Predicting 3d Pose

A method to predict 3d pose of Drosophila melanogaster from 2d pose

Marco Pietro Abrate

Supervisor: Semih Gunel

Project type: semester project

15/08/2019

Contents

1. Introduction **3**

2. State of the art **4**

3. Methods **5**

3.1. Network design 5

3.2. Data preprocessing 7

3.3. Training details 9

4. Results **10**

5. Discussion **10**

Appendices **11**

Abstract

DeepFly3D is a computational pipeline that uses seven cameras for inferring the 3d pose of tethered, adult Drosophila. The cameras are used by the software to triangulate the 3d position of the joints. However, they are positioned all around the platform where the fly stands, occupying the space that could be used for other tools (e.g. lateral lasers for optogenetic stimulation of neurons, ???). A scheme of the setup can be seen in Figure 1. The aim of this project is to reduce the number of cameras mounted on the platform that are used to predict 3d pose of Drosophila melanogaster. The main approach to tackle the problem is to introduce a system that, given 2d joint locations of the limbs of the fly, predicts 3d position without the need of triangulation. This reduces the number of needed cameras from seven to two. The couple of cameras would be located on each side of the insect, left and right. To reach the goal, at first, a relatively simple deep feed forward network is implemented, adapting the one used by Matrinez et al. [4] to predict 3d human pose on the Human3.6M dataset. Secondly, the output data of DeepFly3D (i.e. 2d and 3d joints location) is modified to fit the requirements of the network. Different preprocessing approaches are carried out, in order to reach better performance. Lastly, ???.

The implemented network, trained on ??? and tested on ???, reaches an average test error of ??? per joint. Since the average error is the same/lower/higher as the one obtained by triangulation with seven cameras, the approach results successful/unsuccessful.

Future work ???

In one paragraph indicate - what is the overarching goal, why it is important, what is the subaim you are tackling, what is the approach you took, which results suggest that your approach was successful or unsuccessful, what are perspectives for future work. Please include 1 or 2 representative figures.

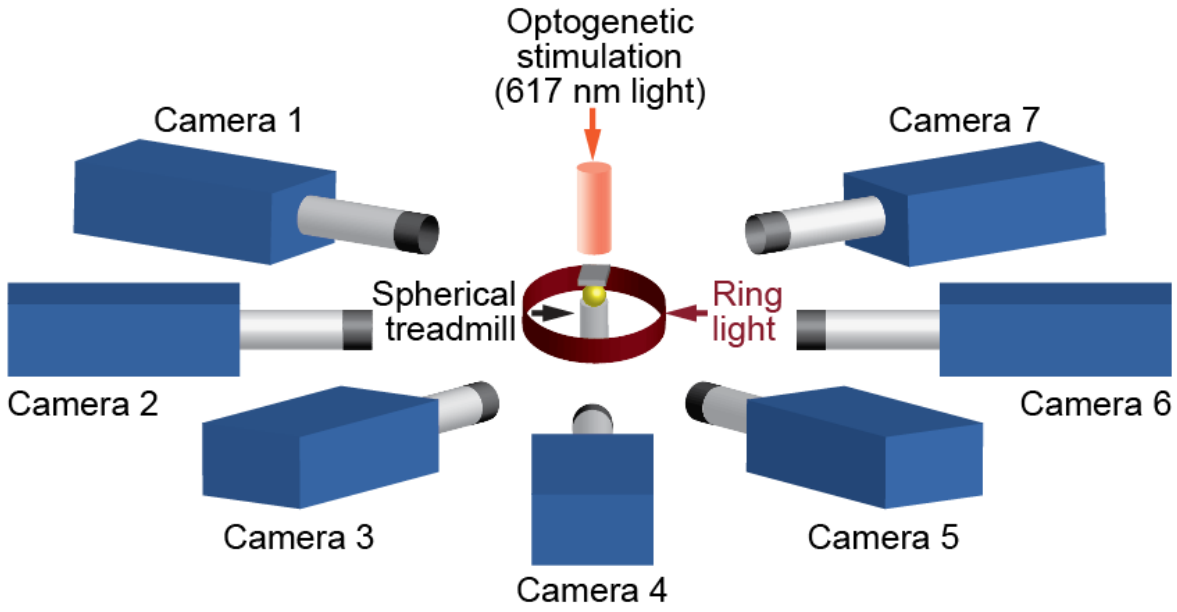


Figure 1: The seven cameras setting used to monitor Drosophila melanogaster

1. Introduction

The aim of DeepFly3D is to have a precise quantification of movements of Drosophila melanogaster, while mounted on the platform, where optogenetic stimulation from above is possible (Figure 1). This is essential for understanding how neurons, biomechanics, and the environment influence and give rise to animal behaviors. 3d joint and appendage tracking promises to accelerate the dissection of neural control principles, particularly in the genetically tractable and numerically simple nervous system of Drosophila melanogaster. To obtain 3d joints position, the software first infers 2d locations from the seven different angles, using a stacked hourglass network. Then triangulates the seven different 2d images to have a full 3d estimation of the six limbs, the abdomen and the antenna of the fly. The software, when triangulating, relies on pictorial structures and belief propagation message passing to detect and correct erroneous pose estimates. In this project, only the six limbs are taken into account, since they are the most important part to study Drosophila movements.

The input to DeepFly3D is video data from seven cameras. As one can see in Figure **??**, the seven cameras, standing on every side of the platform, occlude the possibility to stimulate the fly from a lateral position. This is a disadvantage because ???. This project tackles the problem by predicting the 3d pose of the limbs using only two separate cameras, one for each side of the insect. In particular, the cameras used by the new method are camera number 2 and 6, with respect to Figure 1. This makes it possible to remove the two back cameras (number 1 and 7) and, most importantly, the three frontal ones (number 3, 4 and 5) from the initial setting. This is an improvement because ???.

The prediction of 3d pose using only two cameras is achieved thanks to the power of separating pose approximation into two problems: first the estimation of 2d joints (as it is already done by the software DeepFly3D) and then the prediction of the 3d pose [4]. The contribution of this project is the design and analysis of a neural network that performs 2d-to-3d pose estimation as good as the triangulation method. The newly implemented model is fast, understandable and easy to reproduce.

Expand on the overarching goal, its importance, and the subaim you are tackling.

2. State of the art

The starting point of this project is the deep neural network implemented by Martinez et al. [4], used to predict human 3d pose from 2d joints on the Human3.6M dataset. Their method considerably improves upon the previous best 2d-to-3d pose estimation results, while using a simpler architecture. This result was reached mostly because of an important design change, which is also the driving force of this project. The new method allows to dramatically reduce the number of cameras in the setup of Figure 1. The idea is to use 2d and 3d points as inputs and outputs, in contrast to recent work that has used raw images [3, 6, 8, 9] or 2d probability distributions [9] as inputs, and 3d probabilities [6], 3d motion parameters [10] or basis pose coefficients and camera parameter estimation [9] as outputs. While 2d detections carry less information, their

low dimensionality makes them very appealing for training deep neural networks.

3. Methods

The goal is to estimate joint locations in 3-dimensional space given a 2-dimensional input. More formally, the input is a series of 2d points $\mathbf{x} \in \mathbb{R}^{2n}$, and the output is a series of points in 3d space $\mathbf{y} \in \mathbb{R}^{3n}$. The aim is to learn a function $f : \mathbb{R}^{2n} \rightarrow \mathbb{R}^{3n}$ that minimizes the prediction error over a dataset of N frames:

$$f = \min_f \frac{1}{N} \sum_{i=1}^{N} \mathcal{L}(f(\mathbf{x}_i) - \mathbf{y}_i)$$

In practice, $\mathbf{x}_i$ is obtained from the output of the DeepFly3D stacked hourglass network, and is considered as ground truth. $\mathbf{y}_i$ is the triangulation result of the seven 2d poses, as pointed out above. n is the number of joints to predict. The aim is to train a model able to predict 3d poses $\mathbf{y}_i$ using 2d inputs $\mathbf{x}_i$. The network is trained to predict one side of the fly. Once the network is trained, it can be used also for the other side of the insect. To predict all six limbs, two cameras are necessary. The focus is on systems where f is a deep neural network.

3.1. Network design

The method is based on a simple, deep, multilayer neural network with batch normalization [2], dropout [7], Rectified Linear Units (RELUs) [5] and residual connections [1]. Figure 2 presents the scheme of the network. Two extra linear layers are not depicted in the figure: one applied to the input, which increases its dimensionality to 1024, and one applied before the final prediction, that produces outputs of size $3n$. In the experiments, 2 residual blocks are added. This means that there are 6 linear layers in total, and the model contains between 4 and 5 million trainable parameters. Hereafter, the building blocks of the implemented network are expanded.

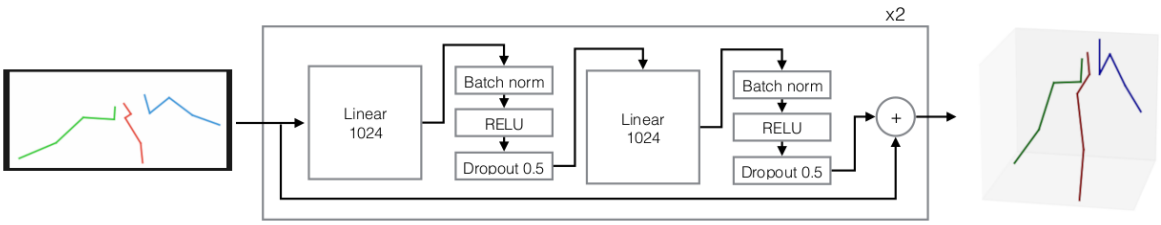


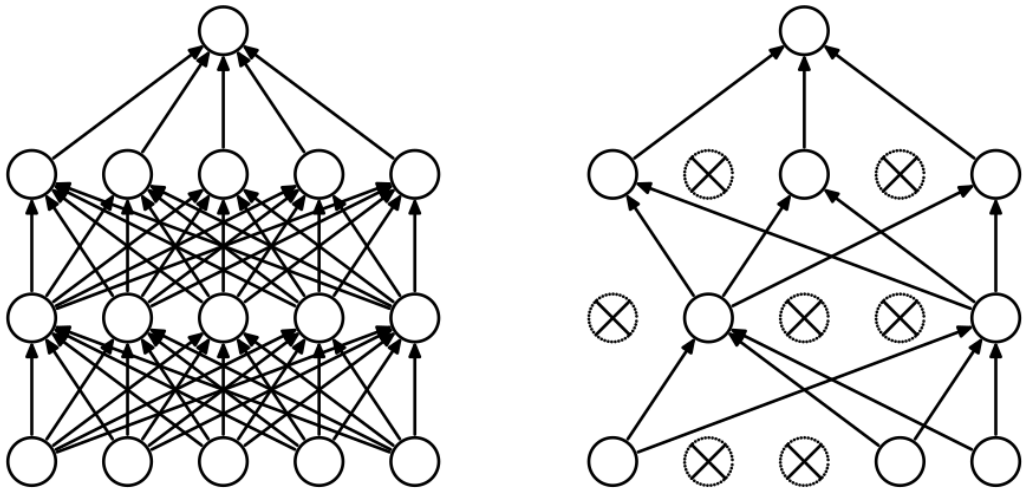
Figure 2: Scheme of the system used in the project. The building block is a linear layer, followed by batch normalization, drop out and a RELU activation. This is repeated twice, and the two blocks are wrapped in a residual connection. The outer block is repeated twice.

- **Batch normalization** It allows to use much higher learning rates and be less careful about the initialization parameters. This method takes a step towards reducing internal covariate shift, and in doing so dramatically accelerates the training [2]. The technique consists in whitening each scalar feature independently, by making it have zero mean and variance of one. For a layer with d-dimensional input $\mathbf{x} = (x^{(1)}, ..., x^{(d)})$, the normalization is

$$\hat{x}^{(k)} = \frac{x^{(k)} - \mathbb{E}[x^{(k)}]}{\sqrt{Var[x^{(k)}]}}$$

- **Dropout** It prevents overfitting and provides a way of approximately combine exponentially many neural network architectures efficiently. The term refers to temporarily dropping out units (hidden and visible) in a neural network, along with all their incoming and outgoing connections, as shown in Figure 3. In this project system, each unit is retained with a fixed probability $p = 0.5$, which seems to be optimal for a wide range of tasks [7]. Training a neural network with dropout can be seen as training a collection of 2^n thinned networks with weight sharing. At test time, a single neural net is used, without dropout. The weights of this network are multiplied by p.

- **Linear RELU-layers** Since the system is dealing with low-dimensional points as inputs and outputs, simple and computationally cheap linear layers can be used. RELUs are a standard choice to add non-linearities in deep neural networks [5].



(a) Standard Neural Net (b) After applying dropout.

Figure 3: **Left**: A standard neural network with two hidden layers. **Right**: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped.

- **Residual connection** Since deeper neural network are more difficult to train, a residual learning framework is used to ease the training, gain accuracy from increased depth and avoid degradation [1].

- **Max-norm constraint** This is a constrained on the weights of each layer, so that their maximum norm is less or equal to 1. Martinez et al. [4] showed that this technique, coupled with batch normalization, stabilizes training and improves generalization when the distribution differs between training and test examples.

3.2. Data preprocessing

One important aspect of the work carried by Martinez et al. [4] is data preprocessing. Since the dataset used in this project treats insects instead of humans, different approaches must be taken in order to make frames more suitable for the network. Hereafter, the list of preprocessing methods used in this project is presented.

- **Standard normalization** Subtraction of the mean and division by the standard

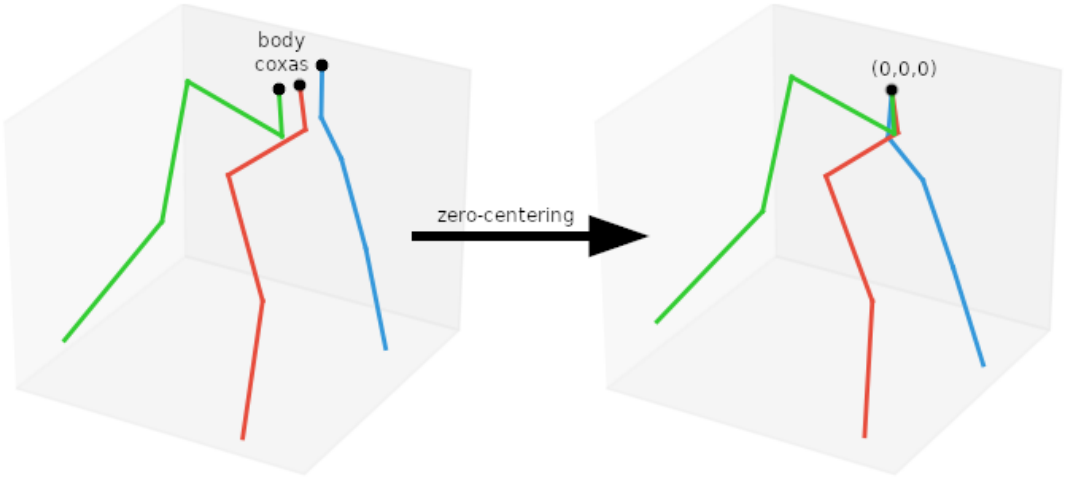


Figure 4: **Left**: Original 3d pose. **Right**: Same 3d pose after applying zero-centering around the three body coxas.

deviation to the 2d inputs and 3d outputs.

- **Zero-centering** Prediction of 3d global position is too complex. For humans, the problem is tackled by zero-centering the 3d pose around the hip joint. For Drosophila melanogaster, the idea is to zero-center the 3d location of all joints around the three body coxas. This is possible because all three of them are always approximately in the same position (TODO: put variance in the dataset). An example of this method, applied on one frame, is shown in Figure 4. The same approach is taken for 2d images.

- **Camera coordinates** Since it is unrealistic to expect the network to infer the 3d joint positions in an arbitrary coordinate space, a natural choice of global coordinate frame is the camera frame. This makes the 2d to 3d problem similar across different cameras, preventing overfitting to a particular global coordinate frame.

$$\mathbf{x}_i^{(proj)} = \text{hstack}(\mathbf{R}, \mathbf{t}) \cdot \text{vstack}(\mathbf{x}_i, \mathbf{1})$$

Where $\mathbf{R}$ is the rotation matrix, $\mathbf{t}$ is the translation vector, hstack and vstack

stack matrices column and row wise, respectively.

Now, the methods that were tried out, but did not result successful, are listed:

- **Data augmentation** The dataset has, for each frame, information about all the seven cameras. Therefore, it was possible to train the network adding information from other cameras, besides the two already considered as a baseline. In particular, images from cameras number 1 and 3 (Figure 1) were added to the training dataset. Doing so, even with the above-mentioned camera projection, was not useful for predicting 3d position of the fly with 2d data coming from cameras number 2 and 6. The average test error increased to ???.

- **Procrustes analysis** The dataset is made of recordings from different experiments. Before implementing the camera coordinates, procrustes analysis, which finds optimal translation, rotation and scaling parameters, was used to superimpose body coxas location.

- **Low-pass filter** The movements of the fly are fast and the triangulated 3d joints location oscillate in a small space with high frequency, even when the insect stands still. A low-pass filter was used to smooth the dataset inputs and outputs.

3.3. Training details

- dataset details
- epochs
- optimizer: adam [22 martinez]
- learning rate
- batch size
- Kaiming initialization

- how long does it take to train the network?
- and to predict on a test frame?

4. Results

Justify why you chose specific approaches to tackle the subaim. Summarize your findings with figures that have readable plots with labeled axes and scales, refer to all figures in the text, have numbered references to citable previous work. Do not necessarily present results in chronological order: present results in an order that makes sense for the project goals.

5. Discussion

Which conclusions are supported by your results? How would you have done things differently if you could do the project again? How should future researchers follow up on your work?

References

[1] 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.

[2] Sergey Ioffe and Christian Szegedy. “Batch normalization: Accelerating deep network training by reducing internal covariate shift”. In: *arXiv preprint arXiv:1502.03167* (2015).

[3] Sijin Li, Weichen Zhang, and Antoni B Chan. “Maximum-margin structured learning with deep networks for 3d human pose estimation”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2015, pp. 2848–2856.

[4] Julieta Martinez et al. “A Simple yet Effective Baseline for 3D Human Pose Estimation”. In: *The IEEE International Conference on Computer Vision (ICCV)*. Oct. 2017.

[5] Vinod Nair and Geoffrey E Hinton. “Rectified linear units improve restricted boltzmann machines”. In: *Proceedings of the 27th international conference on machine learning (ICML-10)*. 2010, pp. 807–814.

[6] Georgios Pavlakos et al. “Coarse-To-Fine Volumetric Prediction for Single-Image 3D Human Pose”. In: *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. July 2017.

[7] Nitish Srivastava et al. “Dropout: a simple way to prevent neural networks from overfitting”. In: *The journal of machine learning research* 15.1 (2014), pp. 1929–1958.

[8] Xiaowei Zhou et al. “Sparse representation for 3D shape estimation: A convex relaxation approach”. In: *IEEE transactions on pattern analysis and machine intelligence* 39.8 (2016), pp. 1648–1661.

[9] Xiaowei Zhou et al. “Sparseness meets deepness: 3D human pose estimation from monocular video”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 4966–4975.

[10] Xingyi Zhou et al. “Deep kinematic pose regression”. In: *European Conference on Computer Vision*. Springer. 2016, pp. 186–201.

Appendices

First appendix

A. First section

B. Second section

Second appendix

C. First section

D. Second section