et al. did could be due to them having a 30 dimensional representation of their data extracted by a VAE while ours just had three dimensions [2]. It could hence increase the behavioural resolution to extract the variables with a VAE instead of manually.

Lags parameter search On the initial data we worked on the lag parameter for the ARHMM did not affect the log-likelihood of the fit. When applying it on data generated in a later stage of the development process a significant increase of the log-likelihood was however noted. When combining it with the κ parameter the model did however sometimes fail to segment behaviour and created one state over the whole state sequence. If this breakdown is possible to avoid a well chosen l could make the model more accurate.

Build complex behaviour from syllables. In the paper by Wiltschko et al. the authors state that behaviour is made up of sub-second behavioural syllables [2]. In this project we have not considered how combinations of state sequences may be combined, which could be explored in future work to classify more complex behavioural routines.

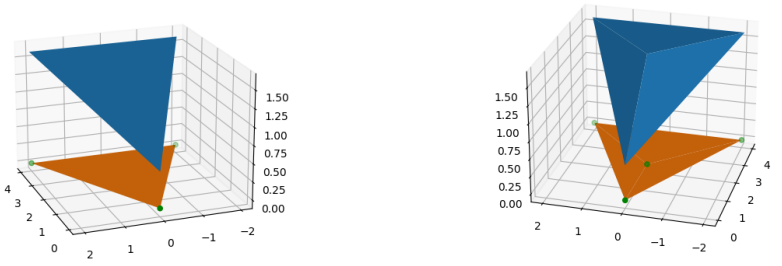
8.2 Evaluation state consistency

One important aspect that we do not cover in this report is how to measure the consistency of states between different data, i.e. if a state constitutes the same behaviour in one mice as in another. Our current evaluation of this result is visual analysis of state-videos (available on GitHub[3]). In these videos we find found that all states seem to constitute similar behaviours in subsequences of data for mouse #2050 as in #2053. However, these inspections are merely visual and may suffer from confirmation bias. It is therefore recommended that future work be focused on quantitative measures of this consistency. Suggestions include comparing mean state duration histograms and template matching, as briefly described in 8.6.

8.3 3D space

An important assumption that we made in our research is that mouse behaviour could be modelled based on on pose data extracted in only two dimensions. We had at first intended to extract 3D pose information, at least for the head of the mouse. We did this by considering the recorded 2D coordinates at a given time as a projection of the full 3D coordinates onto the xy plane. Then, given that the head of a mouse is approximately rigid, the distance between coordinates for a given mouse are fixed - in 3 dimensions at least. We recorded these distances (ear to ear, midpoint of ears to snout) from video footage and took a smooth maximum of each as a baseline. This would correspond to when the coordinates were essentially "flat" - parallel to the xy plane. Then the z coordinates at a given time could be extracted using Pythagoras' Theorem, treating the baseline as the hypotenuse, and the projection and z as the other sides. (figure 13a)

[3] https://github.com/mpierrau/DD2430_Project_Course_in_Data_Science



(a) 3D mouse coordinates projected onto XY plane.

(b) Adding a 4th non-coplanar point breaks the symmetry when calculating the z values of the coordinates.

Figure 13: Extracting 3D coordinates from recorded 2D projection. See 8.3

However a drawback to this method is that due to the symmetry of the equations, it is unknown whether a given z coordinate is positive or negative, with respect to the other, or indeed to the baseline. This symmetry resulted in the fact that when calculating the intrinsic angles of rotation of the head in three dimensions, we would be correct only in 25% of cases. These angles of rotation are of special interest to the research team at KI, as they can be directly linked to distinct neural activities, and so offer important clues to neural decoding of behaviour. To overcome this DLC could be used to track another non-coplanar point on the head. This would break the symmetry in calculating z coordinates and thus allow the angles of rotation to be calculated correctly (figure 13b). A convenient way to do this would be to attach some protruding object to the forehead of the mouse.

Another way to extract a 3D kinematic would be to use a depth camera. We chose not to use one in this project due to initial technical limitations, but for further work we recommend using a depth camera (we were considering an Intel RealSense D435i).

8.4 DLC learning

For the training we used frames where the arena was illuminated of various intensities, which resulted in varying image contrast qualities, which effectively works as data augmentation and improved the accuracy, especially in critical samples, such as when the mouse was located in dark corners. We did not study these effects quantitatively, which might be worth exploring to improve DLC's performance further. Furthermore, the trained network may be improved by re-labeling outlier frames using available DLC functions.

8.5 Network architecture study

It might be of interest to conduct a thorough study of the performance of different convolutional network architectures for the purposes of pose estimation. Our choice of network was based on computational efficiency for the online implementation, but perhaps an accurate deeper network with better predictions for the labels would be preferred in some offline scenarios.

8.6 Evaluate trained mice behaviour

One of initial goals of this project was to investigate how behaviour in a trained mouse changes when a visual stimuli is presented in its field of view.

When doing so it is important not to draw any conclusions regarding the number of observed behaviours in a video simply from the number of chosen states by the model. As of now, the model is not particularly robust, and K's of similar value show only marginal difference in log-likelihood.

In order to find a more quantitative measure of how generalizable/stereotypical the captured behaviours are, one can apply template matching using minimal Euclidean distances between trajectories, as described Wiltschko et al. [2].

We found that quantitative analysis of the segmented behavioural states is highly complex, and this goal was outside the given time span for this project.

References

[1] Niko Tinbergen. *Animal Behaviour*. 1st ed. new York: Time Inc., 1965.

[2] Alexander B Wiltschko et al. "Mapping sub-second structure in mouse behavior". In: *Neuron* 88.6 (2015), pp. 1121–1135.

[3] Diederik P Kingma and Max Welling. "An introduction to variational autoencoders". In: *arXiv preprint arXiv:1906.02691* (2019).

[4] Kevin Luxem et al. "Identifying Behavioral Structure from Deep Variational Embeddings of Animal Motion". In: *bioRxiv* (2020).

[5] Eleanor Batty et al. "BehaveNet: nonlinear embedding and Bayesian neural decoding of behavioral videos". In: *Advances in Neural Information Processing Systems*. 2019, pp. 15706–15717.

[6] Weizhe Hong et al. "Automated measurement of mouse social behaviors using depth sensing, video tracking, and machine learning". In: *Proceedings of the National Academy of Sciences* 112.38 (2015), E5351–E5360.

[7] Mathis A. et al. "DeepLabCut: markerless pose estimation of user-defined body parts with deep learning". In: *Nature neuroscience* 21(9) (2018), pp. 1281–1289.

[8] Ivars Namatevs. "Deep Convolutional Neural Networks: Structure, Feature Extraction and Training". In: *Information Technology and Management Science* 20 (Dec. 2017). DOI: 10.1515/itms-2017-0007.

[9] J. Deng et al. "ImageNet: A Large-Scale Hierarchical Image Database". In: *CVPR09*. 2009.

[10] Ivars Namatevs. "Deep Convolutional Neural Networks: Structure, Feature Extraction and Training". In: *Information Technology and Management Science* 20 (Dec. 2017). DOI: 10.1515/itms-2017-0007.

[11] Leonid Pishchulin et al. "DeepCut: Joint Subset Partition and Labeling for Multi Person Pose Estimation". In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. June 2016.

[12] Eldar Insafutdinov et al. "Deepercut: A deeper, stronger, and faster multi-person pose estimation model". In: *European Conference on Computer Vision*. Springer. 2016, pp. 34–50.

[13] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[14] Sepp Hochreiter. "The Vanishing Gradient Problem During Learning Recurrent Neural Nets and Problem Solutions". In: *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems* 6 (Apr. 1998), pp. 107–116. DOI: 10.1142/S0218488598000094.

[15] Gary Kane et al. "Real-time, low-latency closed-loop feedback using markerless posture tracking". In: *BioRxiv* (2020).

[16] Sebastian Thrun, Wolfram Burgard, and Dieter Fox. *Probabilistic Robotics (Intelligent Robotics and Autonomous Agents)*. The MIT Press, 2005. ISBN: 0262201623.

[17] Kaiming He et al. *Deep Residual Learning for Image Recognition*. 2015. arXiv: 1512.03385 [cs.CV].

[18] Mark Sandler et al. *MobileNetV2: Inverted Residuals and Linear Bottlenecks*. 2019. arXiv: 1801.04381 [cs.CV].

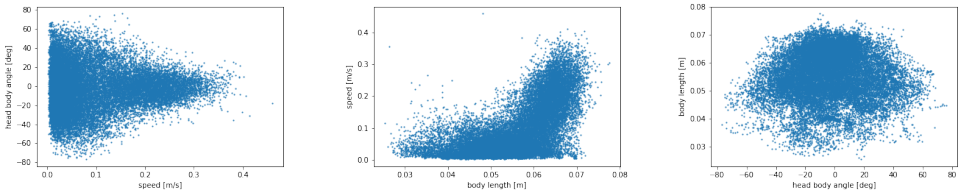
A Supplementary Material

A.1 Extraction of variables

The speed variable was extracted by taking the center of gravity of the mouse in one frame and taking the difference between that frame and four frames back divided by the time between those frames, i.e 8 ms. The yaw is computed as the angle between the vector spanned by the ears and the vector from the tail to the midpoint of the ears. The body length is computed as the length between the midpoint of the ears and the tail.

A.2 Time scale analysis

Although the correlation between body length and speed is 0.6 there is clearly almost no correlation between them for low speeds. Therefore we decided that the variables actually do contain different information and thus can provide us with important information about the mouse's behaviour.

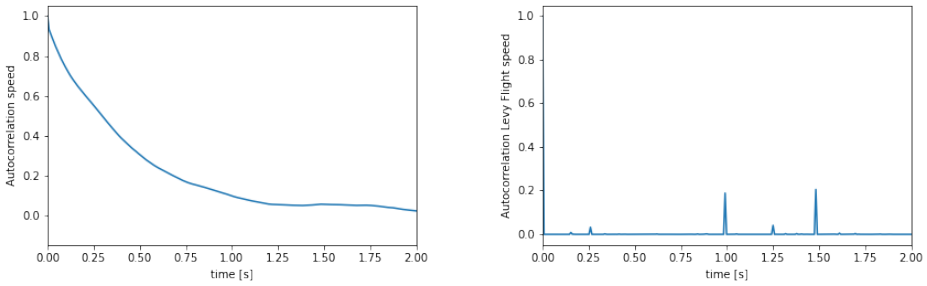


(a) Correlation between speed and head angle in free roaming mouse

(b) Correlation between speed and body length in free roaming mouse

(c) Correlation between body length and head angle in free roaming mouse

Figure 14: Correlations between our variables

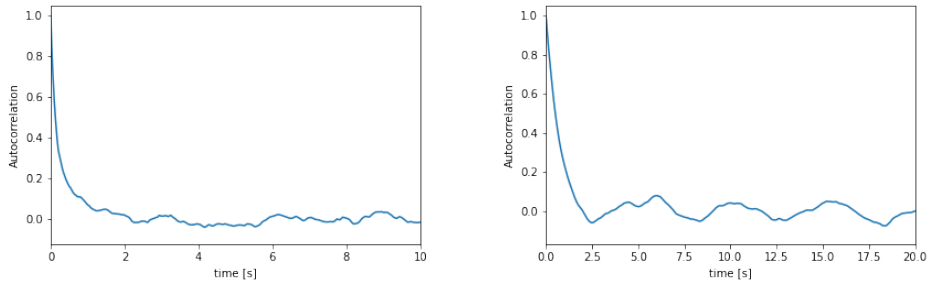


(a) Autocorrelation function for speed from mouse data $\tau = 0.4$

(b) Autocorrelation function for speed from Levy Flight simulation

Figure 15: Autocorrelation function for speed for real data and Levy Flight simulation

By comparing the ACF of the speed that comes from real mouse data to the ACF of speed in the levy flight generation and random walk in 15 we can clearly see that there are non-random behaviour in our data. In the ACF for the mouse data speed decreases up until around one second while in the two randomly generated ACF's the speed decreases instantly to zero. This is similar results to the one provided by Wiltschko et al. where they concluded that there are behaviours on the sub second time scale [2].

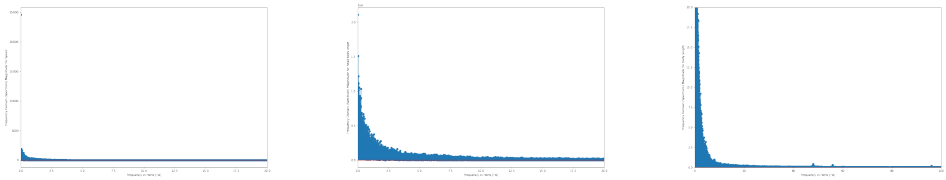


(a) Autocorrelation function for head body angle $\tau = 0.43$

(b) Autocorrelation function for body length $\tau = 0.18$

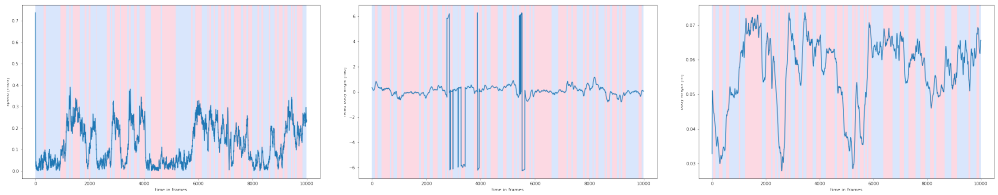
Figure 16: Autocorrelations functions for head body angle and body length

Furthermore our autocorrelation functions for the head body angle and body length are provided in figure 16 where it is clear that there exists correlation in time due to the functions decreasing towards zero after a few seconds. This in contrast to the behaviour of a randomly initiated variable that instantly decreases to zero.



(a) Speed **(b)** Head body angle **(c)** Body length

Figure 17: Fourier spectrum over the chosen variables



(a) Speed, mean changetime 264 ms

(b) Head body angle, mean changetime 244 ms

(c) Body length. mean changetime 306 ms

Figure 18: Changepoint analysis of the three first variables for the first 10000 frames of the mouse #2050 video The change between blue and red indicates a change of state.

To further investigate on what time scale behaviour can be found we provide a Change Point analysis as seen in figure 18. From this we can draw the conclusion that the mean time between changes in our three variables are on the time scale of 200-300 ms. This is similar to the results achieved in [2].

Furthermore we performed Fourier spectrum analysis to the data in order to see if there was noise that could be filtered out. As can be seen in figure 17 where we can see that almost all information if located in the frequency range below 5 Hz. This is in the same time range as the results from the ACF and changepoint analysis so we decided to filter the data through a Gaussian low pass filter with cut off frequency of 5 Hz.

That we get similar time scales as the ones Wiltschko et al. implies that the three variables are reasonable choices and that the actual behaviours do not exist on the millisecond time frame. [2]

$$CL_{i,j} = \frac{1}{N_j} \sum_{k=1}^{N_j} \log \frac{p(y_k^{(j)}|\theta^{(i)})}{p(y_k^{(j)}|\theta^{(j)})} \tag{2}$$

A.3 Computing cross-likelihood

For evaluating our results the cross-likelihood was computed. This was inspired by the appendix of Wiltschko et al. where it is described as a method to evaluate Hidden Markov Models [2]. For two states $i, j \in 1, 2, ..., K$ with assigned data segments $\{y_k^{(i)}\}_{k=1}^{N_i} = \{y_{t_a:t_b} : x_{t_a:t_b} = i, x_{t_a-1} \neq i \neq x_{t_b+1}\}$ and $\{y_k^{(j)}\}_{k=1}^{N_j} = \{y_{t_a:t_b} : x_{t_a:t_b} = j, x_{t_a-1} \neq j \neq x_{t_b+1}\}$ and autoregressive parameters $\theta^{(i)}$ and $\theta^{(j)}$ the corresponding i, j entry of the cross-likelihood matrix was calculated as described in equation 2.

A.4 Merging states

When merging states in the ARHMM model we take two states at the time and calculate the expectation step in the Expectation Maximisation algorithm. The returned matrix contains probabilities for states K at times T so it has dimension $K \times T$. We then sum the entries belonging to the two states and reduce the dimension to $(K-1) \times T$ and performs the maximisation step on this new matrix. We can then use the likelihoods from the maximisation steps to obtain a matrix as in figure 10. It is then possible to compare how much the likelihood would increase or decrease by merging the two states represented in each cell in the matrix.

A.5 On working with KI

As KI is an ambitious and driven research organisation it is fairly easy to get caught up in their ambitions. To make sure that you are able to deliver a report at the end of the semester we recommend that you fairly early set up a project scope so that you and KI know what you will be doing. By doing this it is easier to avoid starting to focus on things that do not concern your project.

A.6 Ethical aspects

All mice used in the experiments were trained and handled by educated KI personnel according to ethical permit number 9179-2017.

A.7 State trajectories

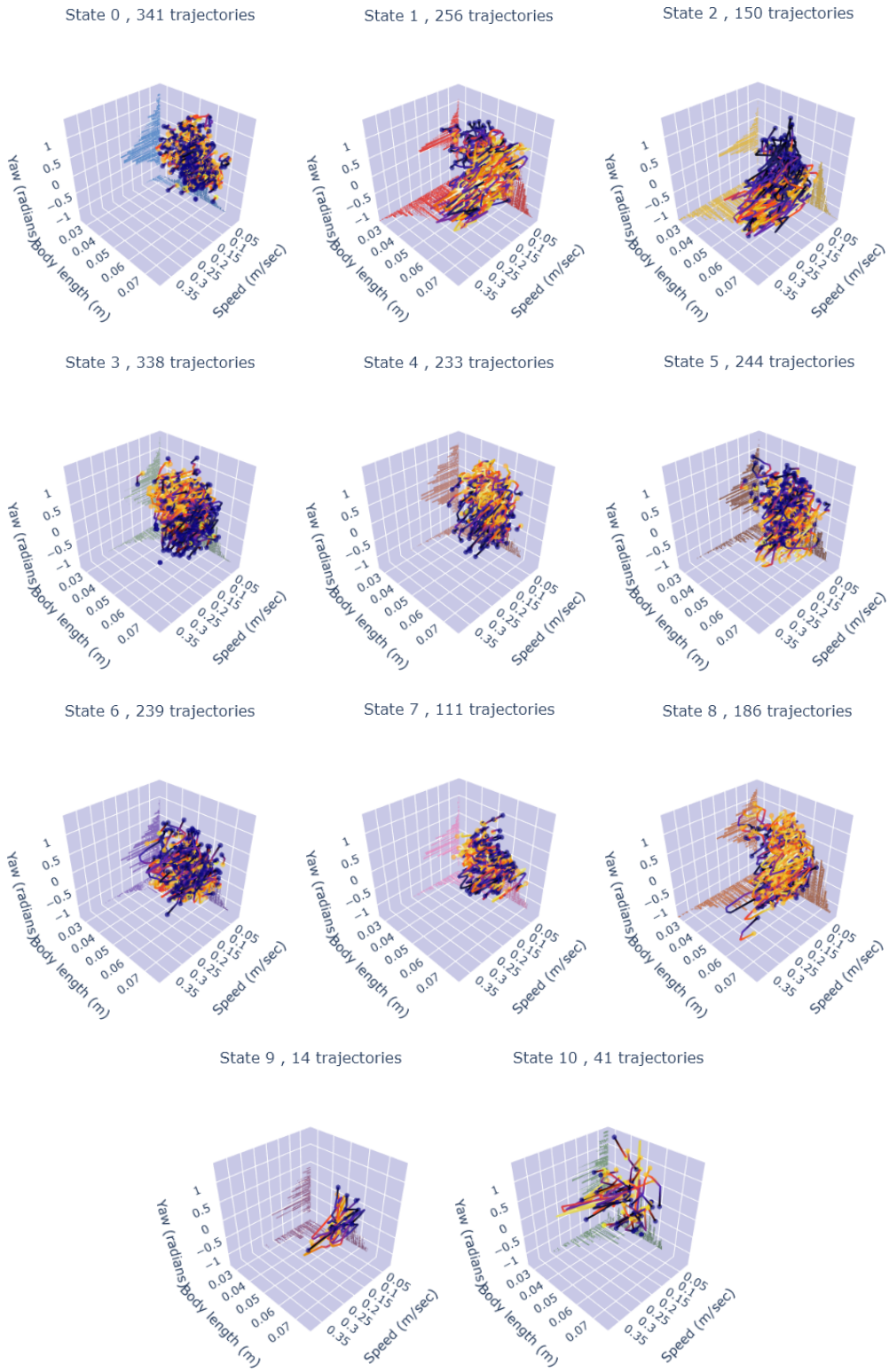


Figure 19: State trajectories for model trained on mouse #2050 data with $\kappa = 2500$ and $K = 11$. Trajectories are plotted against the three variables speed (x-axis), body length (y-axis) and yaw (z-axis). The colour of the trajectories indicate their direction, from blue to yellow