Real-time Detailed Video Analysis of Fruit Flies

CS229 Fall 2018 Final Project

Steven Herbst
sherbst@stanford.edu

ABSTRACT

In this paper, I describe a real-time image processing pipeline for fruit fly videos that can detect the position, orientation, sex, and (for male flies) wing angles. The machine learning algorithms used include a decision tree, linear and logistic regressions, and principal component analysis. The Histogram of Oriented Gradients [2] descriptor is used as well to generate features. Ultimately, I achieved a processing throughput of 84 frames per second on 1530x1530 grayscale frames without GPU acceleration, and demonstrated high accuracy across several metrics.

1 INTRODUCTION

Fruit flies are often used in neurobiology experiments because they can be genetically modified using well-established tools and because it is feasible to do *in vivo* whole-brain imaging of them (in addition to other reasons). In these experiments, it is often desirable to measure the behavior of each fly by recording its position and orientation over time, along with its leg and wing positions.

Automated tools for doing this kind of video analysis exist, but are usually too slow to run in real time. This is problematic for two reasons: First, real-time operation is a prerequisite for running closed-loop behavior experiments. Second, in an experiment where flies are recorded continuously over a long period (e.g., a circadian rhythm study), video processing will become the bottleneck for experimental throughput unless it runs in real-time (or faster).

To address these issues, I sought to develop a tool for the real-time video analysis of fruit flies. In this project, I chose to focus on a specific experiment being developed in Prof. Tom Clandinin's lab at Stanford to study the courtship interaction between one male and one female fly. As a result, the input to my algorithm is a grayscale video of the two flies (Figure 1), and the outputs are the position, orientation, and sex of each fly. In addition, the wing angles of the male fly are reported, which are needed to measure the rapid wing movement it uses to produce characteristic courtship songs.

In this paper, I'll start off by describing some existing tools for fly video analysis (Section 2), and will then move on to describe the dataset I worked with (Section 3). Next, I'll describe the algorithm I developed, which consists of four distinct processing steps using machine learning (Section 4). Finally I'll wrap up the experimental results (Section 6) and conclusion (Section 7). The source code and dataset for this project are available on GitHub at https://github.com/sgherbst/cs229-project.git.

2 RELATED WORK

The tool considered a "gold standard" of sorts for automated fruit fly video analysis is called FlyTracker [4]. Developed in the Computational Vision Lab at Caltech, it is a MATLAB-based workflow for recording the the body position, leg position, body angle, and wing angle of many flies interacting with each other. The image processing technique is based on operations such as thresholding and morphological image transforms. In informal experiments, I found it to be about 10x slower than real time.

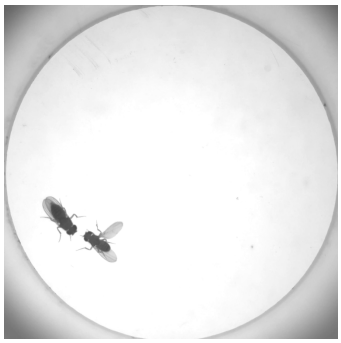

Figure 1: Grayscale video of the interaction between a male and female fly that served as the input for this project.

In another project [6], researchers sought to apply an unsupervised learning approach to study interactions between pairs of flies. The inputs to their unsupervised algorithm were two "egocentrically-aligned" videos; each of these videos was centered and cropped to one of the flies, and it was rotated so that the fly faced in a nominal direction. Their approach to segmenting the two flies includes (1) using a Gaussian mixture model applied to a histogram of pixel intensities to set various thresholds, and (2) using the pixel distance from the fly body as a criterion for assigning fly appendages to one fly or the other. In rare cases where the fly bodies themselves could not be segmented, a watershed algorithm was used to separate the two flies, which sometimes resulted in fly wings or legs being masked out. The processing speed of this approach was not reported.

Finally, in the past year, two different approaches to fly video analysis using deep neural networks were published: DeepLabCut [9] and LEAP [12]. In DeepLabCut, researchers leveraged transfer learning to reduce the number of labeled video frames required; the starting point was a neural network for feature extraction pre-trained on the massive ImageNet dataset [3]. The speed of DeepLabCut was 30 Hz for 682x540 images using GPU acceleration. For LEAP, the developers created a custom neural network using a simpler architecture than DeepLabCut, targeting higher throughput. The end result was a 185 Hz processing rate (albeit on smaller images) and a one-hour training time (both using GPU acceleration).

In this project, I sought to combine various aspects of these previous studies. On one hand, I wanted to develop an algorithm that could run in real-time on large frames (1530x1530) without GPU acceleration, and I wanted training to be fast to allow for more experimentation. Hence, I needed to further simplify the machine learning models as compared to DeepLabCut and LEAP. But I still

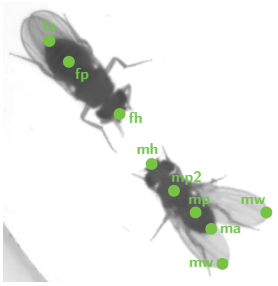

Figure 2: The dataset consisted of individual video frames labeled with various key points on both flies.

wanted to apply supervised learning to take advantage of labeled data, departing from the hand-crafted image processing rules of FlyTracker.

3 DATASET

The source data for this project was a 15 minute grayscale video of the interaction between one male fly and one female fly. The video was furnished by Dr. Ryan York of Prof. Clandinin's lab as an example of the kind of footage that will be produced by the experimental rig they are developing. I did a bit of initial preprocessing to crop the video to a 1530x1530 frame that exactly contained the circular well in which the flies were placed.

Working from the cropped video, I then hand-annotated 326 frames using LabelMe [14] to indicate the positions of heads, abdomens, and a point within the fly body. For male flies, I also annotated wing angles via additional points (Figure 2). To select frames for labeling, I first used a script to randomly choose about one hundred frames from the video. Since flies tend to move around in bursts, this didn't produce a particularly varied dataset, so I hand-selected the remaining two hundred or so frames to focus on "interesting" cases, such as points in time where the flies were crawling along the wall, were in contact, or were moving their wings extensively[1].

Additional preprocessing and data augmentation was used throughout the image processing pipeline, and these steps will be covered in the next section.

4 METHODS

As shown in Figure 3, the image processing pipeline has four distinct stages that use machine learning. First, contours are extracted from the frame and filtered to those containing an individual fly. When there are two fly contours (the most common case), the second pipeline stage identifies which contour is the male fly and which is the female fly. Next, the 0–360° orientation of each fly is determined. Finally, for the male fly, the angles of the right and left wing are determined.

The pipeline described below was implemented in Python using the packages `scikit_learn` [11], `opencv-python` [1], `numpy` [10], `matplotlib` [5], `imutils`, `joblib`, and `tqdm`.

[1]LEAP [12] has an interesting approach to solving this problem by first using PCA to represent frames on a reduced basis and then using clustering to identify frames with distinct types of fly poses. But it does assume that the frames have already been egocentrically-aligned.

4.1 Pipeline Stage: FlyCount

As shown in Figure 4, the first pipeline stage identifies the contours of individual flies. This is done by first thresholding the image, eliminating the background and fly appendages, and leaving only the fly body. Contours are then extracted from the thresholded image, and each contour is classified as containing zero, one, or two flies. A contour might contain zero flies on account of image noise or a background object, and could contain two flies if the flies are in contact.

For each contour, its area is computed and used as the input of a decision tree that identifies the number of flies in the contour. In general, each node of a decision tree splits the incoming data into two groups based on one of the input features. In this case, the cost function for evaluating the quality of splits was the Gini impurity [15] $\sum_{k=1}^{J} p_k (1 - p_k)$, where p_k is the proportion of the examples in the split of class k, and J is the total number of classes (in this case 3).

During training, the decision tree was constructed so as to minimize the Gini impurity in a greedy fashion, seeking to incrementally improve the homogeneity of the subsets produced by each split. As can be seen in 4, the resulting logic is simple and can be easily interpreted: small contours contain zero flies, medium contours contain one fly, and big contours contain two flies.

4.2 Pipeline Stage: ♂♀ vs ♀♂

If there are two contours with one fly each, the next stage of the image processing pipeline determines which is the male fly and which is the female fly. This classification is done jointly; that is, the classifier is a given a list of the two contours and asked whether that list is ordered male-female or female-male. In general, this is fairly straightforward, since female fruit flies are larger than male fruit flies. But in some cases, such as when the flies are climbing along the walls, making the distinction can be a bit trickier.

There are four input features for this pipeline stage: namely, the area and aspect ratio of both contours (the latter determined via image moment analysis [13]). As shown in Figure 5, these features are rescaled to zero mean and unit variance, then run through a logistic regression. Briefly, the goal of logistic regression is to determine a maximum-likelihood estimate of θ for a hypothesis $h_\theta(x) = 1/\left(1 + \exp\left(-\theta^T x\right)\right)$. This leads to the following single-example update rule:

$$\theta := \theta + \alpha (y - h_\theta(x)) x$$

In this case, a rescaling step is needed before training the logistic regression because contour areas and aspects ratios are of vastly different scales, so the above update rule would otherwise perform quite poorly. To further improve the quality of training, data augmentation was applied by swapping the order of the two contours and their labels for each example.

5 PIPELINE STAGE: ORIENTATION

In the third pipeline stage, the $0 - 360°$ orientation of both flies is determined, and this is done in a way that reduces the machine learning task to a binary classification. As pre-processing, the image is first masked to everything expect the body of one fly, after which point the orientation of the fly is determined using image

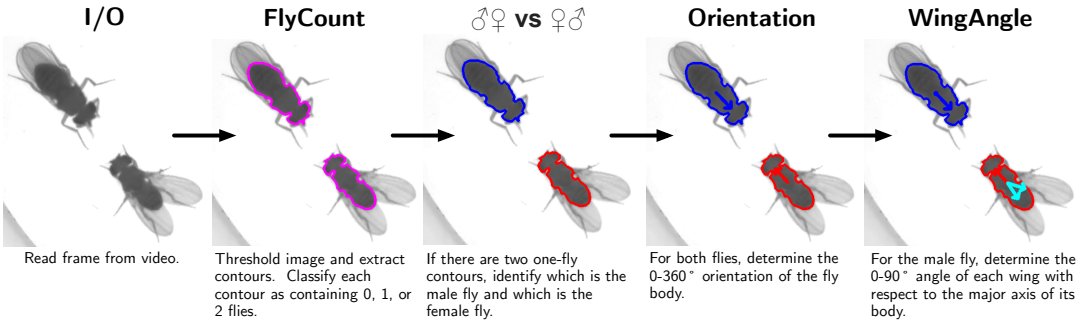

Figure 3: The image processing pipeline developed in this project.

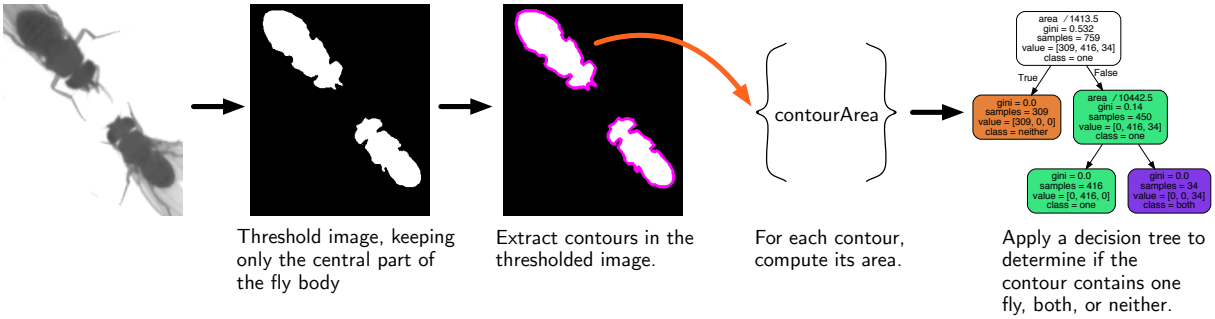

Figure 4: The "FlyCount" pipeline stage.

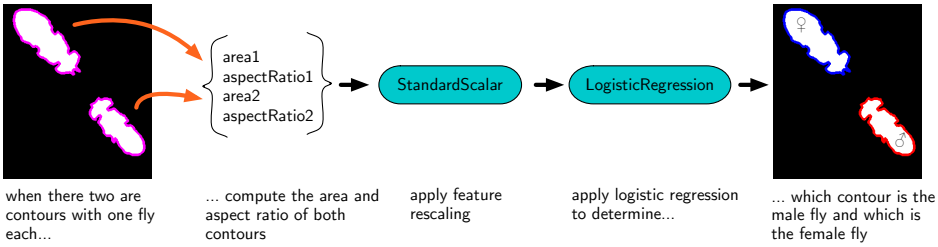

Figure 5: The "♂♀ vs ♀♂" pipeline stage.

moments [13, 16]:

$$\theta \approx \frac{1}{2}\tan^{-1}\left(\frac{2\mu_{11}}{\mu_{20}-\mu_{02}}\right)$$

where μ_{pq} is central image moment of the masked image, defined by the summation:

$$\mu_{pq} = \sum_x \sum_y (x-\bar{x})^p (y-\bar{y})^q f(x,y)$$

where $(\bar{x}, \bar{y})$ is the center of mass of the image and $f(x,y)$ is the intensity at pixel (x,y). Unfortunately, the orientation angle produced using this approach has a sign ambiguity; it cannot discern whether an object is facing forwards or backwards. As a result, I needed to develop a machine learning algorithm to decide if the orientation computed via image moments should be corrected by adding 180°. As shown in Figure 6, I did this by computing a HOG descriptor for the fly image after rotating it by the calculated value of θ, then reduced then dimensionality of this descriptor using principal component analysis (PCA), and finally used logistic regression to decide whether the 180° correction was required.

Briefly, the HOG descriptor [2] captures information about silhouettes in an image by computing a histogram of gradient directions over 8x8 patches throughout the image. In each of these patches, there are 64 pixels; the magnitude and direction of the image gradient is computed at each one. A histogram is then built by binning the pixels into 9 angular ranges, using the gradient magnitude to weight the contribution of each pixel. The histograms of all 8x8 patches are then flattened into vector.

After computing the HOG descriptor for a fly image, the descriptor is projected onto a basis of 15 principal components, or in other words, the eigenvectors corresponding to the 15 largest eigenvectors of $\sum_i x^{(i)} x^{(i)^T}$, where $x^{(i)}$ is the HOG descriptor of the ith training example.

As the final step of this pipeline stage, the dimensionally-reduced HOG descriptor is fed into a logistic regression[2]. Somewhat surprisingly, as shown in Figure 8, even using just two principal components yields a clear distinct between forward-facing and backward-facing flies, although I ended up using a larger number of components to improve test error (in this case, 15 components explained about 60% of the feature variance).

5.1 Pipeline Stage: Wing Angle

In the final pipeline stage, the angles of the right and left wings of the male fly are determined. Similarly to the previous stage, preprocessing is used to construct a region of interest (ROI) containing just the male fly and its wings. That ROI can then be orientated in an upright direction using the results of the preceding pipeline stage. The preprocessing uses median blurring, adaptive thresholding, and erosion to preserve fly wings while removing fly legs.

In my approach, the wing angles are determined separately by dividing the image of the male fly into two halves, one for each wing. As shown in Figure 7, a HOG descriptor is computed for each half-image, which is then projected onto a basis of principal components. Interestingly, a linear relation is quite apparent between the wing angle and just the first principal component (Figure 9). (Ultimately, I used 40 principal components to reduce test error.)

The final step in this pipeline stage is a linear regression on the reduced-dimensionality HOG descriptors. Briefly, linear regression seeks to minimize the mean-squared error between predicted and given labels. The optimal parameters θ can be computed directly by the equation $\theta = \left(X^T X\right)^{-1} X^T y$, where X contains the features of all training examples and y is a vector of their labels. After computing θ, the predicted label given features x is simply $\theta^T x$.

6 RESULTS

For all four pipeline stages, error was evaluated by retaining one third of the dataset for testing; this test set was not used at any point during training. The first three stages of the pipeline were classifiers, so their test error is reported simply as misclassification error. As seen in Table 1, the performance of the classifiers is generally very good. The "FlyCount" classifier does not make any errors since its classification is quite straightforward, while the "♂♀ vs. ♀♂" and "Orientation" classifiers[3] make only occasional errors, which tend to occur when one or both of the flies is climbing along a wall. Since the depth of focus of the camera setup is small, the flies can get quite blurry as they start to climb, and this can make classification difficult even for a human.

"WingAngle" is a regression stage; its performance is given as a standard deviation of error since the mean error is small (0.592°). As shown in Table 1, the standard deviation of test error was 2.92°, which can be interpreted as a 95% confidence interval of about ±6°. Given that fly wings only move about 90°, this error is not insignificant, but in practice wing angle tracking qualitatively looks good when superimposed on a video of the flies (e.g., https://bit.ly/2SGMXTQ). The wing angles over time are also plotted in Figure 10, which reveals there is sufficient resolution to see that wings tend to move in opposite directions[4]. In addition, large oscillations can be seen in the wings at various points; these represent "courtship songs" created by the rapid motion of the male fly's wings.

Model	Train Error	Test Error	N_{train}	N_{test}
FlyCount	0.0%	0.0%	759	253
♂♀ vs. ♀♂	0.2%	0.7%	423	141
♂ orientation	0.0%	0.0%	358	120
♀ orientation	0.8%	0.0%	259	87
WingAngle	$\sigma = 2.06°$	$\sigma = 2.92°$	354	118

Table 1: Model accuracy summary.

Step	Runtime	% total
I/O	3.9 ms	32.5 %
FlyCount	2.2 ms	18.3%
♂♀ vs. ♀♂	0.7 ms	5.8%
Orientation	1.9 ms	15.8%
WingAngle	3.3 ms	27.5%

Table 2: Timing breakdown for the image processing pipeline.

Since one of the key goals of this project was to achieve real-time operation, the timing breakdown of the different stages in the image processing pipeline is shown in Table 2. Overall, the average processing throughput was 84.0 frames per second (FPS) on a 2.8 GHz Intel Core i7 CPU with 16 GB RAM (no GPU acceleration). This was 2.4x faster than the source video rate, indicating that real-time operation was achieved. As it turns out, the most time-consuming step was reading in the video frame itself (3.9 ms), although this was followed closely by the "WingAngle" stage (3.3 ms), which had the most involved preprocessing of any stage.

7 CONCLUSION

In this report, I described a real-time image processing pipeline intended for neurobiology experiments that takes as input a grayscale video of a pair of fruit flies and produces as output an estimate of the position, orientation, and sex of each fly, in addition to the wing angles for the male fly. The pipeline had four distinct stages employing machine learning techniques, including a decision tree, logistic regresion, linear regression, and PCA. For both classification and regression tasks, I demonstrated good accuracy on 1530x1530 frames at a throughput of 84 FPS (without GPU acceleration).

In the "Orientation" and "WingAngle" stages, HOG was used in conjunction with PCA to produce input features for a trained model. In both cases, I was surprised how well this worked even when just one or two principal components were used. I think one reason this approach was successful was that I designed the preprocessing stages in a way that increased HOG variance due to the variable of interest (orientation or wing angle) while decreasing

[2]Data augmentation for this regression was performed by rotating all images 180° and inverting their labels

[3]"Orientation" is treated as a classifier because the role of its logistic regression is to determine whether an image, after being oriented vertically by image moment analysis, needs to be rotated 180°

[4]In fact, the cross-correlation coefficient of the two wing angle waveforms is -0.64.

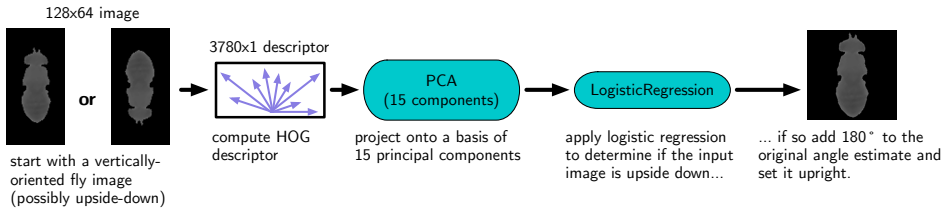

Figure 6: The "Orientation" pipeline stage.

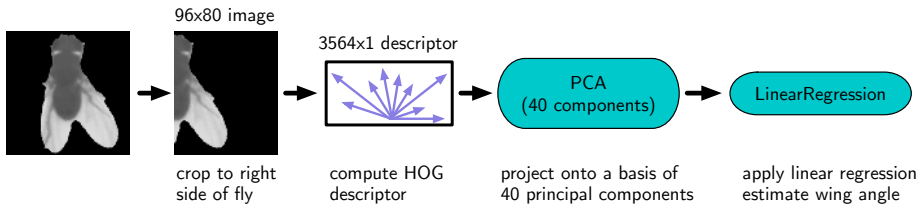

Figure 7: The "WingAngle" pipeline stage.

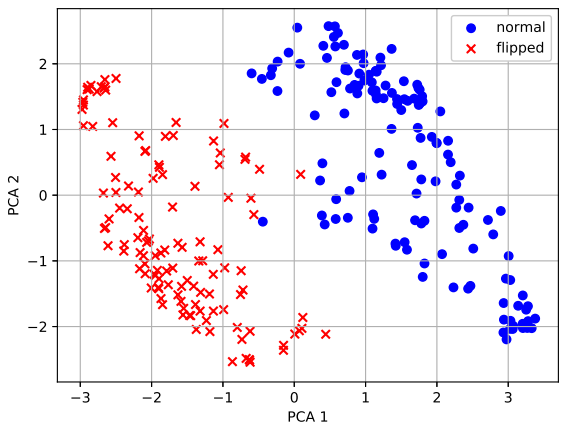

Figure 8: The first two principal components of the HOG descriptor used in the "Orientation" pipeline stage (for female flies).

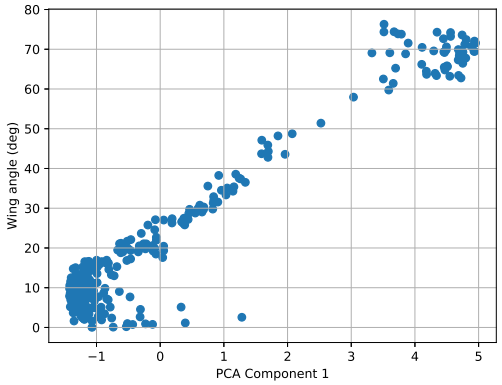

Figure 9: The wing angle can be predicted fairly well with even just the first principal component of the HOG descriptor.

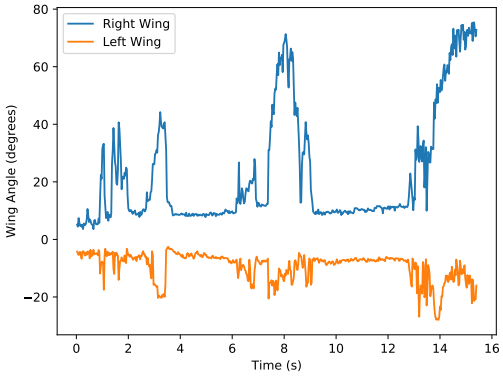

Figure 10: Male fly wing angle, as determined by the "WingAngle" stage of the processing pipeline.

HOG variance due to other variables (fly legs, the other wing, the other fly, background roughness, etc.). This in turn was a useful lesson about the role of preprocessing and feature selection in machine learning.

In the future, there are a number of possible directions to explore. First, I could try measuring leg positions from the video; preliminary experiments suggest that legs tips are selected fairly reliably with a keypoint detector such as SIFT [7], so this could be cast as a keypoint classification problem. It would also be interesting to get access to a computer with CUDA support to see what processing throughput could be achieved using GPU-accelerated OpenCV routines. Finally, I plan to work with Dr. York to explore the application of unsupervised learning methods such as TSNE [8] to the feature vectors generated by the image processing pipeline I developed for this project.

REFERENCES

[1] G. Bradski. 2000. The OpenCV Library. *Dr. Dobb's Journal of Software Tools* (2000).

[2] Navneet Dalal and Bill Triggs. 2005. Histograms of Oriented Gradients for Human Detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01 (CVPR '05)*. IEEE Computer Society, Washington, DC, USA, 886–893. https://doi.org/10.1109/CVPR.2005.177

[3] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. 2009. ImageNet: A Large-Scale Hierarchical Image Database. In *CVPR09*.

[4] Eyrun Eyjolfsdottir, Steve Branson, Xavier P. Burgos-Artizzu, Eric D. Hoopfer, Jonathan Schor, David J. Anderson, and Pietro Perona. 2014. Detecting Social Actions of Fruit Flies. In *Computer Vision – ECCV 2014*. Springer International Publishing, 772–787. https://doi.org/10.1007/978-3-319-10605-2_50

[5] John D. Hunter. 2007. Matplotlib: A 2D Graphics Environment. *Computing in Science & Engineering* 9, 3 (2007), 90–95. https://doi.org/10.1109/mcse.2007.55

[6] Ugne Klibaite, Gordon J Berman, Jessica Cande, David L Stern, and Joshua W Shaevitz. 2017. An unsupervised method for quantifying the behavior of paired animals. *Physical Biology* 14, 1 (feb 2017), 015006. https://doi.org/10.1088/1478-3975/aa5c50

[7] David G. Lowe. 2004. Distinctive Image Features from Scale-Invariant Keypoints. *Int. J. Comput. Vision* 60, 2 (Nov. 2004), 91–110. https://doi.org/10.1023/B:VISI.0000029664.99615.94

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

[9] Alexander Mathis, Pranav Mamidanna, Kevin M. Cury, Taiga Abe, Venkatesh N. Murthy, Mackenzie Weygandt Mathis, and Matthias Bethge. 2018. DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience* 21, 9 (Aug. 2018), 1281–1289. https://doi.org/10.1038/s41593-018-0209-y

[10] Travis Oliphant. 2006–. NumPy: A guide to NumPy. USA: Trelgol Publishing. http://www.numpy.org/ [Online; accessed <today>].

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

[12] Talmo D. Pereira, Diego E. Aldarondo, Lindsay Willmore, Mikhail Kislin, Samuel S.-H. Wang, Mala Murthy, and Joshua W. Shaevitz. 2018. Fast animal pose estimation using deep neural networks. (May 2018). https://doi.org/10.1101/331181

[13] Raphael Candelier. 2018. Tracking object orientation with image moments. http://raphael.candelier.fr/?blog=Image%20Moments [Online; accessed 13-December-2018].

[14] Bryan C. Russell, Antonio Torralba, Kevin P. Murphy, and William T. Freeman. 2008. LabelMe: A Database and Web-Based Tool for Image Annotation. *Int. J. Comput. Vision* 77, 1-3 (May 2008), 157–173. https://doi.org/10.1007/s11263-007-0090-8

[15] Wikipedia contributors. 2018. Decision tree learning — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Decision_tree_learning&oldid=871969828 [Online; accessed 13-December-2018].

[16] Wikipedia contributors. 2018. Image moment — Wikipedia, The Free Encyclopedia. https://en.wikipedia.org/w/index.php?title=Image_moment&oldid=859704100 [Online; accessed 13-December-2018].