Thèse n° 9124

Using Animal Motion Capture to Learn Neural Representations

Présentée le 20 mai 2022

Faculté informatique et communications
Laboratoire de vision par ordinateur
Programme doctoral en informatique et communications

pour l'obtention du grade de Docteur ès Sciences

par

Semih GÜNEL

Acceptée sur proposition du jury

Dr M. Rajman, président du jury
Prof. P. Fua, Prof. P. P. Ramdya, directeurs de thèse
Prof. B. de Bivort, rapporteur
Prof. H. Kjellström, rapporteuse
Prof. A. Mathis, rapporteur

École
polytechnique
fédérale
de Lausanne

2022

Diyar-ı küfrü gezdim beldeler kâşaneler gördüm
Dolaştım mülk-i islamı bütün viraneler gördüm

Bulundum ben dahi dar-üş-şifa-yı Bab-ı Âli'de
Felatun'u beğenmez anda çok divaneler gördüm

—— Ziya Paşa

Acknowledgements

This thesis would not have been complete without the help and support of many people. I take this opportunity to express my gratitude to all of them. My special thanks go to Aylin for giving me strength during my studies. I also want to thank my parents and my sister. I will never forget my friends from the CVLab and Neuroengineering Laboratory. I am grateful to Chin-lin, Florian, Gizem, Laura, Daniel, Stefanie, Sena, Işınsu, Buğra, Erhan, Vidit, Róger, Agata, and many more for all the fun time. I am grateful to Helge, Sina, and Mateusz for their guidance. Lastly, I would like to thank my thesis advisors, Pascal Fua and Pavan Ramdya, for providing the opportunity for this work.

Lausanne, April 11, 2022 Semih Günel

Abstract

Understanding behavior from the neural activity is a fundamental goal in neuroscience. It has practical applications in building robust brain-machine interfaces, human-computer interaction, and assisting patients with neurological disabilities. Despite the ever-growing demand driven by the applications, neural action decoding methods still require laborious manual annotations. Furthermore, a different model must be trained for each new individual, requiring even more annotation and overwhelming the resources of most applications. This thesis proposes a self-supervised way to tackle the unsupervised domain adaptation for the neural action decoding problem. Our method does not require manual annotations and can generalize across individuals. To do that, we propose to use animal motion capture to guide the encoding of neural action representations together with a set of neural and behavioral augmentations. We, therefore, divide the thesis into two parts. For the first part, we propose multiple novel ways to learn robust markerless motion capture systems for animals. In the second part, we then use motion capture to learn an unsupervised neural action decoding algorithm.

To start, we present a novel unsupervised markerless 2D pose estimation system. Extracting pose information from laboratory animals requires significant annotation labor. Our model requires only unrealistic renderings of the target animal, without any manual pose labeling. While deep learning models cannot be trained on unrealistic renderings of laboratory animals due to the domain gap, we propose a deformation-based generative network transforming the renderings into realistic images while at the same time keeping the pose information intact. Our model enables us to train a pose estimation network on generated but realistic images without manual annotations.

Second, we propose a markerless self-calibrating motion capture system on tethered animals. Although extracting a 2D pose is enough for some applications, 2D pose information is incomplete due to information loss from the camera projection. On the other hand, triangulating 3D poses from multiple images requires camera calibration. This tedious process is not ideal for many laboratory setups, for example, a small model organism *Drosophila*. We model a self-calibrating system that uses the animal as the calibration pattern. To make the system even more reliable, we use the shape and skeleton of the animal as the 3D prior, therefore fixing the inference mistakes of the deep learning based methods.

Third, we propose a monocular 3D motion capture algorithm for laboratory animals. Although 3D pose information provides all the necessary information about the movement, multi-camera systems require expensive hardware and multi-view synchronization. We propose a monocular system that can still recover 3D pose without requiring multiple views. We test our model on numerous lab animals, including flies, mice, rats, and macaques. We show that our monocular motion capture system performs reliably and on-par with multi-camera motion capture systems.

Lastly, we propose a novel unsupervised domain adaptation method for neural decoding of actions, which uses the previously mentioned animal markerless motion capture systems for domain adaptation. We show that our neural action decoding algorithm can successfully generalize across individuals. Furthermore, our method can provide better neural action decoding performance on low data regimes when compared to supervised and self-supervised neural action decoding algorithms.

Keywords: 3D human pose estimation, 3D animal pose estimation, 3D computer vision, deep learning, action recognition, neural action decoding, brain-machine interface, self-supervised learning, contrastive learning

Résumé

Le décodage neuronal, c'est-à-dire relier l'activité cérébrale aux comportements, est un objectif fondamental des neurosciences. Il y a de nombreuses applications pratiques incluant la construction d'interfaces cerveau-machine robustes, l'interaction homme-ordinateur et l'aide aux patients souffrants de troubles neurologiques. Malgré la demande toujours croissante suscitée par ces applications, les méthodes de décodage neuronal nécessitent encore de laborieuses annotations manuelles. De plus, un modèle différent doit être entraîné pour chaque nouvel individu, ce qui nécessite encore plus d'annotations et accapare les ressources de la plupart des applications. Cette thèse propose une méthode auto-supervisée pour réaliser l'adaptation d'un domaine non supervisédans le cadre du décodage neuronal. Notre méthode ne nécessite pas d'annotations manuelles et peut être généralisée à tous les individus. Pour ce faire, nous proposons d'utiliser l'enregistrement du mouvement animal pour guider l'encodage des représentations d'actions neuronales avec un ensemble d'augmentations neuronales et comportementales. Nous divisons donc la thèse en deux parties. Pour la première partie, nous proposons plusieurs nouvelles façons d'apprentissage à partir de systèmes de capture de mouvement robustes et sans marqueur pour les animaux. Nous utilisons ensuite les captures de mouvement pour entraîner un algorithme de décodage neuronal non supervisé dans la deuxième partie.

Pour commencer, nous présentons un nouveau système d'estimation de poses 2D sans marqueur et non supervisé. L'extraction d'informations de poses à partir d'animaux de laboratoire nécessite normalement un travail d'annotation important. Notre modèle ne nécessite que des représentations irréalistes de l'animal ciblé, sans aucun travail manuel. Alors que les modèles de Deep Learning ne peuvent pas être entraînés sur des représentations irréalistes d'animaux de laboratoire en raison de l'écart de domaine, nous proposons un réseau génératif basé sur la déformation transformant les représentations irréalistes en images réalistes tout en gardant les informations de poses intactes. Notre modèle nous permet d'entraîner un réseau d'estimation de poses basé sur des images générées mais réalistes sans avoir besoin d'annotations manuelles.

Deuxièmement, nous proposons un système de capture de mouvement auto-calibrant sans marqueur sur des animaux attachés. Bien que l'extraction d'une pose 2D soit suffisante pour certaines applications, les informations de pose 2D sont incomplètes en raison de la projection de la caméra. D'autre part, la triangulation de la pose 3D à partir de plusieurs images nécessite une calibration. Cependant, cette configuration n'est pas idéale pour de nombreuses configurations de laboratoire, par exemple, dans le cas d'un organisme modèle de 3 mm tel que la drosophile. Par conséquent, nous modélisons un système auto-calibrant qui utilise l'animal comme motif de calibration. Pour rendre le système encore plus fiable, nous utilisons la forme et le squelette de l'animal comme base 3D, corrigeant ainsi les erreurs d'inférence des méthodes basées sur le Deep Learning.

Troisièmement, nous proposons un algorithme monoculaire de capture de mouvement 3D pour les animaux de laboratoire. Bien que les informations de pose 3D fournissent toutes les informations nécessaires sur le mouvement, les systèmes multi-caméras nécessitent un matériel coûteux et une synchronisation multi-vues. Nous proposons ainsi un système monoculaire qui permet de toujours retrouver la pose 3D sans nécessiter plusieurs vues. Nous testons notre réseau sur de nombreux animaux de laboratoire, notamment des mouches, des souris, des rats et des macaques. Nous montrons que notre système de capture de mouvement monoculaire fonctionne de manière fiable et comparable aux systèmes de capture de mouvement utilisant plusieurs caméras.

Enfin, nous proposons une nouvelle méthode d'adaptation de domaine non supervisée pour le décodage neuronal, qui utilise les systèmes de capture de mouvement sans marqueur d'animaux mentionnés précédemment pour l'adaptation de domaine. Nous montrons que notre algorithme de décodage neuronal peut se généraliser avec succès à travers différents individus. De plus, notre méthode peut fournir de meilleures performances de décodage neuronal sur des petits ensemble de données par rapport aux algorithmes supervisés et fonctionne mieux que les algorithmes de décodage neuronal non supervisés précédemment développés.

Mots Clés : estimation de pose humaine 3D, estimation de pose animale 3D, vision 3D par ordinateur, Deep Learning, reconnaissance d'action, décodage neuronal, interface cerveau-machine, méthode d'apprentissage auto-supervisée,

Contents

Acknowledgements **i**

Abstract (English) **iii**

Abstract (French) **iv**

List of Figures **xi**

List of Tables **xxi**

1 Introduction **1**

1.1 Problem Definition 4
1.2 Motivation and Applications 5
1.3 Challenges 6
1.4 Contributions 7
1.5 Outline 8

2 Related Work **9**

2.1 Neural Decoding 11
2.2 Human and Animal 3D Pose Estimation 12

3 Unsupervised 2D pose estimation **15**

3.1 Introduction 17
3.2 Method 18
  3.2.1 Spatial Deformation 19
  3.2.2 Appearance Transfer 22
  3.2.3 Pose Estimation 23
  3.2.4 Implementation details 24
3.3 Evaluation 26
  3.3.1 Quality of Unpaired Image Translation 28
  3.3.2 Pose Domain Transformation 29
  3.3.3 2D Pose estimation 30

3.3.4 Ablation study 31
3.4 Conclusion 33

4 Multi-view uncalibrated 3D pose estimation 37
4.1 Introduction 39
4.2 Method 40
4.2.1 From 2D landmarks to 3D trajectories 44
4.2.2 3D pose correction 47
4.2.3 Experimental setup 51
4.2.4 Unsupervised behavioral classification 52
4.3 Evaluation 53
4.3.1 DeepFly3D 53
4.3.2 2D pose performance and improvement using pictorial structures 54
4.3.3 3D pose permits robust unsupervised behavioral classification 54
4.4 Conclusion 55

5 Single-view 3D pose estimaton 65
5.1 Introduction 67
5.2 Method 68
5.2.1 Theoretical basis for LiftPose3D 68
5.2.2 Obtaining 3D pose ground truth data by triangulation 68
5.2.3 LiftPose3D network architecture and optimization 70
5.2.4 Weak perspective augmentation 70
5.2.5 Camera-angle augmentation 70
5.2.6 Linear domain adaptation 71
5.2.7 Experimental systems and conditions 72
5.2.8 2D pose estimation 74
5.2.9 Training the LiftPose3D network 74
5.2.10 Comparing joint angles derived from lifted 3D and 2D poses 76
5.3 Evaluation 76
5.3.1 Predicting 3D poses with a single camera at an arbitrary position 76
5.3.2 Predicting 3D poses despite occluded keypoints 78
5.3.3 Lifting diverse experimental data without 3D ground truth 80
5.3.4 *Drosophila* LiftPose3D station 81
5.4 Conclusion 81

6 Learning Neural Action Representations 89
6.1 Introduction 91
6.2 Method 93
6.2.1 Problem Definition 94

6.2.2 Contrastive Representation Learning 94
6.3 Evaluation 99
6.3.1 Datasets 99
6.3.2 Baselines 103
6.3.3 Benchmarks 106
6.3.4 Results 107
6.3.5 Ablation Study 108
6.4 Conclusion 108

7 Concluding Remarks 111
7.1 Summary 113
7.2 Limitations and Future Work 113

A An appendix 115
7.0.1 Dataset sources and splits. 115

Bibliography 117

Curriculum Vitae 135

List of Figures

1 **2D Animal Pose Estimation:** 2D pose estimation on animals is a challenging problem to solve. The challenges include the shape and texture difference across animals, variation in illumination, occlusion and clutter, variation of viewpoint. **(A, C)** shows a macaque moving inside open enclosure, taken from OpenMonkeyStudio Dataset [14]. **(B)** shows rats within a circular arena taken is from CAPTURE dataset [15]. **(D, E, I)** are taken from Animal-Pose dataset [16]. **(f)** shows a tethered *Drosophila* 2D pose, example is taken from DeepFly3D dataset [11]. **(G)** an example of horse 2D pose taken from Horse-10 dataset [17]. **(H)** Multi-animal mice example is taken from [18]. 4

2 **Domain Gap in Neural Signals:** Neural signals have different structures across individuals. **(A)** MC2P dataset (introduced in the Chapter 6) measure different axons across different individuals in tethered adult *Drosophila*. Images differ in terms of total brightness, the location and number of visible neurons, and the shape and size of axons [25]. **(B)** ECoG (AJILE) dataset has different electrode placements for each subject [26]. Each electrode records seperate regions, therefore convey different information. 6

3 **Approach.** Our most morphologically complex example is the six-legged *Drosophila*: a) We transfer synthetic images and their keypoint annotations to realistically looking images using only unpaired examples of the two domains. b) Our method enables training of a pose detector that c) can be applied to real images for neuroscientific studies. 18

4 **Domain examples with large discrepancy in appearance, shape and pose.** Translating from rendering to real images requires bridging the domain gap without having pixel nor pose correspondences. It is particularly challenging in our setting, as even the realistic fly character shows significant differences in shape (body and limb width) as well as pose (legs stretched). 19

5 **Overview of our deformation-based image translation method.** Our model has two steps. In the first step, the deformation from source domain A to target domain B is estimated for input image $\mathbf{I}^A$ and it's silhouette $\mathbf{S}^A$ via two networks G_D and STN (Spatial Transformer Network). Their output is an explicit deformation field parameterized by the global, affine transformation θ and a local, non-linear warping ϕ. Then, the deformed silhouette is transformed into the full output image $\hat{\mathbf{I}}^B$ with image generator G_I. Discriminators D_S and D_I enable unpaired training. D_S uses the Straight Through Estimator (STE) to backpropagate gradients through thresholding operations. 20

6 **Explicit deformation ensures transfer of keypoints.** The deformation field is inferred as part of a) source image segmentation to target image segmentation transfer (including global, affine transformation) and b) segmentation to target image translation. c) The same deformation field is applied to transfer known keypoints from source to target. 21

7 **Texture discriminator influence.** Without the adversarial discriminator, the image generator is disturbed by an irregular silhouette boundary. In our model, the adversarial D_I creates a link to the deformed silhouettes $\hat{S}^B$ enabling end-to-end training. 23

8 **Qualitative comparison.** Existing unpaired image translation methods can generate realistic images on worm and fish, but exhibit artifacts for the thin legs of the *Drosophila* and zebrafish examples. Ours succeeds on all three classes. . 28

9 **Automatic Pose Annotation.** Our method faithfully transfers poses across domains, while Cycle-GAN, the best performing baseline, loses correspondence on all three datasets. 30

10 **Pose estimation accuracy.** The accumulated error curves show the accuracy (vertical axis) for different PCK thresholds (horizontal axis). Our method clearly outperforms the baselines and approaches the manually supervised reference. . 31

11 **Qualitative pose estimation results.** The estimator provides decent results across all three animals. Occasional failures (last two rows) happen when legs cross, at occlusions, and for the fine fish tail. Training on Cycle-GAN images does not succeed. 32

12 **Ablation study.** All our contributions are important, removing the global spatial transformer leads to lower local details (bends legs), only global transformation does not correct shape and pose differences (thinner, straight legs), and predicting the vector field (not its derivative) produces foldovers. *silhouette estimated from an unpaired target domain image. 33

13 **Ablation study on D_I.** Without D_I, small artifacts in the generated (deformed) segmentation masks lead to unrealistic images. 34

14 **Qualitative image generation results.** Our approach can generate realistic and diverse poses, which are transferred across domains faithfully. Our method works on all three tested animals, including the *Drosophila* dataset with superimposed legs that are on the ball that has no correspondence in the source domain. 35

15 **Pose estimation result comparison.** Training a pose estimator on our generated images yields accurate detections with far less failures when compared to CycleGAN, the best performing baseline. The *Drosophila* case is most challenging as the legs are thin and self-similar. 36

16 Deriving 3D pose from multiple camera views. **(A)** Raw image inputs to the Stacked Hourglass deep network. **(B)** Probability maps output from the trained deep network. For visualization purposes, multiple probability maps have been overlaid for each camera view. **(C)** 2D pose estimates from the Stacked Hourglass deep network after applying pictorial structures and multi-view algorithms. **(D)** 3D pose derived from combining multiple camera views. For visualization purposes, the 3D pose has been projected onto the original 2D camera perspectives. **(E)** 3D pose rendered in 3D coordinates. Immobile thorax-coxa joints and antennal joints have been removed for clarity. 41

17 The DeepFly3D pose estimation pipeline. **(A)** Data acquisition from the multi-camera system. **(B)** Training and retraining of 2D pose. **(C)** 3D pose estimation. 42

18 The DeepFly3D annotation tool. This GUI allows the user to manually annotate joint positions on images from each of 7 cameras. Because this tool can be accessed from a web browser, annotations can be performed in a distributed manner across multiple users more easily. A full description of the annotation tool can be found in the online documentation: https://github.com/NeLy-EPFL/DeepFly3D. Scale bar is 50 pixels. 44

19 Camera calibration. **(A)** Correcting erroneous 2D pose estimations by using epipolar relationships. Only 2D pose estimates without large epipolar errors are used for calibration. x_2 represents a 2D pose estimate from the middle camera. Epipolar lines are indicated as blue and red lines on the image plane. **(B)** The triangulated point, X_T, uses the initial camera parameters. However, due to the coarse initialization of each camera's extrinsic properties, observations from each camera do not agree with one another and do not yield a reasonable 3D position estimate. **(C)** The camera locations are corrected, generating an accurate 3D position estimate by optimizing Equation 13 using only the pruned 2D points. . 57

20 3D pose correction for one leg using the MAP solution and pictorial structures. **(A)** Candidate 3D pose estimates for each keypoint are created by triangulating local maxima from probability maps generated by the Stacked Hourglass deep network. **(B)** For a selection of these candidate estimates, we can assign a probability using Equation 14. However, calculating this probability for each pair of points is computationally intractable. **(C)** By exploiting the chain structure of Equation 14, we can instead pass a probability distribution across layers using a belief propagation algorithm. Messages are passed between layers as a function of parent nodes, describing the belief of the child nodes on each parent node. Grayscale colors represent the calculated belief of each node where darker colors indicate higher belief. **(D)** Corrected pose estimates are obtained during the second backward iteration, by selecting the nodes with largest belief. We discard nodes (x's) that have non-maximal belief during backwards message passing. Note that beliefs have been adjusted after forward message passing. 58

21 DeepFly3D graphical user interface (GUI). The top-left buttons enable operations like 2D pose estimation, camera calibration, and saving the final results. The top-right buttons can be used to visualize the data in different ways: as raw images, probability maps, 2D pose, or the corrected pose following pictorial structures. The bottom-left buttons permit frame-by-frame navigation. A full description of the GUI can be found in the online documentation: https://github.com/NeLy-EPFL/DeepFly3D 58

22 Pose correction using pictorial structures. **(A)** Raw input data from four cameras, focusing on the pretarsus of the middle left leg. **(B)** Probability maps for the pretarsus output from the Stacked Hourglass deep network. Two maxima (white arrowheads) are present on the probability maps for camera 5. The false-positive has a larger unary probability. **(C)** Raw predictions of 2D pose estimation without using pictorial structures. The pretarsus label is incorrectly applied (white arrowhead) in camera 5. By contrast, cameras 4, 6, and 7 are correctly labeled. **(D)** Corrected pose estimation using pictorial structures. The false-positive is removed due to the high error measured in Equation 14. The newly corrected pretarsus label for camera 5 is shown (white arrowhead). 59

23 DeepFly3D graphical user interface (GUI) used with the Human3.6M dataset [29]. To use the DeepFly3D GUI on any new dataset (*Drosophila* or otherwise), users can provide an initial small set of manual annotations. Using these annotations, the software calculates the epipolar geometry, performs camera calibration, and trains the 2D pose estimation deep network. A description of how to adopt DeepFly3D for new datasets can be found in the Methods section and, in greater detail, online: https://github.com/NeLy-EPFL/DeepFly3D 59

24 A schematic of the seven camera spherical treadmill and optogenetic stimulation system that was used in this study. 60

25 Mean Absolute Error distribution. **(A)** PCK (percentage of keypoints) accuracy as a function of mean absolute error (MAE) threshold. **(B)** Evaluating network prediction error in a low data regime. The Stacked Hourglass network (blue circles) shows near asymptotic prediction error (red dashed line), even when trained with only 400 annotated images. After 800 annotations, there are minimal improvements to the MAE. **(C)** MAE for different limb landmarks. Violin plots are overlaid with raw data points (white circles). 61

26 Pose estimation accuracy before and after using pictorial structures. Shown are pixel-wise 2D pose errors/residuals (top) and their respective distributions (bottom) **(A)** before, or **(B)** after applying pictorial structures. Residuals larger than 35 pixels (red circles) represent incorrect keypoint detections. Those below this threshold (blue circles) represent correct keypoint detections. 62

27 Unsupervised behavioral classification of 2D pose data is sensitive to viewing angle. **(A)** Behavioral map derived using 2D pose data from three adjacent cameras (Cameras 1, 2, and 3) but the same animals and experimental time points. Shown are clusters (black outlines) with enriched (yellow), or sparsely (blue) populated data. Different clusters are enriched for data from either **(B)** camera 1, **(C)** camera 2, or **(D)** camera 3. Behavioral embeddings were derived using 1 million frames during 4 s of optogenetic stimulation of MDN>CsChrimson (n=6 flies, n=29 trials), aDN>CsChrimson (n=6 flies, n=30 trials), and wild-type control animals (MDN-GAL4/+: n=4 flies, n=20 trials. aDN-GAL4/+: n=4 flies, n=23 trials). 62

28 Unsupervised behavioral classification of 3D joint angle data. Behavioral embeddings were calculated using 3D joint angles from the same 1 million frames used in Figure 3. **(A)** Behavioral map combining all data during 4 s of optogenetic stimulation of MDN>CsChrimson (n=6 flies, n=29 trials), aDN>CsChrimson (n=6 flies, n=30 trials), and wild-type control animals (For MDN-Gal4/+, n=4 flies, n=20 trials. For aDN-Gal4/+ n=4 flies, n=23 trials). The same behavioral map is shown with only the data from **(B)** MDN>CsChrimson stimulation, **(C)** aDN>CsChrimson stimulation, or **(D)** control animal stimulation. Associated videos reveal that these distinct map regions are enriched for backward walking, antennal grooming, and forward walking, respectively. The following video supplements are available: 63

29 **LiftPose3D predicts 3D poses with a single, flexibly positioned camera A** Ground truth 3D poses of tethered *Drosophila* are triangulated using six camera views (3 cameras per keypoint). **B** LiftPose3D predicts 3D poses using deep network-derived 2D poses from only two cameras (red and blue, 1 camera per keypoint). The coordinates are considered relative to a set of root joints (green). The inputs are scaled up and passed twice through the main processing unit (gray rectangle) comprising batch norm, dropout and ReLU wrapped by a skip connection. **C** The output, 3D half-body poses (blue/red), are compared with triangulated 3D poses. Limbs are labeled by left (L)/right (R) and front (1), mid (2), or hind (3) positions. **D** LiftPose3D can be trained using virtual camera projections of 3D poses to lift from cameras within the angles ψ_z, ψ_y, ψ_x (representing ordered yaw, roll, pitch rotations). **E** Error of 3D poses relative to triangulation using three cameras per keypoint. We compare triangulation error using 2 cameras per keypoint (white), test error for a network trained with known camera parameters (orange) and two angle-invariant networks with narrow (green, $\psi_z = \pm 10°$, $\psi_y = \pm 5°$, $\psi_x = \pm 5°$ with respect to a known camera orientation), or wide ranges (red, $\psi_z = \pm 180°$, $\psi_y = \pm 5°$, $\psi_x = \pm 5°$). **F** Error of lifted 3D poses at different virtual camera orientations of the wide-range lifter network and a network with known camera parameters. Blue dots represent lifting errors for a given projected 2D pose. Orange circles represent averages over the test dataset for a given camera. **G** Error of estimated 3D poses for a network trained and tested on different combination of behavioral data including optogenetically induced backward walking (*MDN*, left), antennal grooming (*aDN*, middle), or spontaneous, unstimulated behaviors (*control*, right). **H** Two representative images from the OpenMonkeyStudio dataset. 2D poses are superimposed (black). **I** 3D poses obtained by triangulating up to 62 cameras (red lines), or using a single camera and LiftPose3D (dashed black lines). **J** Absolute errors for different body parts with respect to total body length. Violin plots represent Gaussian kernel density estimates with bandwidth 0.5, truncated at the 99th percentile and superimposed with the median (gray dot), 25th, and 50th percentiles (black line). 83

30 **LiftPose3D performs 3D poses estimation on freely behaving animals with occluded keypoints. A** *Drosophila* behaving freely within a narrow, transparent enclosure. Using a right-angle prism mirror, ventral (top) and side (bottom) views are recorded with one camera. Afterwards, 2D poses are annotated (colored lines). Ventral 2D poses (green box) are used to lift 3D poses. **B** Keypoints near the prism mirror (red and blue) can be tracked in both views and triangulated. Other keypoints (gray) are only visible ventrally and thus have no 3D ground truth. Unilateral ground truth for both sides are obtained by registering the orientation and position of ventral images of the fly. **C** Training data consist of full ventral 2D poses and their corresponding partial 3D poses. **D** Following training, LiftPose3D can predict 3D poses for new ventral view 2D poses. **E** Joint-wise and overall absolute errors of the network's 3D pose predictions for freely behaving *Drosophila*. **F** A similar data preprocessing approach is used to lift ventral view 2D poses of mice (green boxes) walking within a narrow enclosure and tracked using LocoMouse software. LocoMouse ground truth (blue and red) and LiftPose3D (orange) pose trajectories are shown for the right forepaw (top) and hindpaw (bottom) during one walking epoch. Arrowheads indicate where LiftPose3D lifting of the ventral view can be used to correct LocoMouse side view tracking errors (red). Asterisks indicate where inaccuracies in the LocoMouse ventral view ground truth (red) disrupt LiftPose3D's side view predictions (orange). **G** Absolute errors of LiftPose3D and LocoMouse side view predictions for six keypoints with respect to manually annotated ground truth data. **H** Camera image from the CAPTURE dataset superimposed with the annotated 2D pose (left). LiftPose3D uses this 2D pose to recover the full 3D pose (right). **I** LiftPose3D can be trained to lift 3D poses of a freely moving rat with occluded keypoints (open circles). **J** Histograms of the measured lengths of the spinal segment for two different animals. **K** Error distribution over all keypoints for the CAPTURE dataset. 84

31 **A pretrained LiftPose3D network predicts 3D poses for diverse data and when triangulation is impossible.** **A** Linear domain adaptation between domain A (fly on a spherical treadmill) and domain B (fly on a flat surface). 2D poses in B are mapped to A by a linear transformation d_2 then lifted with a network trained only on domain A poses. After lifting, the 3D poses are mapped back to B by another linear transformation d_3. **B** A typical 2D pose in domain B mapped into domain A by the best-fit linear transformation d_2 between poses in B and their nearest neighbors in A. **C** Error between mapped pose and k nearest neighbor poses for d_2, d_3 against the number of poses used to train them ($k = 1$ for d_2 and $k = 2$ for d_3). **D** Lifted 3D pose following domain adapation of a ventral domain B 2D pose and lifting with a network trained on domain A data. The prediction is superimposed with the incomplete ground truth 3D pose in domain B. **E** Lifting error following domain adaptation of domain B poses compared with lifting error in the domain A with no domain adaptation. **F** Freely behaving flies were recorded from below using a low-resolution camera. Following body tracking, the region-of-interest containing the fly was cropped and registered. 2D pose estimation was then performed for 24 visible joints. **G** 2D poses are adapted to the prism-mirror domain. These are then lifted to 3D poses using a network pre-trained with prism-mirror data and coarse-grained to match the lower resolution 2D images in the new experimental system. **H** These 3D poses permit the analysis of claw movements in an otherwise unobserved $x-z$ plane (bottom). **I** Freely behaving fly recorded from below using one high-resolution camera. 2D pose estimation was performed for all 30 joints. Following tracking, a region-of-interest containing the fly was cropped and registered. The same LiftPose3D network trained in panel B—but without coarse-graining—was used to predict **J** 3D poses and **K** unobserved claw movements in the $x-z$ plane (bottom). 85

32 **Joint angles resulting from lifting compared with 3D triangulated ground truth and 2D projections.** Joint angles α, β, γ, and ω for the front, mid, and hind left legs during forward walking. Shown are angles computed from 3D triangulation using DeepFly3D (blue), LiftPose3D predictions (red), and ventral 2D projections α', β', γ', and ω' (green). The mean (solid lines) and standard deviation of joint error distributions (transparency) are shown. Joint angles were computed by Monte Carlo sampling and errors were computed by taking the fluctuation in bone lengths. 86

33 **Training and test loss convergence of the LiftPose3D network applied to a variety of datasets.** Shown are the absolute test errors of LiftPose3D for all joints as a function of optimization epoch. Note that the test error is sometimes lower than the training error because we do not apply dropout at test time. **A** Two-camera data of *Drosophila* on a spherical treadmill (each color denotes a different pair of diametrically opposed cameras). **B** OpenMonkeyStudio dataset (each color denotes a different training run). **C** Single-camera data of *Drosophila* behaving freely in the right-angle prism mirror system. **D** LocoMouse dataset. **E** CAPTURE dataset. 86

34 ***Drosophila* LiftPose3D station.** **A** CAD drawing of the LiftPose3D station indicating major components (color-coded). **B** Photo of the LiftPose3D station. **C** Electronic circuit for building the illumination module on a pre-fabricated prototyping board, electronic components, and additional wiring are indicated (color-coded). **D** Printed circuit board provided as an alternative to the pre-fabricated board for building the illumination module. 87

35 **Our Motion Capture and Two-Photon (MC2P) Dataset.** A tethered fly (*Drosophila melanogaster*) is recorded using six multi-view infrared cameras and a two-photon microscope. The resulting dataset includes the following. **(A)** 2D poses extracted from all views (only three are shown), calculated on grayscale images. **(B)** 3D poses triangulated from the 2D views. **(C)** Synchronized, registered, and denoised single-channel fluorescence calcium imaging data using a two-photon microscope. Shown are color-coded activity patterns for populations of descending neurons from the brain. These carry action information (red is active, blue is inactive). **(D)** Annotations for eight animals of eight different behaviors, four of which are shown here. **(E)** Manual neural segmentation has been performed to extract neural activity traces for each neuron. We will release our MC2P publicly. 91

36 **Domain gap across nervous systems and learned representations. (Top row)** **(A)** Neural imaging data from four different animals in our MC2P dataset. Images differ in terms of total brightness, the location and number of visible neurons, and the shape and size of axons. **(Bottom row)** t-SNE embeddings of different neural representation algorithms on MC2P dataset. Each color denotes a different fly. The two red dots are embeddings of the same action label for two different animals. t-SNE embeddings of **(B)** Raw neural data. **(C)** Contrastive SimCLR [5] representation trained on behavioral and neural pairs, **(D)** SimCLR and domain adaptation using a two-layer MLP discriminator and a Gradient Reversal Layer [241]. **(E)** Ours. The identify of the animals is discarded and the semantic structure is preserved, as evidenced by the fact that the two red dots are very close to one another. 93

37 **Overview of our contrastive learning-based neural action representation learning approach.** First, we sample a synchronized set of behavioral and neural frames, $(\mathbf{b_i}, \mathbf{n_i})$. Then, we augment these data using randomly sampled augmentation functions t_b and t_n. Encoders f_b and f_n then generate intermediate representations $\mathbf{h}^b$ and $\mathbf{h}^n$, which are then projected into $\mathbf{z}_b$ and $\mathbf{z}_t$ by two separate projection heads g_b and g_n. We maximize the similarity between the two projections using an InfoNCE loss. At test time, the red branch and $\mathbf{h}_\mathbf{i}^n$ is used for neural decoding. 95

38 **Swapping augmentation.** Each 3D pose in the motion sequence of Domain 2 is randomly replaced with its neighbors, from the set of domains $\hat{s} \in \mathscr{S}/s$, which includes Domain 1 and Domain 3. The swapping augmentation hides identity information, while keeping pose changes in the sequence minimal. 97

39 **Motion Capture and two-photon dataset statistics.** Visualizing **(A)** the number of annotations per animal and **(B)** the distribution of the durations of each behavior across animals. Unlike scripted human behaviors, animal behaviors occur spontaneously. The total number of behaviors and their durations do not follow a uniform distribution, therefore making it harder to model. 99

40 **Visualizing the temporal correlation between behavioral and neural energies on multiple animals.** The behavioral and neural energies are calculated as the normalized distances between consecutive frames. The multi-modal energies show a similar temporal pattern. The slower neural energy decay is due to the calcium dynamics. 99

41 **Domain Gap in the H3.6M dataset.** Similar to the domain gap across nervous systems, RGB images show a significant domain gap when the camera angle changes across individuals. We guide action recognition across cameras in RGB images using 3D poses and behavioral swapping. 101

42 **Changing window size for the swapping of behavioral modality on the MC2P dataset.** Statistics of the behavioral modality as a function of changing the window size. Decreasing the window size increases clustering homogeneity and Mean Maximum Discrepancy (MMD) when applied to the raw data, therefore suggesting higher quality swapping in individual poses instead of sequences of poses. Swapping augmentation with a smaller window size lowers the degree of perturbation, quantified by Mean Squared Distance. However, identity recognition accuracy does not change considerably when swapping is done with different window sizes. 101

List of Tables

1 **Structured similarity (SSIM) comparison**. The explicit modeling of deformation outperforms baselines, particularly on the complex *Drosophila* images showing complex poses. 29

2 **Pose estimation accuracy comparison on *Drosophila melanogaster*.** A similar improvement as for Drosophila is attained on the other tested laboratory animals, with a particularly big improvements on the zebrafish. Pose-invariant training improves results. 29

3 **Pose estimation accuracy on *C. elegans* and *D. rerio*.** Our method significantly outperforms all baselines and approaches the supervised baseline. Units are given in round brackets. 30

4 **Detailed ablation study on *Drosophila Melanogaimster*.** All model components contribute to the final reconstruction accuracy. Surprisingly, a smaller batch size improved results. 33

5 **Action Recognition Accuracy.** Single- and Across-Subject action recognition results on the MC2P dataset. Neural MLP results for the single-subject task are removed because single subject often do not have enough labels for every action. Smaller numbers are better for Identity Recognition. Our method performs better than previous neural decoding methods and other self-supervised learning methods in all benchmarks, while at the same time closing the domain gap between animals, as shown by the identity recognition task. 104

6 **Architecture details.** Shown are half of the neural encoder f_n and behavior encoder f_b functions. How these encoders are used is shown in **Fig. 3**. Neural encoder f_n is followed by 1D convolutions similar to the behavioral encoder f_b, by replacing the number of filters. Both encoders produce 128 dimensional output, while first half of the neural encoder do not downsample on the temporal axis. *mp* denotes a max-pooling layer. Batch Normalization and ReLU activation are added after every convolutional layer. 109

7 Across-subject (A.S.) and identity recognition (I.R.) results on H3.6M dataset[29] using RGB and 3D pose, and on ECoG Move vs Rest [239] using neural ECoG recordings and 2D pose. For Ours, we remove calcium imaging specific augmentations and only use swapping augmentation. Swapping augmentation closes the domain gap for the contrastive learning and strongly improves across-subject action recognition on both datasets. 110

8 **Ablation Study.** Showing the effect of different augmentations on single-subject, across-subject and identity recognition benchmarks. 110

A.1 **OpenWorm videos.** Strain type and timestamp of the used videos published by [155, 156]. 115

Introduction Part 1

The accurate prediction of animal behavior from brain activity is a fundamental challenge in neuroscience with essential applications in developing robust brain-machine interfaces. Neural decoding of actions can increase patients' mobility with disabilities or neuromuscular diseases and improve our understanding of how the nervous system works [1, 2]. However, most neural decoding methods require manual annotat ions that are both tedious to acquire and error-prone [3, 4]. We aim to use and develop more general self-supervised neural decoding in neuroscience [5, 6]. Self-supervised methods aim to use unlabeled data during training, in order to perform downstream tasks with minimal manual annotations. This thesis proposes to use 3D animal motion capture to learn self-supervised neural decoding algorithms.

While understanding the 3D positioning of an object is trivial for humans, it is a challenging problem for a computer processing images. Through our stereo vision, we perceive the surrounding environment, humans, and objects in 3D. This lets us navigate and interact within the 3D world. Much of the 3D perception has to do with understanding how the objects and people are positioned in the environment. 3D pose estimation in the field of computer vision addresses this problem [7, 8]. Its ultimate goal is to understand the 3D pose of people and animals from 2D images and the human visual system.

Animal pose estimation, as depicted by Fig. 1, refers to predicting the joint locations of an animal from a 2D image. Predicting the 2D or 3D pose from photos or videos is a long-standing computer vision problem with multiple applications, including augmented reality, human-computer interaction, security, and telepresence [9, 10]. However, it is a challenging and ill-posed problem. Projection from 3D onto a 2D image results in the loss of depth information and makes the problem of 3D pose estimation ambiguous [11]. A potential solution to resolve some of the ambiguities is to use multiple camera systems [12]. However, this becomes impractical due to the cost and effort of setting up various cameras' calibrated and synchronous approaches. Another solution to resolve the ambiguities of 3D pose estimation is to use depth sensors [13]. However, active RGB-D sensors are power-hungry and depth sensors cannot be used for small animals such as *Drosophila*, due to scaling issues. This makes 3D pose estimation from RGB sensors more attractive.

Despite many years of sustained effort, pose estimation remains a challenging problem due to the variability in visual appearance, viewpoint variation, illumination changes, occlusions, and high dimensionality of the pose representations. In the face of these challenges, existing approaches are still fragile and error-prone in general unconstrained scenarios.

In the remainder of this chapter, we first define neural decoding of actions and 3D pose estimation problems and then briefly discuss a few practical applications of both problems. We then present several key challenges and summarize our main contributions. Finally, we give an outline of the thesis.

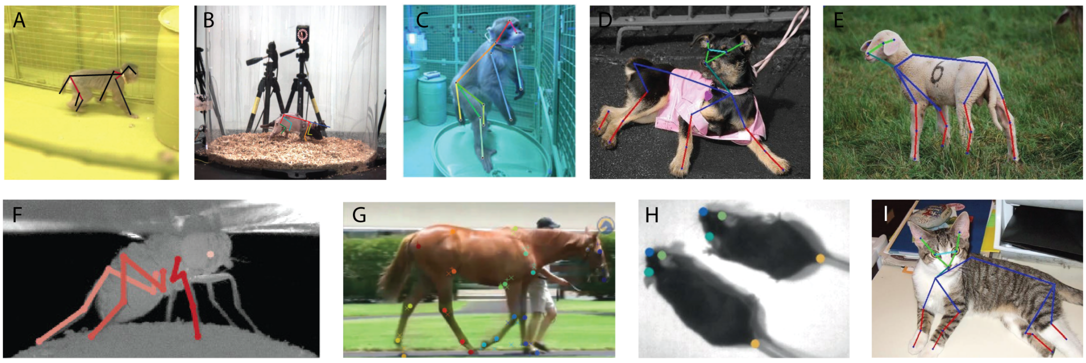


Figure 1 – **2D Animal Pose Estimation:** 2D pose estimation on animals is a challenging problem to solve. The challenges include the shape and texture difference across animals, variation in illumination, occlusion and clutter, variation of viewpoint. **(A, C)** shows a macaque moving inside open enclosure, taken from OpenMonkeyStudio Dataset [14]. **(B)** shows rats within a circular arena taken is from CAPTURE dataset [15]. **(D, E, I)** are taken from Animal-Pose dataset [16]. **(f)** shows a tethered *Drosophila* 2D pose, example is taken from DeepFly3D dataset [11]. **(G)** an example of horse 2D pose taken from Horse-10 dataset [17]. **(H)** Multi-animal mice example is taken from [18].

1.1 Problem Definition

For neural decoding of actions, our goal is to learn an unsupervised signal encoder function that maps a set of neural signal into a low-dimensional representation. We aim for our learned representation to be representative of the underlying action label while being agnostic to animal identity. We assume that we are not given action labels during unsupervised training. Also, note that we do not know when the action starts and ends in the captured data. We have a series of unknown actions performed by a different animal. This makes our problem harder. We assume we are given multi-modal data, neural signals together with motion capture, and we assume the data is captured such that the two modalities are always synchronized (paired) without human intervention and therefore describe the same set of events.

We formulate 3D pose estimation as a regression problem. Given an image or an image sequence, we aim to find a mapping function from robust image features to the 3D pose of an animal. The 3D animal pose can be represented in various ways, including kinematic trees or a set of 3D animal body joint locations. We adopt the latter one, in which we represent the 3D pose of an animal in terms of a skeleton. The output pose is defined in the camera coordinate system and consists of 3D joint locations relative to that of a root joint. We predict the relative configuration of the 3D joint locations with respect to a root joint location and do not consider the absolute 3D joint locations in the camera coordinate system. Therefore, our pose predictions are animal-centric and are not absolute.

1.2 Motivation and Applications

Neural decoding algorithms are necessary for creating robust brain-machine interfaces, human-computer interaction, and assisting patients with neurological disabilities. The automatic and reliable estimation of the 3D human pose has emerged as a pressing need for various sectors and finds numerous applications ranging from augmented reality to surveillance. A few examples are described below.

Disease Treatment: Neural decoding algorithms are essential for treating neurological disorders. Neural decoding methods can be used for decoding speech from a paralyzed person with anarthria [19]. These methods can also restore upper limb movement and sensation through intracortical brain-computer interfaces [20]. Other brain-machine interfaces can be employed for vision restoration with visual prosthetics [21]. On the other hand, 3D human pose estimation can measure the changes in people's physical activities with movement disorders, such as Parkinson's disease and Tourette syndrome.

Human-Computer Interaction: Commercial applications of neural decoding algorithms will be an essential part of the human-computer interaction. The algorithms will allow users to interact with computers using neural signals. Before this is possible, 3D pose information is still a reliable cue to estimate the activity and gesture of a person and ultimately provides a natural computer interface by which human gestures can control computers. This allows for more natural and seamless interaction between humans and computers than the traditional means, including keyboards and touchscreens [22].

Animation and Games. Character animation is essential for animated movies and online games. Conventional approaches rely on motion capture with marker systems to recover the 3D pose of an actor and transfer it to an avatar. 3D pose estimation offers a convenient way to replace this approach.

Augmented Reality and Telepresence: Estimating the 3D human and animal pose and shape is an initial step towards creating digital 3D avatars in the scene [23]. This allows people to interact virtually in augmented reality even when they are far apart.

Security and Surveillance: Human pose and motion provide valuable information about the action and intent of a person in a video-based intelligent surveillance system. Since manual monitoring of the video footage in an extensive network of cameras is impractical and prone to human errors, a markerless motion capture system can assist security personnel in detecting unusual and abnormal activities [24].

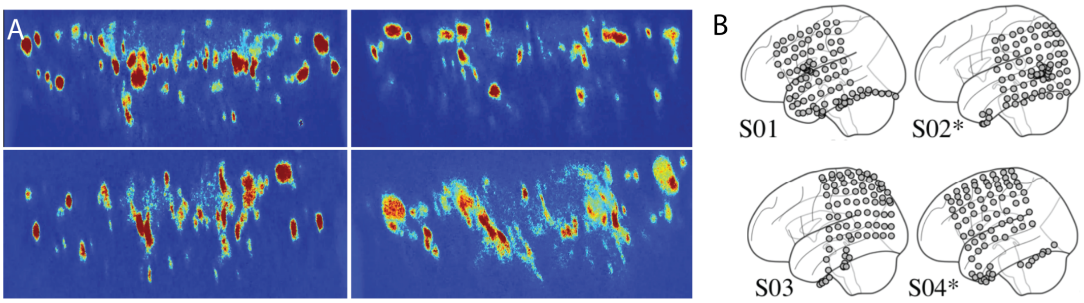


Figure 2 – **Domain Gap in Neural Signals:** Neural signals have different structures across individuals. **(A)** MC2P dataset (introduced in the Chapter 6) measure different axons across different individuals in tethered adult *Drosophila*. Images differ in terms of total brightness, the location and number of visible neurons, and the shape and size of axons [25]. **(B)** ECoG (AJILE) dataset has different electrode placements for each subject [26]. Each electrode records seperate regions, therefore convey different information.

1.3 Challenges

Neural decoding is challenging due to the high signal-to-noise ratio and missing biological understanding of the neural basis of behavior. Also, the domain gap across individuals makes training general models non-trivial. 3D pose estimation of humans and animals is an ambiguous task due to the loss of depth information resulting from projection from 3D to 2D. The problem is complicated by other factors, such as variance in elimination, change in the viewpoint, and challenging case of collecting ground-truth data.

Neural Dynamics: Relevant features for neural decoding often lack a clear baseline, are non-stationary, and occur in a small percentage of the total recording electrodes or pixels [27]. Neural signals also include high noise due to the imaging techniques and electrode placement. Neuroscientists use denoising algorithms such as spike inference to employ neural prior to reducing noise in the neural data [28]. However, many machine learning algorithms still do not perform well processing the neural signal due to the high signal-to-noise ratio.

Lack of Biological Priors: The neural basis of behavior is not very well understood. Therefore, for many downstream tasks, the ground-truth information is not known. This makes it especially hard to assess the quality of the model predictions. On the other hand, it makes supervised learning algorithms not employable in many cases.

Illumination Variance. Illumination conditions play an essential role in the quality of the 3D animal and human pose estimation. Extracting reliable cues becomes challenging in dim light conditions. Ultimately, pose estimation algorithms should generalize to objects captured in arbitrary illumination conditions.

Viewpoint Variance: Images tend to vary significantly when the animals or humans are viewed from different angles. This, in turn, negatively impacts the accuracy and robustness of pose estimation algorithms. Therefore, it is essential to gain invariance against the changes in viewpoint either by collecting a large dataset of images from different viewing angles or extracting viewpoint-invariant features. This is especially important for animal pose estimation, wherein animals are imaged only through a short range of angles in specific setups.

Collecting Ground-Truth Data: Collecting ground-truth 3D human or animal pose data requires annotation effort or expensive marker-based optical motion capture systems [29]. The lack of ground-truth annotations necessitates making the best of limited data. This is especially important for multiple lab animals, where collecting human dataset size labels seems very expensive. This would be possible by learning more unsupervised methods. Collecting ground-truth data for neural decoding is also challenging since the data needs to be annotated by expert neuroscientists.

1.4 Contributions

The main goal of this thesis is to create an unsupervised neural decoding algorithm. For this, we first develop efficient and accurate motion capture methods for animals without using markers. We demonstrate the effectiveness and versatility of our approaches in a wide range of datasets while addressing the challenges mentioned above of pose estimation and neural decoding. We describe below the main contributions of this thesis.

Unsupervised 2D Pose Estimation: We propose the first work of unsupervised pose estimation using unrealistic renderings of the target model. Although previous methods worked on unsupervised pose estimation with human models, they were only successful with realistic renderings. Our work is the first to make an unsupervised pose estimation possible without an accurate model. To make this possible, we propose to use image-to-image translation methods. During translation, we decompose the problem into shape and texture prediction stages. We propose using explicit deformation and gradient to keep the pose information intact during shape translation. For non-differentiable operations like thresholding, we use gradient estimators.

Multi-view uncalibrated 3D pose estimation: We are the first work to extract 3D pose of model organism *Drosophila*. Before that, reliably estimating the 3D pose of the tiny animal was not possible under the current imaging conditions. We propose a self-calibrating and self-correcting motion capture method, which can train itself using active learning. Our self-correction mechanism also significantly outperforms naive triangulation.

Single-view 3D pose estimation: State-of-the-art approaches to monocular 3D animal pose estimation rely on deep learning. They typically involve regressing 2D joint locations from multiple image coordinates from which 3D coordinates are inferred. However, these approaches require expensive multi-view imaging setups. We propose a monocular method for 3D pose estimation on animals. We offer a new domain-adaptation method to make these models work across imaging setup and individuals. We test our method on multiple lab animals, including Drosophila, macaques, mice.

Learning Neural Action Representations: We develop a new unsupervised domain adaptive neural decoding algorithm. Most existing approaches use supervised models, which do not easily generalize across individuals. Instead, we propose a multi-modal self-supervised learning algorithm. We specifically use motion capture to guide the neural representations across domains. Our method yields improvements compared to previous approaches, especially compared to supervised and self-supervised state-of-art neural decoding, when tested across individuals.

1.5 Outline

The remainder of this thesis is organized as follows. We begin Chapter 2 with an overview of the relevant literature on 2D and 3D human and animal pose estimation. Chapter 3 presents our unsupervised 2D animal pose estimation approach from unrealistic animal models. We demonstrate using multiple laboratory animals, including Drosophila, zebrafish, and C. elegans. Chapter 4 introduces our markerless 3D motion capture system for *Drosophila*, which self-calibrates using the 2D pose estimation itself. In Chapter 5, we present a monocular motion capture for laboratory animals. We show that our monocular motion capture system performs similarly to the previous multi-view motion capture systems. Chapter 6 introduces our self-supervised neural action representation learning algorithm for unsupervised neural decoding. Finally, Chapter 7 concludes the thesis with a summary and brief discussion of future research directions.

Related Work Part 2

We start this chapter by reviewing existing neural decoding and human and animal 3D pose estimation, with numerous applications ranging from surveillance to augmented reality and brain-machine interfaces. We then give a brief overview of state-of-the-art.

2.1 Neural Decoding

Recent technological advances have enabled large-scale simultaneous recordings of neural activity and behavior in animals, including rodents, macaques, humans, and the vinegar fly, *Drosophila melanogaster* [30–36]. These advances helped studies of the sensory perception [37, 38], motor control [39], learning and memory [40] and innate behaviors [41]. For preprocessing of two-photon neural data, studies extract neural signals through using ROI detection and constrained nonnegative matrix factorization (CNMFe) [42, 43]. Second, various algorithms have been proposed to solve the deconvolution problem [44, 45]. After pre-preprocessing, to derive a representation for these neural signals, models such as recurrent models [46–48], variational autoencoders [33, 49], and dynamical systems [50, 51] can be used.

Advances in neural recording and neural decoding algorithms also have driven recent progress in BMI approaches. Neural decoders can be used to increase the mobility of patients with disabilities [20, 52], or neuromuscular diseases [2], and can expand our understanding of how the nervous system works [1]. However, these algorithms typically rely on supervised learning [3] and require manual annotations of training data that are both tedious to acquire and error prone [53]. Labeling behavioral-neural datasets requires expensive and arduous manual labor by trained scientists, thus often leaving the vast majority of data unlabeled. Furthermore, decoding algorithms that are useful in the real world must adapt to changing neural signals [54, 55]. Therefore, the ability to infer information from challenging neural data or neural decoding is essential for the development of effective brain-machine interfaces and closed-loop experimentation [56, 57].

There have been previous approaches developed to extract behavioral information from neural data [3, 58, 59]. Training algorithms have decoded neural activity on controlled experimental data with repeated trials. Common implementations include Bayesian decoders [60, 61], and machine-learning models such as support vector machines, k-nearest neighbors, multilayer perceptron, recurrent neural network have been implemented in various studies [3, 62]. Other methods have focused on identifying these two modalities using simple methods, such as correlation analysis, generalized linear models [63–65], or regressive methods [58]. As an application of neural decoding, recent examples include interpreting arm trajectories, and finger movements [66]. Decoded neural signals have been used to control robotic arms and construct BCIs [67]. Neural decoding can be done on different modalities, including neural oscillations scalp electroencephalography (EEG) intracranial electrocorticography (ECoG) [68, 69] or two-photon microscopy [58, 70].

Contrastive learning has been extensively used on human motion sequences to perform action recognition using 3D pose data [71–73] and video-based action understanding [74, 75]. However, a barrier to using these tools in neuroscience is that the statistics of neural data—the locations and sizes of cells for microscopy data, or electrode locations for the signal data—and behavioral data—body part lengths and limb ranges of motion—can be very different from animal to animal, creating a large domain gap. Unlike mammal recordings with limited subject variability, the number of individuals for *Drosophila* recordings are not limited due to practical concerns, therefore making it ideal for testing domain adaptation and representation learning research. Also, *Drosophila* recordings have complete behavior information since we recorded a much more significant percentage of the nervous system.

Existing self-supervised neural decoding methods [26, 68, 76, 77] cannot be used on unlabeled subjects without action labels. A potential solution would be to use domain adaptation techniques to treat each new subject as a new domain. However, existing domain adaptation studies of neural decoding [78, 79] have focused on gradual domain shifts associated with slow changes in sensor measurements rather than the challenge of generalizing across individual subjects. In contrast to these methods, our approach is self-supervised and can generalize to unlabeled subjects at test time without requiring action labels for new individuals. Similarly, it is non-trivial to generalize few-shot domain adaptation methods to multimodal tasks [80, 81]. Thus, the field of neuroscience needs new computational approaches that can extract information from ever-increasing amounts of unlabeled multimodal datasets that also suffer from extensive domain gaps across subjects.

In theory, there are multimodal domain adaptation methods for action recognition that could deal with this gap [82–84]. However, they assume supervision in the form of labeled source data. This is an impractical solution in most laboratory settings, where large amounts of data are collected and limited resources.

This thesis presents the first joint modeling of motion capture and neural modalities to fully extract behavioral information from neural data using a self-supervised learning technique.

2.2 Human and Animal 3D Pose Estimation

Deep learning based human pose estimation methods has recently made significant progress. This is especially true for capturing human movements for which there is enough annotated data to train deep networks [29, 85–89]. A large corpus of the literature focuses on the prediction of 2D key points from images directly [9, 90–94]. There is also a vast literature on capturing 3D pose directly from images or as a function of 2D keypoints instead [95–101]. Weakly [102] and semi-supervised algorithms [103] can further improve the performance of motion capture systems, for example, by using multi-view constraints [104, 105].

To utilize the advances in human motion capture into animal pose estimation, recent efforts have made it possible to perform markerless predictions of 2D poses [14, 18, 106–109]. 2D poses can be converted into 3D animal poses using multi-view stereo systems or lifting methods [110–113]. [114] uses a model-based algorithm, trains on synthetic renderings, and refines on real zebra photographs. However, their quadruped body model does not translate to animals with a different number of legs.

At the same time, realistic animals models have been built for downstream applications such as extracting 3D shape and texture from images for animals such as mice, zebra, and elephant [115–118]. Video and pose data have been used to segment and cluster temporally related behavioral information [119–123].

For pose estimation in *Drosophila*, DeepLabCut provides a user-friendly interface to DeeperCut [18], LEAP [124] tracks limb and appendage landmarks, and DeepFly3D leverages multiple views to capture 3D pose [111]. Nevertheless, all these methods require large amounts of manual labels, which are not available for many animals and cannot be reused when recording the same species in different environments and illumination conditions.

For unsupervised 2D pose estimation methods, image-to-image translation networks can be used. Supervised image-to-image translation methods aim to translate images across domains (e.g., day-to-night, summer-to-winter, photo-to-painting), often using adversarial methods [125] to learn a mapping from input to output images. More recent studies have aimed to translate edges to images [126] and cascaded networks are used to condition on semantic label maps [127]. Style transfer is an image-to-image translation method that works on unpaired examples, aiming to transfer the input image style while preserving the geometry of the target image [128–131]. Initial deep learning approaches optimized an image with respect to the Gram matrix statistics of deep features of the target image [128, 132]. More recent studies have tested different architectures and loss functions [133] and used a contextual loss to transfer the style at the semantic level [134, 135].

Another line of work trains neural networks on unpaired examples for domain translation, including sim2real mappings. Early approaches used weight-sharing [136, 137] and sharing of specific content features [138, 139]. The cycle consistency in Cycle-GAN, which assumes bijective mapping between two domains, can map from zebra to horse [137, 140, 141], but bridging large deformations across domains, such as for going from cat to dog (see Fig. 4) requires alternative network architectures [142], or intermediate keypoint representations [143]; However, none of the methods discussed above establish a fine-grained, dense spatial correspondence between source and target, preventing the accurate transfer of desired keypoint locations.

One way to make sure image-to-image networks do not lose the keypoint correspondence is to handle the deformation explicitly. Explicit deformation has been used in diverse contexts. The spatial transformer network (STN) made affine and non-parametric spatial deformations popular as a differentiable network layer [144]. These approaches have been used to zoom in on salient objects [145], disentangle shape and appearance variations in an image collection [146], and register (brain scan) images to a standard, learned template image [147–150]. [151] introduced global transformation into the Cycle-GAN framework. While similar in spirit, additional advances beyond these approaches are still required to model deformations faithfully on our unpaired translation task.

Unsupervised 2D pose estimation Part 3

Siyuan Li, Semih Günel, Mirela Ostrek, Pavan Ramdya, Pascal Fua, Helge Rhodin; Deformation-aware Unpaired Image Translation for Pose Estimation on Laboratory Animals. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2020.

I worked on conceptualization, methodology (such as decoupling shape and texture information and explicit deformation of the shape), formal analysis, data curation, writing of original draft, writing of review, and editing.

3.1 Introduction

Deep learning-based pose estimation on images has evolved into a practical tool for a wide range of applications, as long as sufficiently large training databases are available. However, in very specialized domains there are rarely large annotation databases. For example, neuroscientists need to accurately capture the poses of all the appendages of fruit flies, as pose dynamics are crucial for drawing inferences about how neural populations coordinate animal behavior. Publicly available databases for such studies are rare and current annotation techniques available to create such a database are tedious and time consuming, even when semi-automated. Given the existence of motion simulators, an apparently simple workaround would be to synthesize images of flies in various poses and use these synthetic images for training purposes. Although image generation algorithms can now generate very convincing *deepfakes*, existing image translation algorithms do not preserve pose geometrically when the gap between a synthetic source and a real target is large. This is critical to our application, as creating matching high-fidelity images would be time consuming.

In this chapter, we introduce a novel approach to generate realistic images of different kinds of laboratory animals—flies, fish, and worms–from synthetic renderings for which labels such as keypoint annotations are readily available. The generated realistic images can then be used to train a deep network that operates on real images, as shown in Fig. 3. The challenge is to condition the generated images in such a way that the keypoints (e.g. skeleton joint positions) in the simulated source transfer to the realistic target; despite large differences in shape and pose as well as for small training sets that are practical, see Fig. 4.

We model the change of 2D pose and shape in terms of a deformation field. This field is then paired with an image-to-image translator that synthesizes appearance while preserving geometry, as shown in Fig. 5. Our approach is inspired by earlier approaches modeling human faces [146] and brain scans [147]. We go beyond these studies in two important ways. First, we introduce silhouettes as an intermediate representation that facilitates independent constraints (loss terms) on shape and appearance. It stabilizes training to succeed without reference images and helps to separate explicit geometric deformation from appearance changes. Furthermore, end-to-end training on unpaired examples is enabled with two discriminators and a straight-through estimator for non-differentiable thresholding operation and patch-wise processing. Second, to cope with large-scale as well as small-scale shape discrepancies, we introduce a hierarchical deformation

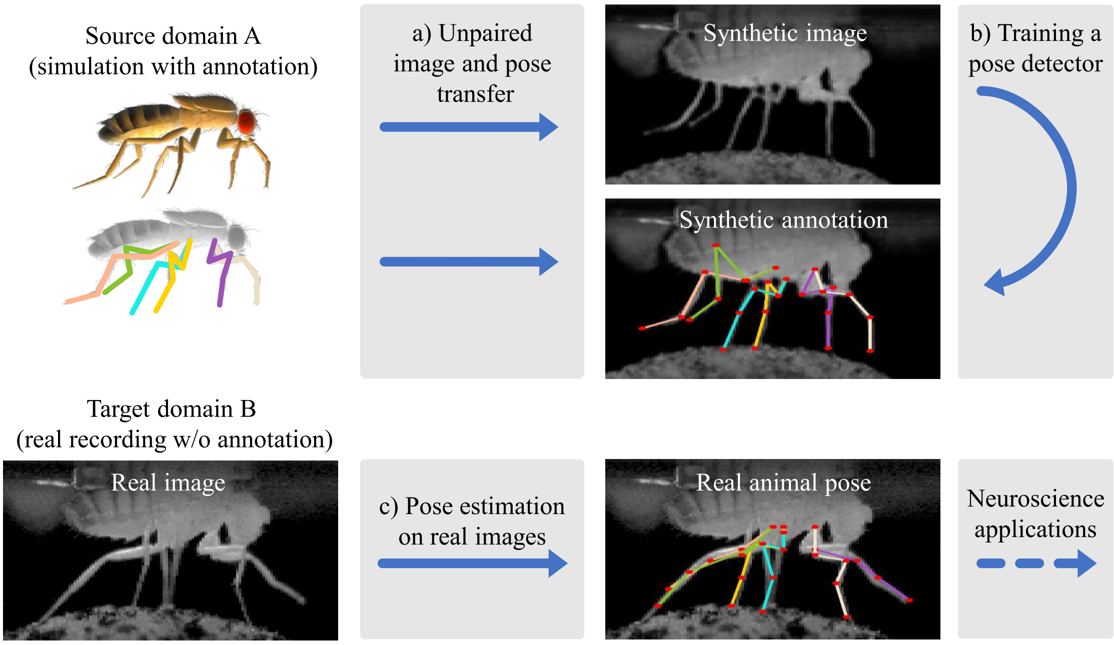

Source domain A
(simulation with annotation)
a) Unpaired image and pose transfer
Synthetic image
b) Training a pose detector
Synthetic annotation
Target domain B
(real recording w/o annotation)
Real image
c) Pose estimation on real images
Real animal pose
Neuroscience applications

Figure 3 – **Approach.** Our most morphologically complex example is the six-legged *Drosophila*: a) We transfer synthetic images and their keypoint annotations to realistically looking images using only unpaired examples of the two domains. b) Our method enables training of a pose detector that c) can be applied to real images for neuroscientific studies.

model to separate global scaling, translation, and rotation from local deformation.

We test our method on flies (*Drosophila melanogaster*), worms (*Caenorhabditis elegans*) and larval zebrafish (*Danio rerio*), see Fig. 4, and compare it against state-of-the-art approaches that rely either on circularity constraint or hand-defined factorizations of style and content. Not only does our method generate more realistic images, but more importantly, when we use the images it generates to train pose estimators we get more accurate results. Nothing in our approach is specific to the animals we worked with and that could also be applied just as well to limbed vertebrates, including rodents and primates.

3.2 Method

Our goal is to translate pose annotations and images from a synthetic domain A to a target domain B for which only unpaired images $\{\mathbf{I}_i^A\}_{i=1}^N$ and $\{\mathbf{I}_i^B\}_{i=1}^K$ exist. In our application scenario, the target examples are frames of video recordings of a living animal and the source domain are simple drawings or computer graphics renderings of a character animated with random deformations of the limbs. Both domains depict images of the same species, but in different pose, shape, and appearance.

Fig. 5 summarizes our approach. To tackle the problem of translating between domains while

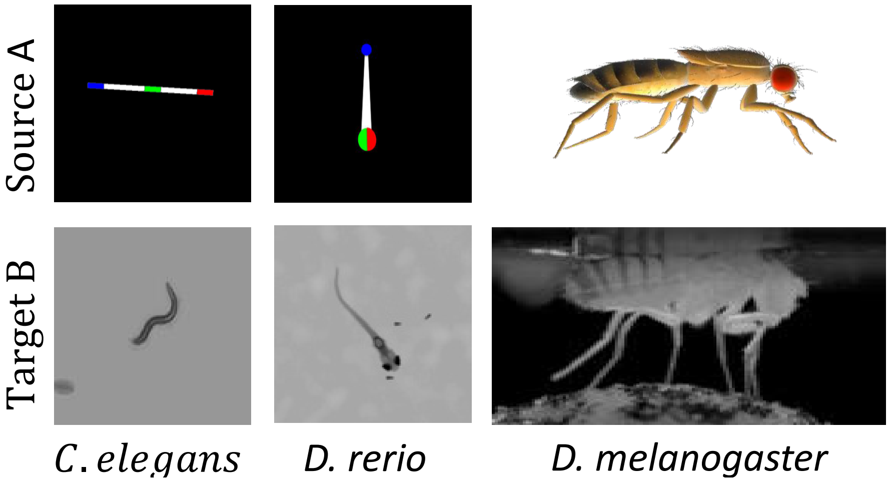


Figure 4 – **Domain examples with large discrepancy in appearance, shape and pose.** Translating from rendering to real images requires bridging the domain gap without having pixel nor pose correspondences. It is particularly challenging in our setting, as even the realistic fly character shows significant differences in shape (body and limb width) as well as pose (legs stretched).

preserving pose correspondence, we separately transfer spatially varying shape changes via explicit deformation of the source images via an intermediate silhouette representation $\hat{\mathbf{S}}^B$ (Section 3.2.1). Subsequently, we locally map from silhouette to real appearance (Section 3.2.2). The final goal is to train a pose estimator on realistic images from synthetic examples (Section 3.2.3). Our challenge then becomes to train neural networks for each, without requiring any paired examples or keypoint annotation on the target domain. To this end, we set up adversarial networks that discriminate differences with respect to the target domain statistics. Learning of the image translation is performed jointly on the objective

$$\mathscr{L} = \mathscr{L}_I + \mathscr{L}_S + \mathscr{R}_{\mathscr{D}}, \tag{1}$$

where $\mathscr{L}_I$ and $\mathscr{L}_S$ are the adversarial losses on generated segmentation and image, and $\mathscr{R}_D$ is a regularizer on the deformation grid. Besides images $\mathbf{I}$, our method operates on segmentation masks $\mathbf{S}$ of the same resolution. The domain origin is denoted with superscripts—$\mathbf{I}^A$, and the domain target (real images) is denoted $\mathbf{I}^B$. We use several generator and discriminator networks, which we denote G and D, respectively, with subscripts differentiating the type—G_I. We explain each step in the following section.

3.2.1 Spatial Deformation

Our experiments showed that using a single, large discriminator, as done by existing techniques, leads to overfitting and forces the generator to hallucinate, due to the limited and unrealistic pose variability of the simulated source. We model shape explicitly through the intermediate

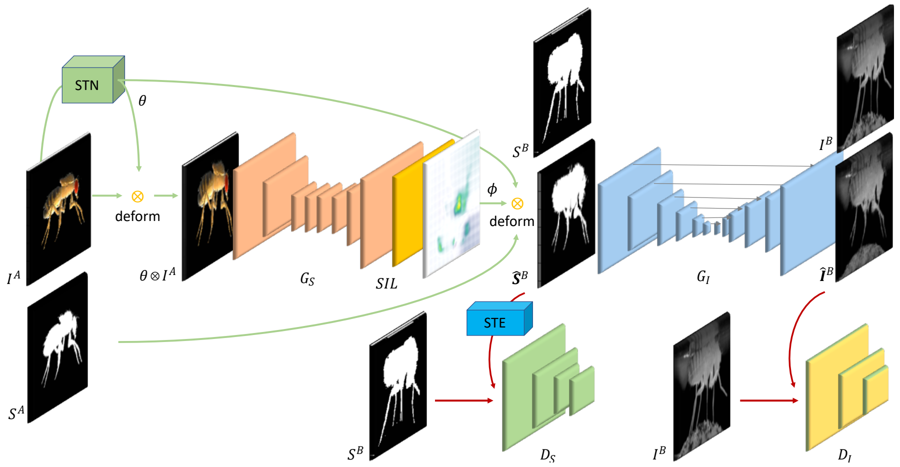


Figure 5 – **Overview of our deformation-based image translation method.** Our model has two steps. In the first step, the deformation from source domain A to target domain B is estimated for input image $\mathbf{I}^A$ and it's silhouette $\mathbf{S}^A$ via two networks G_D and STN (Spatial Transformer Network). Their output is an explicit deformation field parameterized by the global, affine transformation θ and a local, non-linear warping ϕ. Then, the deformed silhouette is transformed into the full output image $\hat{\mathbf{I}}^B$ with image generator G_I. Discriminators D_S and D_I enable unpaired training. D_S uses the Straight Through Estimator (STE) to backpropagate gradients through thresholding operations.

silhouette representation and its changes with a per-pixel deformation field, as shown in Fig. 6. The silhouette lets us setup independent discriminators with varying receptive field; large for capturing global shape and small to fill-in texture. Moreover, the deformation field enables the desired pose transfer while bridging large shape discrepancies.

The first stage is a generator G_S that takes a synthetic image $\mathbf{I}^A$ and mask $\mathbf{S}^A$ as input, and outputs a deformed segmentation mask $\hat{\mathbf{S}}^B$ that is similar to the shapes in B. To model global deformation, we use a spatial transformer network (STN) [144] that takes the synthetic image $\mathbf{I}^A \in \mathbb{R}^{C,H,W}$ as input, and outputs an affine matrix $\theta \in \mathbb{R}^{3,4}$, which models global scaling, translation and rotation differences between the source and target domains. It is trained jointly with a fully-convolutional generator network, G_D, which takes the globally transformed image as input and outputs $\phi \in \mathbb{R}^{2,H,W}$, a per-pixel vector field that models fine-grained differences in pose and shape. The vector at pixel location x in ϕ points to the pixel in the source domain that corresponds to x. Overlaying the source pixels of selected rows and columns of ϕ leads to the deformed grid visualized in Fig. 6. This hierarchical representation allows us to cope with varying degrees of discrepancies between the two domains. We refer to the combined application of these two networks and their output as generator $G_S(\mathbf{I}^A, \mathbf{S}^A) = \phi \otimes \theta \otimes \mathbf{S}^A$, where $\theta = STN(\mathbf{I}^A)$, $\phi = G_D(\theta \otimes \mathbf{I}^A)$, and $\otimes$ denotes the transformation by global and local deformation.

Training G_S requires silhouettes in A and B. Silhouettes $\mathbf{S}^A$ in the source domain are trivially obtainable from synthetic characters by rendering them on a black background. It is relatively

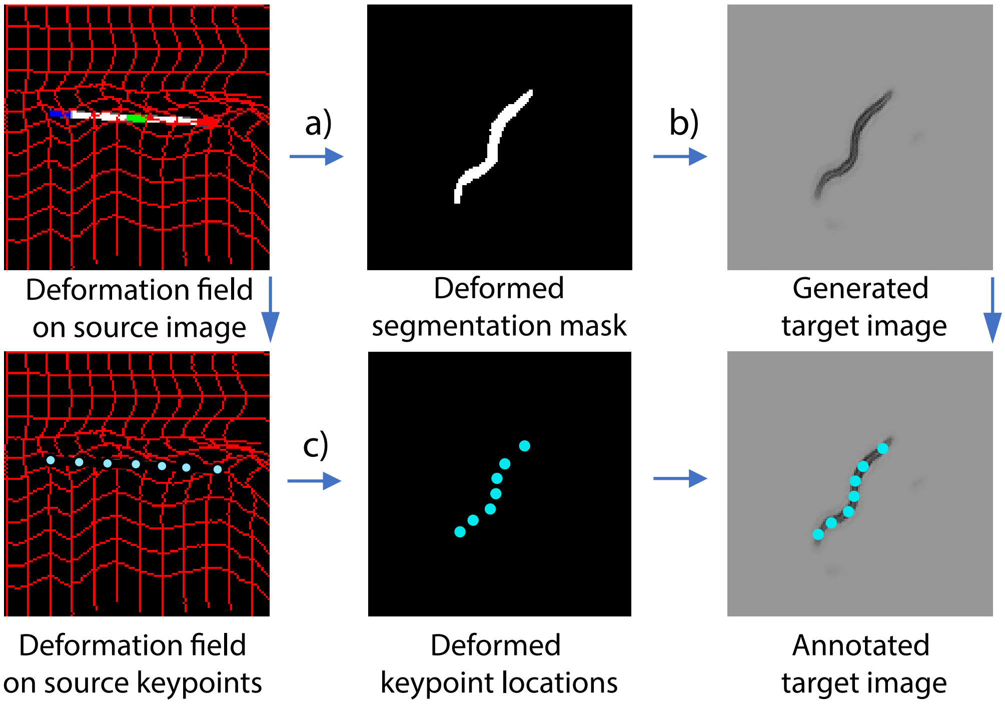

a)
b)
c)
Deformation field on source image
Deformed segmentation mask
Generated target image
Deformation field on source keypoints
Deformed keypoint locations
Annotated target image

Figure 6 – **Explicit deformation ensures transfer of keypoints.** The deformation field is inferred as part of a) source image segmentation to target image segmentation transfer (including global, affine transformation) and b) segmentation to target image translation. c) The same deformation field is applied to transfer known keypoints from source to target.

easy to estimate $\mathbf{S}^B$ on a static background for the target domain as datasets are obtained in controlled lab environments. We will later demonstrate that our model is robust to remaining errors in segmentation.

The difficulty of our task is that all domain examples are unpaired, hence, a constraint can only be set up in the distributional sense. Thus, we train a shape discriminator D_S alongside G_S and train them alternately to minimize and maximize the adversarial loss

$$\begin{aligned} \mathscr{L}_S &= \mathscr{L}_{GAN}(G_{\mathbf{S}}, D_{\mathbf{S}}, \mathbf{S}^A, \mathbf{S}^B) \\ &= \mathbb{E}_{\mathbf{S}^B}[\log D_S(\mathbf{S}^B)] + \mathbb{E}_{\mathbf{S}^A}[\log(1 - D_S(G_S(\mathbf{S}^A)))], \end{aligned} \tag{2}$$

where the expectation is built across the training set of A and B. The adversarial loss is paired with the regularizer

$$\mathscr{R}_D = \alpha(\left\| \nabla\phi_x(A) \right\|^2 + \left\| \nabla\phi_y(A)) \right\|^2) + \beta \left\| \phi(A) \right\|, \tag{3}$$

to encourage smoothness by penalizing deformation magnitude and the gradients of the deformation field, as in [146].

The inputs of the discriminator are binary masks from source domain A and target domain B. However, the deformed masks are no longer binary on the boundary because of the interpolation required for differentiation. Thus, it would be trivial for D_S to discriminate against the real and synthesized masks based on non-binary values. To overcome this issue, we threshold to get a binary mask. Although the threshold operation is not differentiable, we can still estimate the gradients with respect to G_S using a straight through estimator (STE) [152], which treats the threshold as the identity function during backpropagation, and therefore passes the gradients on to the previous layer.

Implementation details. Directly outputting a vector field leads to foldovers that make the training unstable. Instead, we parameterize it as the gradient of the deformation field ϕ, and enforce positivity to prevent foldovers as in [146]. ϕ can be recovered by summing the gradients across the image. The deformation from A to B is implemented with a spatial transformer layer (STL) that infers the value of deformed pixel locations by bilinear interpolation [144] and is differentiable. In contrast to [146], we use a fully convolutional network to learn the local deformation field. The G_D network consists of 3 Resnet blocks between downsampling/upsampling layers. The receptive field of the network is 64 pixels, 1/2 of the image, which is sufficient for our experiments.

The STN network consists of 5 convolutional layers and a fully connected stub to output θ that is preceded by max-pooling and SELU units (this yielded better results in preliminary experiments, compared to ReLU activations).

3.2.2 Appearance Transfer

Once the shape discrepancies between the two domains have been estimated and corrected by G_S, we then generate the appearance of the target domain on the deformed silhouettes $\hat{\mathbf{S}}^B = G_S(\mathbf{I}^A, \mathbf{S}^A)$. We deploy a generator G_I that is configured to preserve the source shape, only filling in texture details. The input is $\hat{\mathbf{S}}^B$ and the output is a realistic image $\hat{\mathbf{I}}^B$ that matches the appearance of the target domain. We use a discriminator D_I for training, as synthetic and real images are unpaired. In addition, our choice of using the silhouette as an intermediate representation allows us to introduce a supervised loss on $\mathbf{S}(\mathbf{I}^B)$ computed from real images $\mathbf{I}^B$. The training objective is

$$\mathscr{L}_I = \mathscr{L}_{GAN}(G_I, D_I, \mathbf{I}^A, \mathbf{I}^B) + \left\| G_I(\mathbf{S}(\mathbf{I}^B)) - \mathbf{I}^B \right\|, \tag{4}$$

where the GAN loss is defined as before and the second part is the supervised loss which stabilizes training.

Training the supervised loss in isolation without end-to-end training with the adversarial losses leads to artifacts since neither the synthesized nor silhouettes from real images are perfect, see Fig. 7.

The pose distribution of the simulated character can differ even after local and global deformation

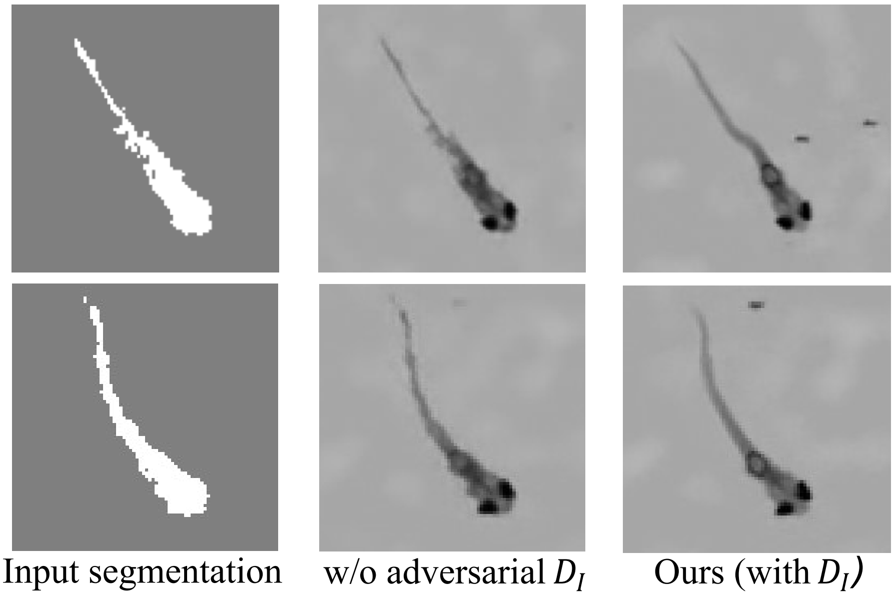


Figure 7 – **Texture discriminator influence.** Without the adversarial discriminator, the image generator is disturbed by an irregular silhouette boundary. In our model, the adversarial D_I creates a link to the deformed silhouettes $\hat{S}^B$ enabling end-to-end training.

as some pose differences cannot be explained by an image deformation. For instance, the occlusion effects of crossing legs on *Drosophila* cannot be undone as a 2D image transformation. A discriminator with a large receptive field could detect these differences and re-position legs at locations without correspondence in the source. To counteract this issue, we make sure D_I has a small receptive field. This is possible without suffering from texture artifacts since the global shape deformation is already compensated by G_S and the texture can be filled in locally.

Implementation details. We use a 7-layer U-Net generator as our backbone network for image translation with G_I. The skip connections in the U-Net help the network preserve the spatial information. For D_I, we use a patch-wise discriminator, consisting of three 4x4 convolutional layers; the first one with stride two and the second one with instance normalization. All activation functions are leaky ReLU. The small receptive field of the patch discriminator additionally helps to maintain the spatial structure of the object and was sufficient in our experiment to reproduce the real appearances faithfully.

3.2.3 Pose Estimation

We use the stacked hourglass network architecture for pose estimation [9]. Stacked hourglass is a fully-convolutional network with several bottlenecks that takes an image **I** and outputs a heatmap **H** of the same aspect ratio but at four times lower resolution due to pooling at the initial layers. The heatmaps **H** are a stack of 2D probability maps with Gaussian distribution, where the maximum value of each channel in the stack indicates one specific joint location. Because our source images are synthesized from 3D character models, we can use the virtual camera matrix to project 3D keypoints, such as the knee joint, onto the image.

To obtain annotations in the target domain, we conveniently use the image deformation operation $\mathbf{H}_d = (\phi, \theta) \otimes \mathbf{H}$ to compute the deformed heatmap $\mathbf{H}_d$ that matches to the synthesized target domain image $\mathbf{I}_d = G_I(G_S(\mathbf{I}^A, \mathbf{S}^A) = G_I((\phi, \theta) \otimes \mathbf{I}^A)$, with ϕ coming from G_D and θ from the STN in G_S. Note, that this is only possible due to the explicit handling of deformations.

Having synthesized realistic examples of the target domain and transferred ground truth heatmaps, it remains to train the pose estimation network in a supervised manner. We use the L_2 loss between the predicted and ground truth heatmaps. At test time, we estimate the corresponding joint location as the argmax of the predicted heatmap, as usual in the pose estimation literature. Because the worm is tail-head symmetric, we compute errors for front-to-back and back-to-front ordering of joints and return the minimum at training and test time. We call this a permutation invariant (PI) training and testing. In the same vein, we regard the correct assignment of the six *Drosophila* legs as an independent task that is extremely hard to solve in the 2D domain. To separate and sidestep this problem, we compute the test error for all possible permutations and return the minimum.

Implementation details. Input images are augmented by random rotations, drawn uniformly from $[-30°, 30°]$. Additional details are given in the supplemental document.

3.2.4 Implementation details

Deformation representation. Directly modeling the deformation as vector field will make the transformation unstable and easily lose the semantic correspondence. For example, a vector field permits coordinate crossing and disconnected areas, which leads to unstable training and divergence. In order to preserve a connected grid topology, we model our deformation close to the difformorphic transformation, which generates the deformation field as the integral of a velocity field. This leads to useful properties such as invertibility and none crossing intersections [153]. However, it is in general expensive to compute the integral over an axis, thus making it difficult to incorporate into deep networks. Instead of modeling a continuous velocity function, we directly model our deformation field ϕ as the integral of the spatial gradient of vector field, as proposed by Shu et al. [146]. We write,

$$\triangledown \phi_x = \frac{\partial \phi}{\partial x} \qquad \triangledown \phi_y = \frac{\partial \phi}{\partial y} \tag{5}$$

where x, y define the gradient directions along the image axes. The ϕ_x and ϕ_y measure the difference of consecutive pixels. By enforcing the difference to be positive (e.g., by using ReLU activation functions; we use HardTanh with range (0, 0.1)), we avoid self-crossing and unwanted disconnected areas. For example, when ϕ_x and ϕ_y equals to 1, the distance between the consecutive pixels is the the same. If $\phi_x, \phi_y > 1$, the distance will increase, otherwise, when ϕ_x ,$\phi_y < 1$, it will decrease.

The second module is the spatial integral layer, also the last layer of deformation spatial gradient generator. This layer sums the spatial gradients along the x and y directions and produces the final deformation field,

$$\phi_{i,j} = (\sum_{m=0}^{i} \nabla\phi_{x_m}, \sum_{n=0}^{j} \nabla\phi_{y_n}), \tag{6}$$

where i, j is the pixel location. Since the u, v in general position do not correspond to one exact pixel location in the source image, we compute the output image using a differentiable bilinear interpolation operation, as for spatial transformers [144].

Shape Discriminator We utilize the 70 × 70 patchGAN discriminator as our backbone structure [125] . The patch-wise design makes the network focus on the local area of the shape. Furthermore, if the shape between two domains is extremely different, the patch-wise design prevents the discriminator from converging too quickly. However, the design also limits the network's awareness of global shape changes [142]. Thus, we add dilation to the second and the third convolution layers of patchGAN. Those dilated layers enlarge the receptive field of our shape discriminator, making it aware of bigger shape variation, giving a better guidance to the generator.

Image Generator. We build our generator on the U-Net architecture, which is proved to be effective in tasks such as pixel-wise image translation and segmentation [154]. The generator contains several fully convolutional down-sampling and up-sampling layers. The skip connections in the generator help to propagate information directly from input features to the output, which guarantee the preservation of spatial information in the output image.

Pose Estimator. We adopt the stacked hourglass human pose estimation network to perform pose estimation on animals [9]. The stacked hourglass network contains several repeated bottom-up, top-down processing modules with intermediate supervision between them. A single stack hourglass module consists of several residual bottleneck layers with max-pooling, following by the up-sampling layers and skip connections. We used 2 hourglass modules in our experiments. The pose estimation network is trained purely on the animal data we generated; without pre-training and manually annotated labels. The ground-truth poses come from the annotations of synthetic animal models. The pose invariant (PI) training is performed in all experiments labeled with *PI training*.

Pose annotation.*Drosophila* has six limbs, each limb has five joints, giving 30 2D keypoints that we aim to detect. By using our image translation model, we generated 1500 images with annotation from the synthetic data. Each image is in size 128 × 128 pixels. The first hourglass network is preceded with convolutional layers that reduce the input image size from 128 × 128 to 32 × 32. The second hourglass does not change the dimension. Thus, the network will output a $30 \times 32 \times 32$ tensor, which represents the probability maps of 30 different joints locations. For training, we create the ground truth label using a 2D Gaussian with mean at the annotated keypoint and 0.5 on the diagonal of the covariance matrix. The training loss is the MSE between the generated probability map and the ground truth label.

We annotated three keypoints on D. rerio and seven keypoints on C. elegans. We use the same network as for Drosophila, but the output tensor adapted to the number of keypoints, $3 \times 32 \times 32$ and $7 \times 32 \times 32$, respectively.

Training The Unpaired Image Translation Network. We use the Adam optimizer with different initial learning rates for different modules. For G_I, D_I, we set the learning rate to 2e−3. For G_S, we set the learning rate to 2e−5 since a slight update will have a big impact on the deformation field due to the integrating the spatial gradient in the last layer of G_S. We set the learning rate of D_S to $1/10$ the one of G_S, which balanced the influence of G_S and D_S in our experiments. In case of *Drosophila* training, we apply linear decay to our learning rates. We start the decay of G_S, D_S at epoch 50 and reduce it to 0 till epoch 100. For fish and worm, we set the learning rate of G_D to $1e-4$ and D_S to $1e-5$, to account for the simpler setting of deforming from a single template image. Moreover, we linearly decay from epoch 100 to epoch 200.

The batch size of the image translation training is set to 4. An other important detail is the initialization of G_S to generate the identity mapping. We achieved that by initially training G_S solely on the regularization term, which pushes it towards this state.

Training Pose Estimation Network. We use Adam optimizer with initial learning rate of 2e−3. We train the pose estimation network for 200 epochs and the learning rate starts to linear decay after epoch 100, till epoch 200.

3.3 Evaluation

In this section, we qualitatively compare our results to canonical baselines and variants of our algorithm, in order to highlight advantages and remaining shortcomings both visually and quantitatively. This includes the task of 2D keypoint localization on the target domain. We test our approach on different neuroscience model organisms in order to demonstrate varying complexity levels of deformation and generality to different conditions. Additional qualitative results and comparisons are given in the supplemental document.

All input and output images are of dimension $(128, 128)$. We operate on gray-scale images, i.e. channel dimension $C = 1$, obtained from infrared cameras, which are commonly used in neuroscience experiments in order to avoid inadvertent visual stimulation. Nevertheless, our method extends naturally to color images.

Datasets. We test on available zebrafish and worm image datasets, by [123] and [155, 156], using 500 and 100 real images for unpaired training. To quantify pose estimation accuracy, we manually annotate a test set of 200 frames with three keypoints (tail and eyes) for the zebrafish and two points (head and tail) for the worm. In these datasets, the background is monochrome and is removed by color keying to obtain the foreground masks. Because of the simplicity of these models, we use a simple, static stick figure as a source image that is augmented by uniformly random rotation and translation. Fig. 8 gives example images.

Our most challenging test case is the *Drosophila* fly. We use the subset of the dataset published alongside [111], which contains transitions between different speeds of walking, grooming and standing captured from a side view and includes annotations for five keypoints for each of the fully-visible legs (four joints and tarsus tip). In this dataset, the fly is tethered to a metal stage of a microscope and the body remains stationary, yet the fly can walk on a freely rotating ball (spherical treadmill), see Fig. 8. To get the target domain segmentation masks, we first crop out the ball and background clutter with a single coarse segmentation mask. This mask is applied to all images due to the static camera setup. The body, including the legs, is then segmented by color keying on the remaining black background. Please note, that at test time, no manual segmentation is used.

We use 815 real images for unpaired training and 200 manually annotated images for testing. On the source side, we render 1500 synthetic images using an off-the-shelf Maya model from turbosquid.com. The source motion is a single robotic walk cycle from [157] which we augment by adding random Gaussian noise to the character control handles. This increases diversity but may lead to unrealistic poses that our deformation network helps to correct.

Metrics. The pose estimation accuracy is estimated as the root mean squared error (RSME) of predicted and ground truth 2D location and percentage of correct keypoints (PCK), the ratio of predicted keypoints below a set threshold. We report results for thresholds ranging from 2 to 45 pixels. We also provide accumulated error histograms and the average PCK difference as the area under the curve (AUC) of the error histogram, to analyze the consistency of the improvements.

In many cases, it is impossible, even for a human, to uniquely identify the leg identity for *Drosophila*. As in [111], we therefore only evaluate the three entirely visible legs. Moreover, we find the optimal leg assignment across the three legs at test time as PI-RSME, PI-PCK, and PI-AUC, using the permutation invariant metric introduced in Section 3.2.3. Because the images of the worm are front-back symmetric, we train and test by permuting keypoints front-to-back and back-to-front. The pose estimation task lets us quantify the made improvements, both due to more realistically generated images (image quality), as well as the preservation of correspondences (geometric accuracy) since the lack of one would already lead to poor pose estimation.

To independently quantify the image quality, we use the structural similarity (SSIM) index [158]. We measure the similarity between all generated images $\hat{\mathbf{I}}^B$ (for every $\mathbf{I}^A$ in A) with a pseudo-randomly sampled reference image $\mathbf{I}^B$.

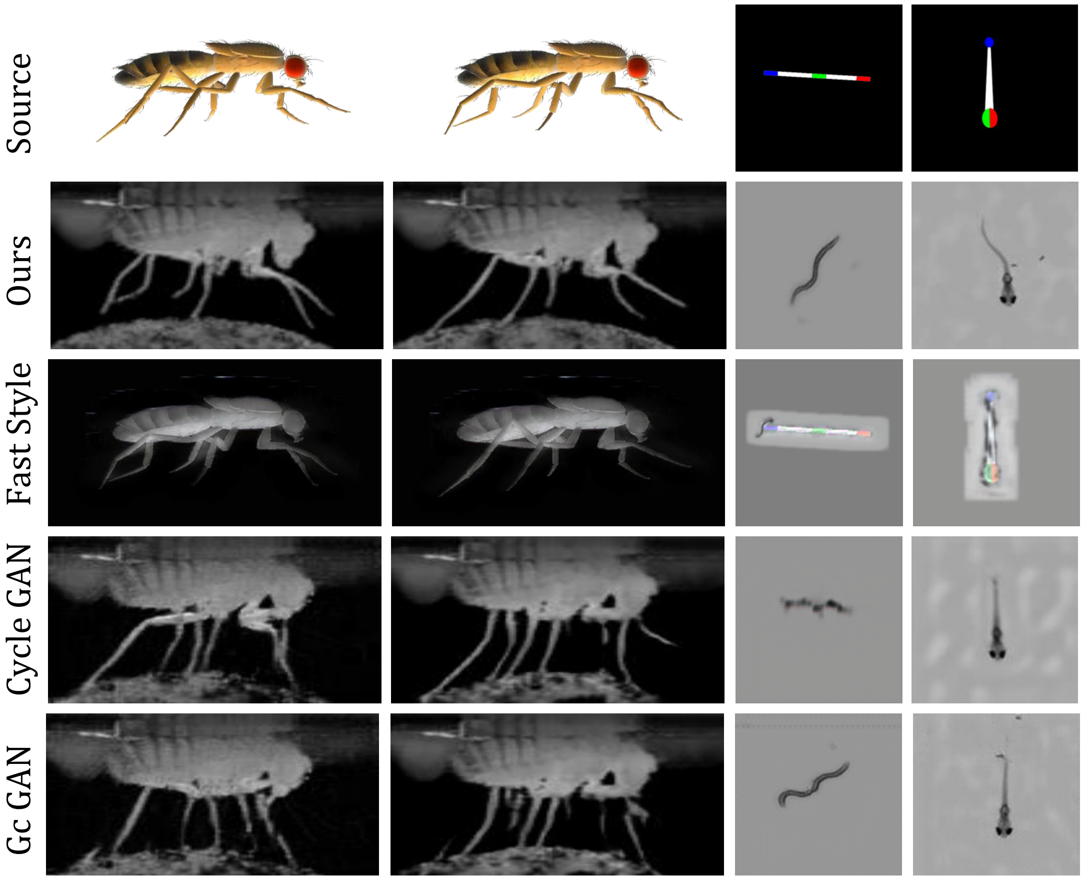


Figure 8 – **Qualitative comparison.** Existing unpaired image translation methods can generate realistic images on worm and fish, but exhibit artifacts for the thin legs of the *Drosophila* and zebrafish examples. Ours succeeds on all three classes.

Baselines. We compare to Fast-Style-Transfer [159], which combines [129, 131, 132], Cycle-GAN [137] and Gc-GAN [151]. With the latter being a state-of-the-art method for image to image translation and the former used to validate that simpler solutions do not succeed. We compare pose estimation with the same architecture, trained directly on the synthetic images, images generated by the above mentioned methods, and on manual annotations of real training images (185 for Drosophila, 100 for worm, and 100 for fish).

3.3.1 Quality of Unpaired Image Translation

The quality of Cycle and Gc-GAN is comparable to ours on the simple worm and fish domains, as reflected visually in Fig. 8 and quantitatively in terms of SSIM in Table 1. For *Drosophila*, our method improves image quality (0.66 vs. 0.39, 0.63 and 0.65). Albeit the core of explicit deformation was to transfer pose annotations across domains, this analysis shows that an explicit mapping and incorporation of silhouettes regularizes and leads to improved results. For instance,

Task	*D.M.*	*C.E.*	*D.E.*
Fast-Style-Transfer	0.3932	0.0539	0.6385
Cycle-GAN	0.6543	0.9034	0.8504
Gc-GAN	0.6392	0.8915	0.8586
Ours	**0.6619**	**0.9143**	**0.8847**

Table 1 – **Structured similarity (SSIM) comparison**. The explicit modeling of deformation outperforms baselines, particularly on the complex *Drosophila* images showing complex poses.

it ensures that thin legs of the fly are completely reconstructed and that exactly six legs are synthesized, while Cycle-GAN and Gc-GAN hallucinate additional partial limbs.

3.3.2 Pose Domain Transformation

Fig. 9 shows that our method faithfully transfers 2D keypoints, obtained for free on synthetic characters, to the target domain. The transferred head and tail keypoints on the worm and fish correspond precisely to the respective locations in the synthesized images, despite having a different position and constellation in the source. This transfer works equally well for the more complex *Drosophila* case. Only occasional failures happen, such as when a leg is behind or in front of the torso, rendering it invisible in the silhouette. Moreover, the eyes of the fish are not well represented in the silhouette and therefore sometimes missed by our silhouette deformation approach.

By contrast, existing solutions capture the shape shift between the two domains, but only implicitly, thereby loosing the correspondence. Poses that are transferred one-to-one from the source do no longer match with the keypoint location in the image. Keypoints are shifted outside of the body, see last column of Fig. 9. The style transfer maintains the pose of the source, however, an appearance domain mismatch remains. We show in the next section that all of the above artifacts lead to reduced accuracy on the downstream task of pose estimation.

3.3.3 2D Pose estimation

The primary objective of this study is to demonstrate accurate keypoint detection on a target domain for which only annotations on synthetic images with different shape and pose exist. Fig. 11 shows qualitative results. We compare the performance of the same keypoint detector trained on images and keypoints generated by ours and the baseline methods. The absolute errors (tables 2 and 3) and accumulated error histograms (Fig. 10) show significant (PCK 15: 83.2 vs. 72.9 Cycle-GAN) and persistent (AUC 85.1 vs 78.4) improvements for Drosophila and the other domains. A similar improvement is visible for the simpler worm and zebrafish datasets, with even higher gains of up to 13 PCK points. Although there remains a gap compared to training on real images with manual labels for small error thresholds, our method comes already close to

	Drosophila melanogaster			
Metric	PI-PCK ↑ (5 pix)	PI-PCK ↑ (15 pix)	PI-AUC ↑ (4-45 pix)	PI-RMSE ↓ (pix)
Synthetic	19.8	67.9	75.75	13.456
Fast-Style-Transfer	15.4	57.6	68.9	17.309
Gc-GAN	11.9	68.7	76.3	13.175
Cycle-GAN	15.0	72.9	78.4	12.302
Ours	**38.6**	**83.2**	**85.1**	**9.289**
Supervised	72.2	88.8	90.35	6.507

Table 2 – **Pose estimation accuracy comparison on *Drosophila melanogaster*.** A similar improvement as for Drosophila is attained on the other tested laboratory animals, with a particularly big improvements on the zebrafish. Pose-invariant training improves results.

	Caenorhabditis elegans			*Danio rerio*		
Metric	PI-PCK ↑ (5 pix)	PI-AUC ↑ (2-20 pix)	PI-RMSE ↓ (pix)	PCK ↑ (10 pix)	AUC ↑ (2-20 pix)	RMSE ↓ (pix)
Synthetic	0.0	0.9	67.29	29.3	37.4	20.15
Fast-Style-Transfer	3.1	25.0	20.50	15.6	20.8	19.25
Gc-GAN	9.7	25.0	27.38	68.2	54.5	27.38
Cycle-GAN	45.3	63.2	14.71	68.7	59.1	9.70
Ours	**64.0**	**70.9**	**11.17**	**85.0**	**72.1**	**7.23**
Supervised	93.1	91.3	4.15	97.6	84.9	4.35

Table 3 – **Pose estimation accuracy on *C. elegans* and *D. rerio*.** Our method significantly outperforms all baselines and approaches the supervised baseline. Units are given in round brackets.

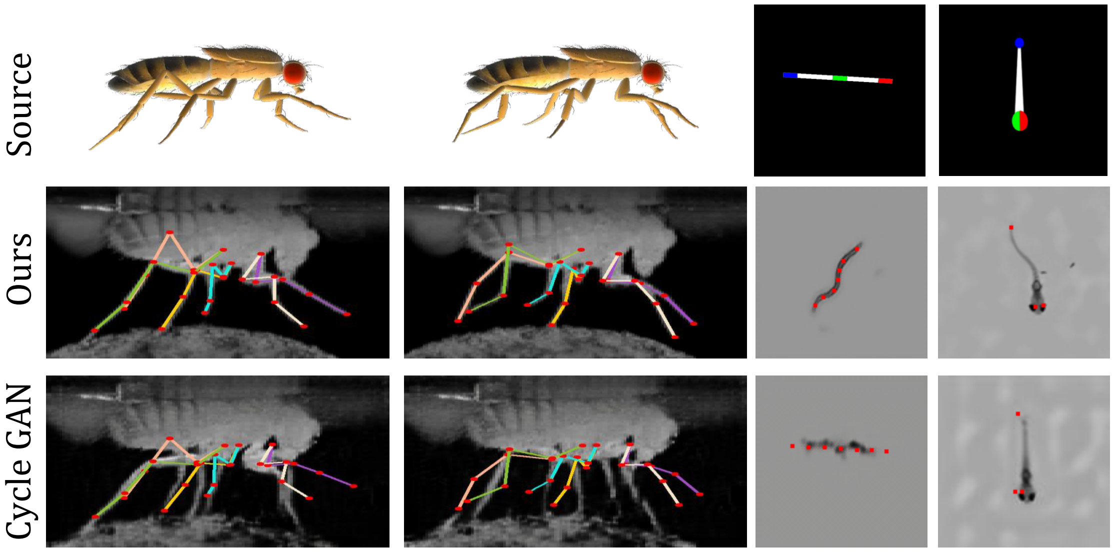


Figure 9 – **Automatic Pose Annotation.** Our method faithfully transfers poses across domains, while Cycle-GAN, the best performing baseline, loses correspondence on all three datasets.

the supervised reference method in PCK 15 and above. Compared to existing unpaired image translation methods, we gain a large margin.

The effect of the introduced explicit and hierarchical deformation model and the two-stage shape-separating training is analyzed independently in the following section.

3.3.4 Ablation study

We compared our full model at PCK-15 (83.2), to not using one of our core contributions: no deformation (64.9), only global affine (57.3), only local non-linear (79.3), and directly encoding a vector field (69.0). The numbers and Fig. 12 shows that all contributions are important. Also end-to-end training with D_I is important, as shown in Fig. 7, and by additional examples in the supplemental document.

The additional metrics in Table 4 show that our contributions improve consistently across different PCK thresholds.

Each of our contributions is significant with gains of 8 to 30 on PCK-15 and 4 to 14 AUC points. Notably, using global affine deformation is worse than without any deformation. This may be because the affine network rotates the body of synthetic fly to match the shape of real fly. However, the rotation also affects the leg orientation, which leads to less realistic poses. It is best to use global and local motion together (Ours).

Moreover, Fig. 13 provides additional results comparing the generated image quality with and without using D_I. Clear improvements are gained for the fish and Drosophila. For instance,

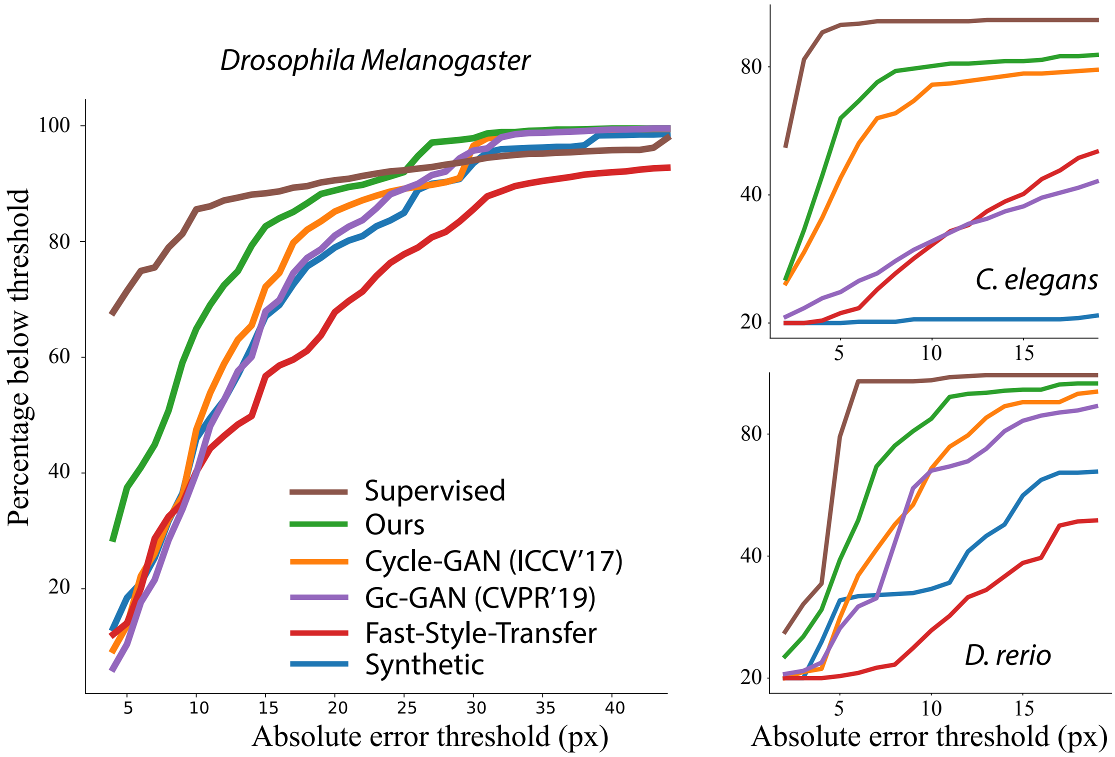


Figure 10 – **Pose estimation accuracy.** The accumulated error curves show the accuracy (vertical axis) for different PCK thresholds (horizontal axis). Our method clearly outperforms the baselines and approaches the manually supervised reference.

legs are properly superimposed on the ball, while holes arise without D_I (therefore, without end-to-end training). No significant improvement could be observed on the worm case due to its simplicity.

Fig. 14 provides additional examples of the image generation quality and the accuracy of the associated keypoint annotations, inferred via our explicit deformation field.

Moreover, Fig. 15 shows additional examples of the pose estimation quality compared to using Cycle-GAN. Our approach produces much fewer miss classifications, for instance, in the case of extreme bending positions of the worm.

3.4 Conclusion

In this chapter, we have presented an approach for translating synthetic images to a real domain via explicit shape and pose deformation that consistently outperforms existing image translation methods. Our method allows us to train a pose estimator on synthetic images that generalize to real ones; without requiring manual keypoint labels. One of our test cases is on *Drosophila* tethered to a microscope used to measure neural activity. By combining our improvements on pose

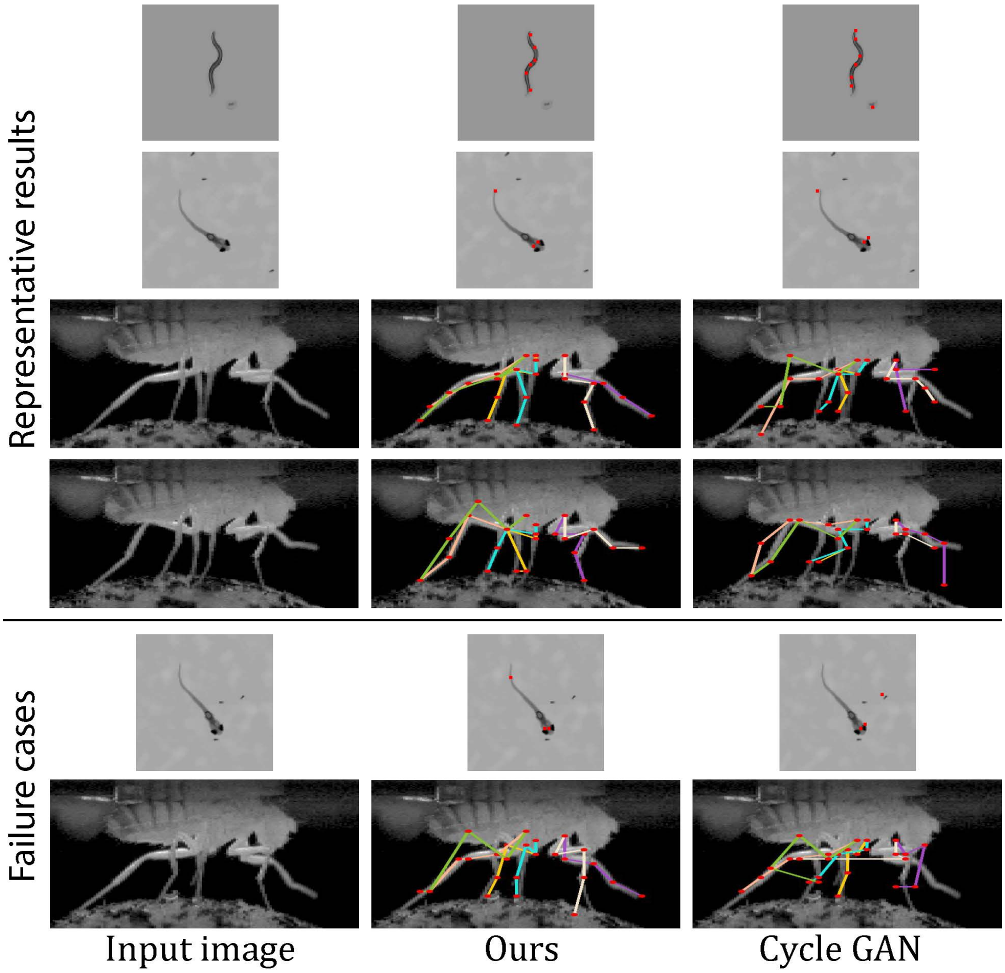


Figure 11 – **Qualitative pose estimation results.** The estimator provides decent results across all three animals. Occasional failures (last two rows) happen when legs cross, at occlusions, and for the fine fish tail. Training on Cycle-GAN images does not succeed.

estimation with state-of-the-art microscopy, we anticipate more rapid advances in understanding the relationship between animal behaviour and neural activity.

For some domains the assumption of a target segmentation mask is constraining. For instance, our method is not applicable for transferring synthetic humans to real images on cluttered backgrounds. In the future, we plan on integrating unsupervised segmentation, as demonstrated by [160] for single-domain image generation.

Although we could synthesize a variety of poses for the worm and fish using a single stylized source image, our method was not able to synthesize entirely new *Drosophila* poses, because crossing legs could not be modeled using a 2D image deformation. Therefore, sufficiently varied examples were needed in the source domain. Moreover, symmetries and self-similarities can lead to flipped limb identities. This remains a hard, open problem that we do not address in this

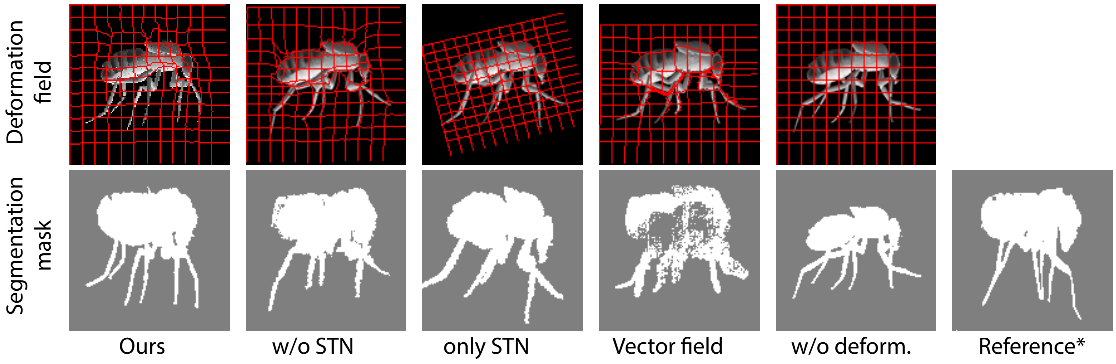


Figure 12 – **Ablation study.** All our contributions are important, removing the global spatial transformer leads to lower local details (bends legs), only global transformation does not correct shape and pose differences (thinner, straight legs), and predicting the vector field (not its derivative) produces foldovers. *silhouette estimated from an unpaired target domain image.

Metric	Batch size	PI-PCK ↑ (5 pix)	PI-PCK ↑ (15 pix)	PI-AUC ↑ (4-45 pix)
Ours	128	28.0	74.4	80.6
Ours	12	**38.6**	**83.2**	**85.1**
Ours w/o global deformation	12	31.4	79.2	84.0
Ours w/o deformation	12	18.5	64.9	74.9
Ours w/o local deformation	12	13.1	57.4	73.8
Ours using vector field	12	18.6	69.1	79.0

Table 4 – **Detailed ablation study on *Drosophila Melanogaimster*.** All model components contribute to the final reconstruction accuracy. Surprisingly, a smaller batch size improved results.

study. Nevertheless, we believe that this is an important first step and that temporal cues and multi-view can be used to find a consistent and correct assignment in the future, following ideas used in [105] to perform pose estimation of humans and monkeys.

The attained accuracy is not yet perfect (see bottom of Fig. 11) and can only be used to classify gross behaviours. Additional advances are needed to obtain pixel accurate reconstructions which would enable analysis of detailed movements, such as the activation of individual muscles.

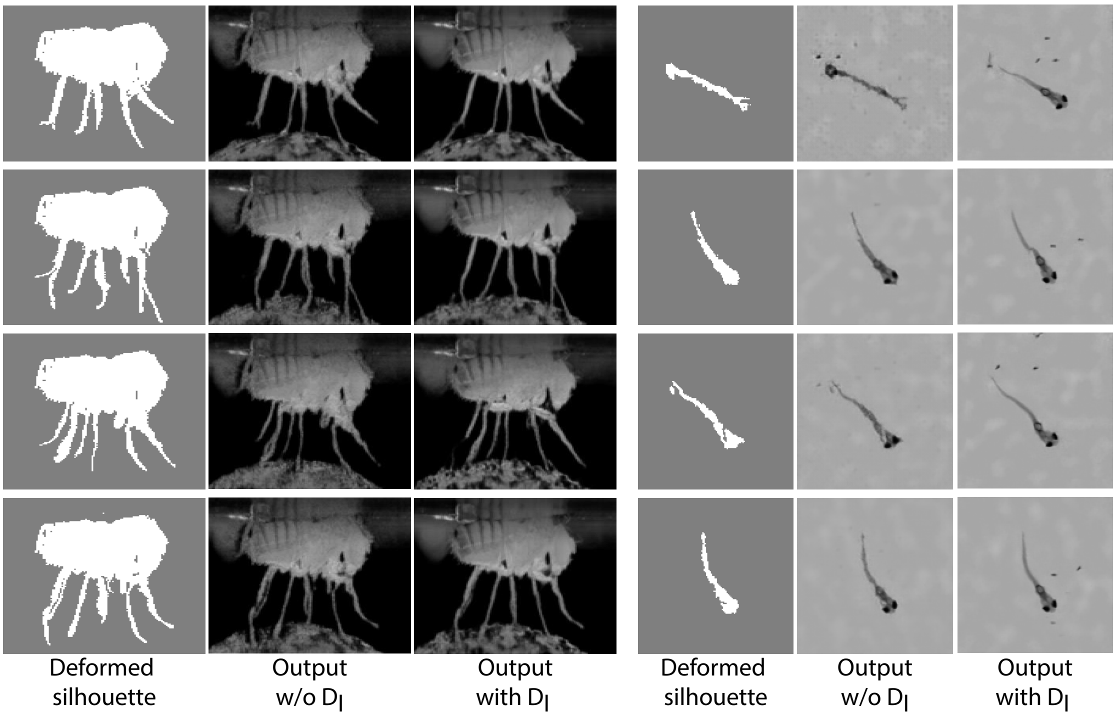


Figure 13 – **Ablation study on D_I.** Without D_I, small artifacts in the generated (deformed) segmentation masks lead to unrealistic images.

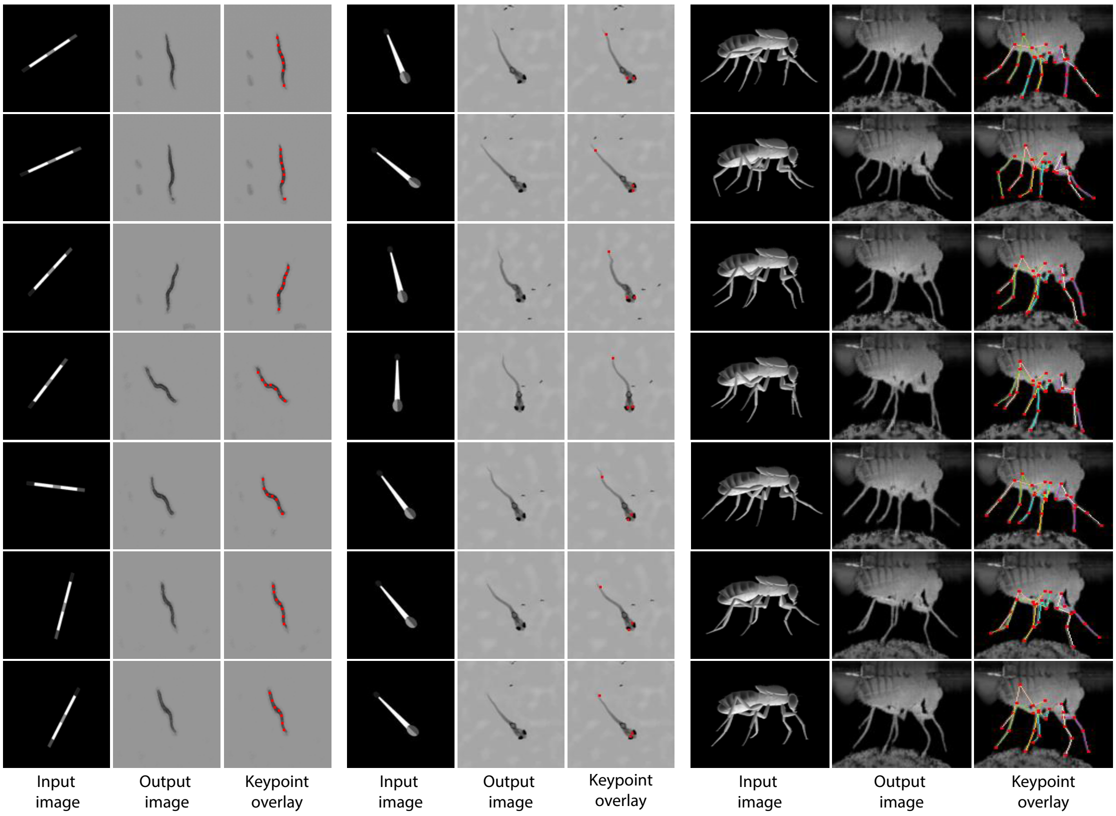


Figure 14 – **Qualitative image generation results.** Our approach can generate realistic and diverse poses, which are transferred across domains faithfully. Our method works on all three tested animals, including the *Drosophila* dataset with superimposed legs that are on the ball that has no correspondence in the source domain.

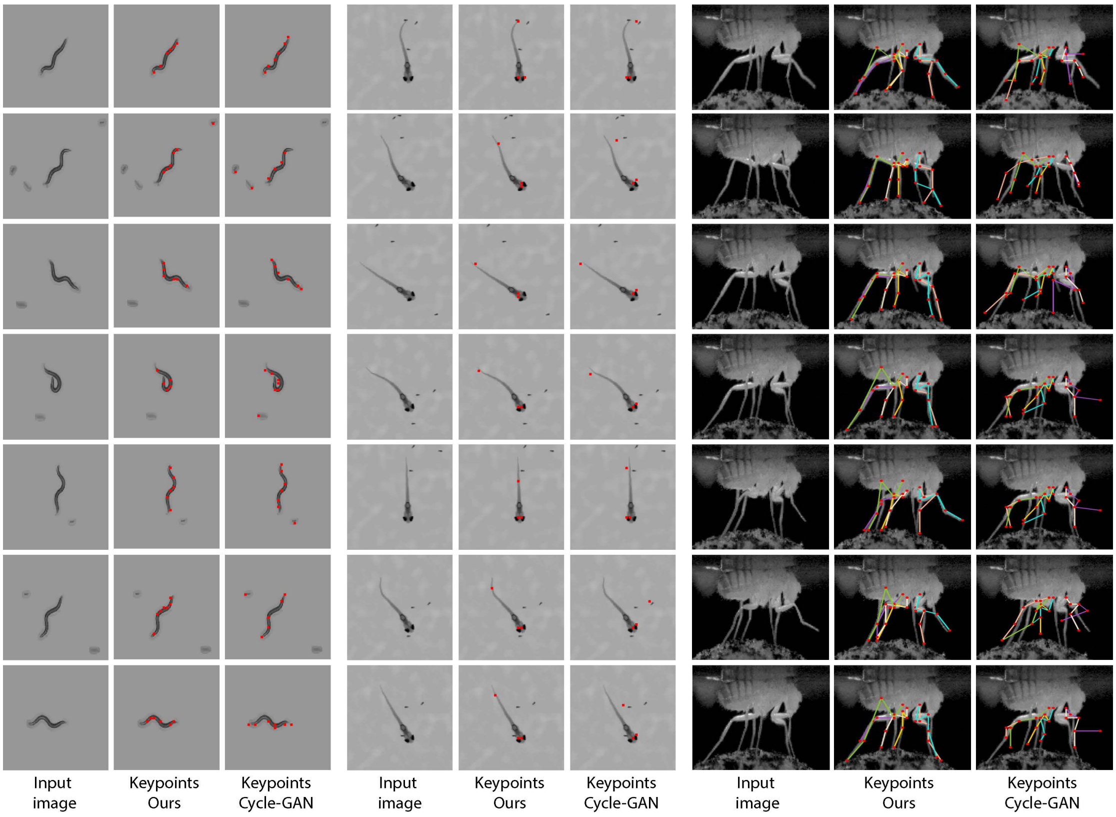


Figure 15 – **Pose estimation result comparison.** Training a pose estimator on our generated images yields accurate detections with far less failures when compared to Cycle-GAN, the best performing baseline. The *Drosophila* case is most challenging as the legs are thin and self-similar.

Part 4

Multi-view uncalibrated 3D pose estimation

Semih Günel, Helge Rhodin, Daniel Morales, Pavan Ramdya, Pascal Fua; DeepFly3D, a deep learning-based approach for 3D limb and appendage tracking in tethered, adult Drosophila, eLife, 2019.

4.1 Introduction

The precise quantification of movements is critical for understanding how neurons, biomechanics, and the environment influence and give rise to animal behaviors. For organisms with skeletons and exoskeletons, these measurements are naturally made with reference to 3D joint and appendage locations. Paired with modern approaches to simultaneously record the activity of neural populations in tethered, behaving animals [30–32], 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 the fly, *Drosophila melanogaster*.

However, algorithms for reliably estimating 3D pose in such small *Drosophila* sized animals have not yet been developed. Instead, multiple alternative approaches have been taken. For example, one can affix and use small markers—reflective, colored, or fluorescent particles—to identify and reconstruct keypoints from video data [161–163]. Although this approach works well on humans [164], in smaller, textitDrosophila-sized animals markers likely hamper movements, are difficult to mount on sub-millimeter scale limbs, and, most importantly, measurements of one or even two markers on each leg [163] cannot fully describe 3D limb kinematics. Another strategy has been to use computer vision techniques that operate without markers. However, these measurements have been restricted to 2D pose in freely behaving flies. Before the advent of deep learning, this was accomplished by matching the contours of animals seen against uniform backgrounds [165], measuring limb tip positions using complex TIRF-based imaging [166], or measuring limb segments using active contours [167]. In addition to being limited to 2D rather than 3D pose, these methods are complex, time-consuming, and error-prone in the face of long data sequences, cluttered backgrounds, fast motion, and occlusions that naturally occur when animals are observed from a single 2D perspective.

As a result, in recent years the computer vision community has largely forsaken these techniques in favor of deep learning-based methods. Consequently, the effectiveness of monocular 3D human pose estimation algorithms has improved greatly. This is especially true when capturing human movements for which there is enough annotated data to train deep networks effectively. Walking and upright poses are prime examples of this, and state-of-the-art algorithms [8, 10, 95–100, 168–170] now deliver impressive real-time results in uncontrolled environments. Increased robustness to occlusions can be obtained by using multi-camera setups [98, 171–173] and triangulating the 2D detections. This improves accuracy while making it possible to eliminate false detections.

These advances in 2D pose estimation have also recently been used to measure behavior in laboratory animals. For example, DeepLabCut provides a user-friendly interface to DeeperCut, a state-of-the-art human pose estimation network [174], and LEAP [124] can successfully track limb and appendage landmarks using a shallower network. Still, 2D pose provides an incomplete representation of animal behavior: important information can be lost due to occlusions, and movement quantification is heavily influenced by perspective.

Approaches used to translate human 2D to 3D pose have also been applied to larger animals, like lab mice and cheetahs, but require the use of calibration boards. These techniques cannot be easily transferred for the study of small animals like *Drosophila*: adult flies are approximately 2.5 mm long and precisely registering multiple camera viewpoints using traditional approaches would require the fabrication of a prohibitively small checkerboard pattern, along with the tedious labor of using a small, external calibration pattern. Moreover, flies have many appendages and joints, are translucent, and in most laboratory experiments are only illuminated using infrared light (to avoid visual stimulation)—precluding the exploitation of color information.

To overcome these challenges, we introduce DeepFly3D, a deep learning-based software pipeline that achieves comprehensive, rapid, and reliable 3D pose estimation in tethered, behaving adult *Drosophila* (16). DeepFly3D is applied to synchronized videos acquired from multiple cameras (24). It first uses a state-of-the-art deep network [9] and then enforces consistency across views (20). This makes it possible to eliminate spurious detections, achieve high 3D accuracy, and use 3D pose errors to further fine-tune the deep network to achieve even better accuracy (26). To register the cameras, DeepFly3D uses a novel calibration mechanism in which the fly itself is the calibration target (19). During the calibration process, we also employ sparse bundle adjustment methods, as previously used for human pose estimation [175–177]. Thus, the user doesn't need to manufacture a prohibitively small calibration pattern, or repeat cumbersome calibration protocols. We explain how users can modify the codebase to extend DeepFly3D for 3D pose estimation in other animals (23 and see Methods). Finally, we demonstrate that unsupervised behavioral embedding of 3D joint angle data (28) is robust against problematic artifacts present in embeddings of 2D pose data (27). In short, DeepFly3D delivers 3D pose estimates reliably, accurately, and with minimal manual intervention while also providing a critical tool for automated behavioral data analysis.

4.2 Method

With synchronized *Drosophila* video sequences from seven cameras in hand, the first task for DeepFly3D is to detect the 2D location of 38 landmarks. These 2D locations of the same landmarks seen across multiple views are then triangulated to produce 3D pose estimates. This pipeline is depicted in 17. First, we will describe our deep learning-based approach to detect landmarks in images. Then, we will explain the triangulation process that yields full 3D trajectories. Finally, we will describe how we identify and correct erroneous 2D detections automatically.

Deep Network Architecture. We aim to detect five joints on each limb, six on the abdomen, and one on each antenna, giving a total of 38 keypoints per time instance. To achieve this, we adapted a state-of-the-art Stacked Hourglass human pose estimation network [9] by changing the

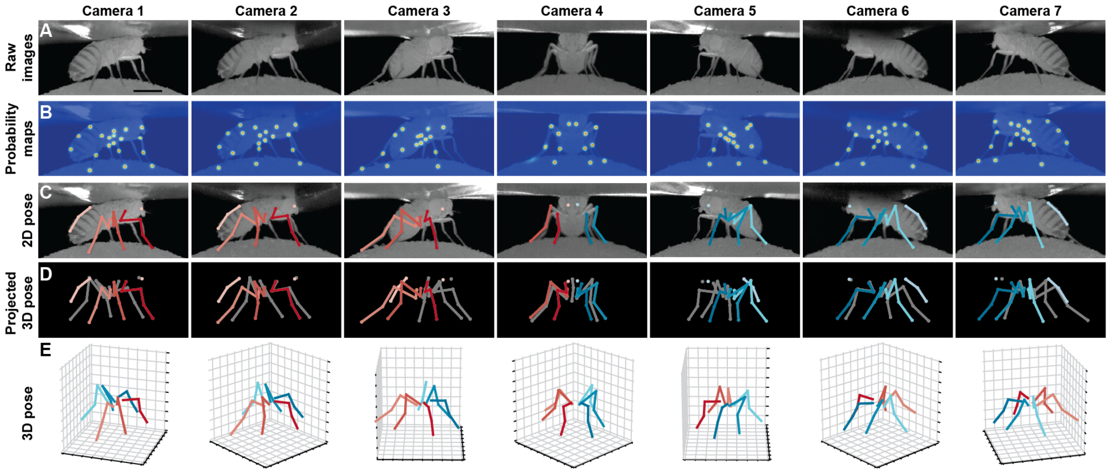


Figure 16 – Deriving 3D pose from multiple camera views. **(A)** Raw image inputs to the Stacked Hourglass deep network. **(B)** Probability maps output from the trained deep network. For visualization purposes, multiple probability maps have been overlaid for each camera view. **(C)** 2D pose estimates from the Stacked Hourglass deep network after applying pictorial structures and multi-view algorithms. **(D)** 3D pose derived from combining multiple camera views. For visualization purposes, the 3D pose has been projected onto the original 2D camera perspectives. **(E)** 3D pose rendered in 3D coordinates. Immobile thorax-coxa joints and antennal joints have been removed for clarity.

input and output layers to accommodate a new input image resolution and a different number of tracked points. A single hourglass stack consists of residual bottleneck modules with max pooling, followed by up-sampling layers and skip connections. The first hourglass network begins with a convolutional layer and a pooling layer to reduce the input image size from 256×512 to 64×128 pixels. The remaining hourglass input and output tensors are 64×128. We used 8 stacks of hourglasses in our final implementation. The output of the network is a stack of probability maps, also known as heatmaps or confidence maps. Each probability map encodes the location of one keypoint, as the belief of the network that a given pixel contains that particular tracked point. However, probability maps do not formally define a probability distribution: their sum over all pixels does not equal 1.

2D pose training dataset. We trained our network for 19 keypoints, resulting in the tracking of 38 points when both sides of the fly are accounted for. Determining which images to use for training purposes is critical. The intuitively simple approach—training with randomly selected images—may lead to only marginal improvements in overall network performance. This is because images for which network predictions can already be correctly made give rise to only small gradients during training. On the other hand, manually identifying images that may lead to incorrect network predictions is highly laborious. Therefore, to identify such challenging images, we exploited the redundancy of having multiple camera views (see section *3D pose correction*). Outliers in individual camera images were corrected automatically using images from other

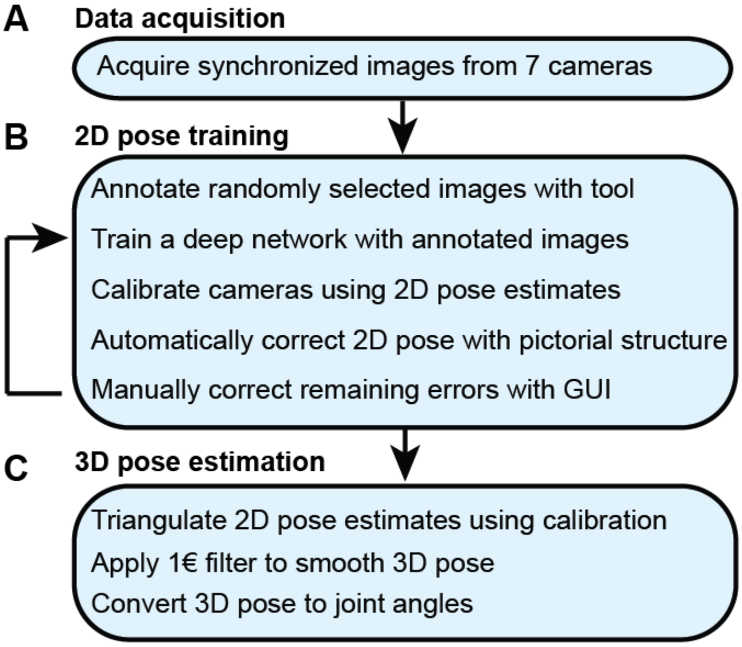

A
Data acquisition
Acquire synchronized images from 7 cameras
B
2D pose training
Annotate randomly selected images with tool
Train a deep network with annotated images
Calibrate cameras using 2D pose estimates
Automatically correct 2D pose with pictorial structure
Manually correct remaining errors with GUI
C
3D pose estimation
Triangulate 2D pose estimates using calibration
Apply 1€ filter to smooth 3D pose
Convert 3D pose to joint angles

Figure 17 – The DeepFly3D pose estimation pipeline. **(A)** Data acquisition from the multi-camera system. **(B)** Training and retraining of 2D pose. **(C)** 3D pose estimation.

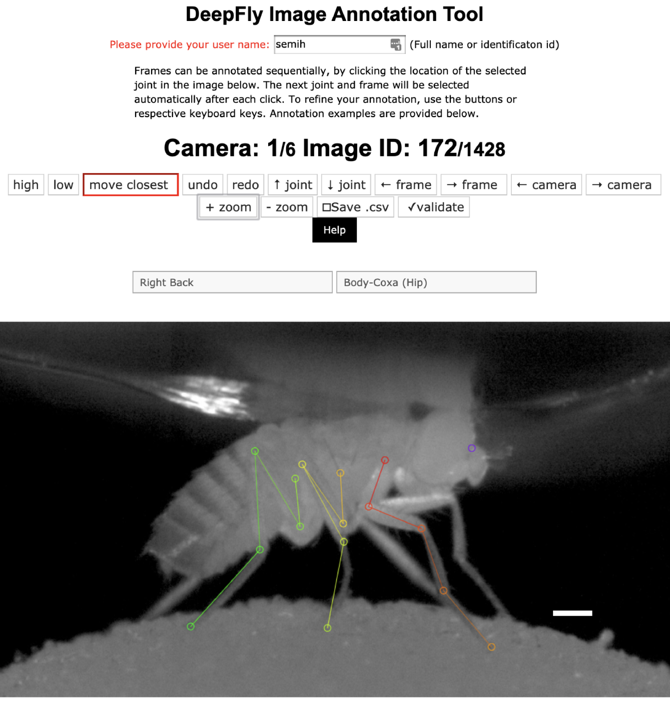
cameras, and frames that still exhibited large reprojection errors on multiple camera views were selected for manual annotation and network retraining. This combination of self supervision and active learning permits faster training using a smaller manually annotated dataset [173]. The full annotation and iterative training pipeline is illustrated in 17. In total, 40,063 images were annotated: 5,063 were labeled manually in the first iteration, 29,000 by automatic correction, and 6,000 by manually correcting those proposed by the active learning strategy.

Deep network training procedure. We trained our Stacked Hourglass network to regress from 256×512 pixel grayscale video images to multiple 64×128 probability maps. Specifically, during training and testing, networks output a $19 \times 64 \times 128$ tensor; one 64×128 probability map per tracked point. During training, we created probability maps by embedding a 2D Gaussian with mean at the ground-truth point and 1px symmetrical extent, i.e., with $\sigma = 1px$ on the diagonal of the covariance matrix. We calculated the loss as the L_2 distance between the ground-truth and predicted probability maps. During testing, the final network prediction for a given point was the probability map pixel with maximum probability. We started with a learning rate of 0.0001 and then multiplied the learning rate by a factor of 0.1 once the loss function plateaued for more than 5 epochs. We used an RMSPROP optimizer for gradient descent, following the original Stacked Hourglass implementation, with a batch-size of 8 images. Using 37,000 training images, the Stacked Hourglass network usually converges to a local minimum after 100 epochs (20 hours on a single GPU).

Network training details. Variations in each fly's position across experiments are handled by the translational invariance of the convolution operation. In addition, we artificially augment training images to improve network generalization for further image variables. These variables include (i) illumination conditions – we randomly changed the brightness of images using a gamma transformation, (ii) scale – we randomly rescaled images between 0.80x - 1.20x, and (iii) rotation – we randomly rotated images and corresponding probability maps ±15°. This augmentation was enough to compensate for real differences in the size and orientation of tethered flies across experiments. Furthermore, as per general practice, the mean channel intensity was subtracted from each input image to distribute annotations symmetrically around zero. We began network training using pretrained weights from the MPII human pose dataset [89]. This dataset consists of more than 25,000 images with 40,000 annotations, possibly with multiple ground-truth human pose labels per image. Starting with a pretrained network results in faster convergence. However, in our experience, this does not affect final network accuracy in cases with a large amount of training data. We split the dataset into 37,000 training images, 2,063 testing images, and 1,000 validation images. None of these subsets shared common images or common animals, to ensure that the network could generalize across animals, and experimental setups. 5,063 of our training images were manually annotated, and the remaining data were automatically collected using belief propagation, graphical models, and active learning, (see section *3D pose correction*). Deep neural network parameters need to be trained on a dataset with manually annotated ground-truth key point positions. To initialize the network, we collected annotations using a custom multicamera annotation tool that we implemented in JavaScript using Google Firebase (18). The DeepFly3D annotation tool operates on a simple web-server, easing the distribution of annotations across users and making these annotations much easier to inspect and control. We provide a GUI to inspect the raw annotated data and to visualize the network's 2D pose estimation (21).

Computing hardware and software. We trained our model on a desktop computing workstation running on an Intel Core i9-7900X CPU, 32 GB of DDR4 RAM, and a GeForce GTX 1080. With 37,000 manually and automatically labeled images, training takes nearly 20 hours on a single GeForce GTX 1080 GPU. Our code is implemented with Python 3.6, Pytorch 0.4 and CUDA 9.2. Using this desktop configuration, our network can run at 100 Frames-Per-Second (FPS) using the 8-stack variant of the Hourglass network, and can run at 420 FPS using the smaller 2-stack version. Thanks to an effective initialization step, calibration takes 3-4 s. Error checking and error correction can be performed at 100 FPS and 10 FPS, respectively. Error correction is only performed in response to large reprojection errors, and does not create a bottleneck in the overall speed of the pipeline.

Accuracy analysis. Consistent with the human pose estimation literature, we report accuracy as Percentage of Correct Keypoints (PCK) and Root Mean Squared Error (RMSE). PCK refers to the percentage of detected points lying within a specific radius from the ground-truth label. We set this threshold as 50 pixels, which is roughly one third of the 3D length of the femur. The final estimated position of each keypoint was obtained by selecting the pixel with the largest probability value on the relevant probability map. We compared DeepFly3D's annotations with



DeepFly Image Annotation Tool

Please provide your user name: semih (Full name or identificaton id)

Frames can be annotated sequentially, by clicking the location of the selected joint in the image below. The next joint and frame will be selected automatically after each click. To refine your annotation, use the buttons or respective keyboard keys. Annotation examples are provided below.

Camera: 1/6 Image ID: 172/1428

Right Back

Body-Coxa (Hip)

Figure 18 – The DeepFly3D annotation tool. This GUI allows the user to manually annotate joint positions on images from each of 7 cameras. Because this tool can be accessed from a web browser, annotations can be performed in a distributed manner across multiple users more easily. A full description of the annotation tool can be found in the online documentation: https://github.com/NeLy-EPFL/DeepFly3D. Scale bar is 50 pixels.

manually annotated ground-truth labels to test our model's accuracy. For RMSE, we report the square root of average pixel distance between the prediction and the ground-truth location of the tracked point. We remove trivial points such as the body-coxa and coxa-femur—which remain relatively stationary—to fairly evaluate our algorithms and to prevent these points from dominating our accuracy measurements.

4.2.1 From 2D landmarks to 3D trajectories

In the previous section, we described our approach to detect 38 2D landmarks. Let $\mathbf{x}_{c,j} \in \mathbb{R}^2$ denote the 2D position of landmark j and the image acquired by camera c. For each landmark, our task is now to estimate the corresponding 3D position, $\mathbf{X}_j \in \mathbb{R}^3$. To accomplish this, we used triangulation and bundle-adjustment [12] to compute 3D locations, and we used pictorial structures [178] to enforce geometric consistency and to eliminate potential errors caused by misdetections. We present these steps below.

Pinhole camera model. The first step is to model the projection operation that relates a specific $\mathbf{X}_j$ to its seven projections in each camera view $\mathbf{x}_{c,j}$. To make this easier, we follow standard practice and convert all Cartesian coordinates $\left[x_c, y_c, z_c\right]$ to homogeneous ones $\left[x_h, y_h, z_h, s\right]$ such that $x_c = x_h/s$, $y_c = y_h/s$, $z_c = z_h/s$. From now on, we will assume that all points are expressed in homogeneous coordinates and omit the h subscript. Assuming that these coordinates are expressed in a coordinate system whose origin is in the optical center of the camera and whose z-axis is its optical axis, the 2D image projection $\left[u, v\right]$ of a 3D homogeneous point $\left[x, y, z, 1\right]$ can be written as

$$\begin{aligned} u &= U/W\,, \\ v &= V/W\,, \\ \begin{bmatrix} U \\ V \\ W \end{bmatrix} &= \mathbf{K} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \text{ with } \mathbf{K} = \begin{bmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}, \end{aligned} \tag{7}$$

where the 3 × 4 matrix **K** is known as the *intrinsic parameters matrix*—scaling in the x and y direction and image coordinates of the principal point c_x and c_y—that characterizes the camera settings.

In practice, the 3D points are not expressed in a coordinate system tied to the camera, especially in our application where we use seven different cameras. Therefore, we use a world coordinate system that is common to all cameras. For each camera, we must therefore convert 3D coordinates expressed in this world coordinate system to camera coordinates. This requires rotating and translating the coordinates to account for the position of the camera's optical center and its orientation. When using homogeneous coordinates, this is accomplished by multiplying the coordinate vector by a 4 × 4 *extrinsic parameters matrix*

$$\mathbf{M} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ 0 & 1 \end{bmatrix}, \tag{8}$$

where **R** is a 3 × 3 rotation matrix and **T** a 3 × 1 translation vector. Combining 7 and 8 yields

$$\begin{aligned} u &= U/W\,, \\ v &= V/W\,, \\ \begin{bmatrix} U \\ V \\ W \end{bmatrix} &= \mathbf{P} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}, \text{ where } \mathbf{P} = \mathbf{MK} \text{ is a } 3 \times 4 \text{ matrix.} \end{aligned} \tag{9}$$

Camera distortion. The pinhole camera model described above is an idealized one. The projections of real cameras deviate from it and these deviations are referred to as distortions and need to be accounted for. The most significant one is known as radial distortion because the error grows with the distance to the image center. For the cameras we use, radial distortion can be expressed as

$$\begin{aligned} u_{\text{pinhole}} &= u\left(1 + \mathrm{k}_1^x \mathrm{r}^2 + \mathrm{k}_2^x \mathrm{r}^4\right) , \\ v_{\text{pinhole}} &= v\left(1 + \mathrm{k}_1^y \mathrm{r}^2 + \mathrm{k}_2^y \mathrm{r}^4\right) , \end{aligned} \tag{10}$$

where $\left[u, v\right]$ is the actual projection of a 3D point and $\left[u_{\text{pinhole}}, v_{\text{pinhole}}\right]$ is the one the pinhole model predicts. In other words, the four parameters $\{k_1^x, k_2^x, k_1^y, k_2^y\}$ characterize the distortion. From now on, we will therefore write the full projection as

$$\begin{aligned} \mathbf{X} &= \pi(\mathbf{x}) = f_d(f_p(\mathbf{x})) , \\ \mathbf{X} &= \left[x, y, z\right] , \\ \mathbf{x} &= \left[u, v\right] , \end{aligned} \tag{11}$$

where f_p denotes the ideal pinhole projection of 9 and f_d the correction of 10.

Triangulation. We can associate to each of the seven cameras a projection function π_c like the one in 11, where c is the camera number. Given a 3D point and its projections $\mathbf{x}_c$ in the images, its 3D coordinates can be estimated by minimizing the *reprojection error*

$$\underset{\mathbf{X} \in \mathbb{R}^4}{\operatorname{argmin}} \sum_{c=1}^{7} e_c \|\pi_c(\mathbf{X}) - \mathbf{x}_c\|_2^2 , \tag{12}$$

where e_c is one if the point was visible in image c and zero otherwise. In the absence of camera distortion, that is, when the projection π is a purely linear operation in homogeneous coordinates, this can be done for any number of cameras by solving a Singular Value Decomposition (SVD) problem [12]. In the presence of distortions, we replace the observed u and v coordinates of the projections by the corresponding u_{pinhole} and u_{pinhole} values of 11 before performing the SVD.

Camera calibration. Triangulating as described above requires knowing the projection matrices $\mathbf{P}_c$ of 9 for each camera c, corresponding distortion parameters $\{k_1^x, k_2^x, k_1^y, k_2^y\}$ of 10, together with the intrinsic parameters of focal length and principal point offset. In practice, we use the focal length and principal point offset provided by the manufacturer and estimate the remaining parameters automatically: the three translations and three rotations for each camera that define the corresponding matrix $\mathbf{M}$ of extrinsic parameters along with the distortion parameters.

To avoid having to design the exceedingly small calibration pattern that more traditional methods use to estimate these parameters, we use the fly itself as calibration pattern and minimize the reprojection error of 12 for all joints simultaneously while allowing the camera parameters to also change. In other words we look for

$$\underset{\substack{\pi_{c\,1\leq c\leq 7}\\ \mathbf{X_j}_{1\leq j\leq m}}}{\operatorname{argmin}} \sum_{c=1}^{7}\sum_{j=1}^{m} e_{c,j}\rho(\pi_c(\mathbf{X}_j)-\mathbf{x}_{c,j}), \tag{13}$$

where $\mathbf{X_j}$ and $\mathbf{x}_{c,j}$ are the 3D locations and 2D projections of the landmarks introduced above and ρ denotes the Huber loss. 13 is known as bundle-adjustment [12]. Huber loss is defined as

$$\rho_\delta(a) = \begin{cases} \frac{1}{2}a^2 & \text{for } |a| \leq \delta \\ \delta\left(|a| - \frac{1}{2}\delta\right) & \text{otherwise} \end{cases}.$$

Replacing the squared loss by the Huber loss makes our approach more robust to erroneous detections $\mathbf{x}_{c,j}$. We empirically set δ to 20 pixels. Note that we perform this minimization with respect to ten degrees-of-freedom per camera: three translations, three rotations, and four distortions.

For this optimization to work properly, we need to initialize these ten parameters and we need to reduce the number of outliers. To achieve this, the initial distortion parameters are set to zero. We also produce initial estimates for the three rotation and three translation parameters by measuring the distances between adjacent cameras and their relative orientations. To initialize the rotation and translation vectors, we measure the distance and the angle between adjacent cameras, from which we infer rough initial estimates. Finally, we rely on epipolar geometry [12] to automate outlier rejection. Because the cameras form a rough circle and look inward, the epipolar lines are close to being horizontal. Thus, corresponding 2D projections must belong to the same image rows, or at most a few pixels higher or lower. In practice, this means checking if all 2D predictions lie in nearly the same rows and discarding *a priori* those that do not.

4.2.2 3D pose correction

The triangulation procedure described above can produce erroneous results where the 2D estimates of landmarks are wrong. Additionally, it may result in implausible 3D poses for the entire animal because it treats each joint independently. To enforce more global geometric constraints, we rely on pictorial structures [178] as described in 20. Pictorial structures encode the relationship between a set of variables (in this case the 3D location of separate tracked points) in a probabilistic setting using a graphical model. This makes it possible to consider multiple 2D locations $\mathbf{x}_{c,j}$ for each landmark $\mathbf{X}_c$ instead of only one. This increases the likelihood of finding the true 3D pose.

Generating multiple candidates. Instead of selecting landmarks as the locations with the maximum probability in maps output by our Stacked Hourglass network, we generate multiple candidate 2D landmark locations $x_{c,j}$. From each probability map, we select ten local probability maxima that are at least one pixel apart from one another. Then, we generate 3D candidates by triangulating 2D candidates in every tuple of cameras. Because a single point is visible from at most four cameras, this results in at most $\binom{4}{2} \times 10^2$ candidates for each tracked point.

Choosing the best candidates. To identify the best subset of resulting 3D locations, we introduce the probability distribution $P(L|I,\theta)$ that assigns a probability to each solution L, consisting of 38 sets of 2D points observed from each camera. Our goal is then to find the most likely one. More formally, P represents the likelihood of a set of tracked points L, given the images, model parameters, camera calibration, and geometric constraints. In our formulation, I denotes the seven camera images $I = \{I_c\}_{1\leq c\leq 7}$ and θ represents the set of projection functions π_c for camera c along with a set of length distributions $S_{i,j}$ between each pair of points i and j that are connected by a limb. L consists of a set of tracked points $\{L_i\}_{1\leq i\leq n}$, where each L_i describes a set of 2D observations $l_{i,c}$ from multiple camera views. These are used to triangulate the corresponding 3D point locations $\overline{l_i}$. If the set of 2D observations is incomplete, as some points are totally occluded in some camera views, we triangulate the 3D point $\bar{l}_i$ using the available ones and replace the missing observations by projecting the recovered 3D positions into the images, $\pi_c(\bar{l}_i)$ in 9. In the end, we aim to find the solution $\hat{L} = \operatorname{argmax}_L P(L|I,\theta)$. This is known as Maximum a Posteriori (MAP) estimation. Using Bayes rule, we write

$$P(L|I,\theta) \propto P(I|L,\theta)P(L|\theta)\,, \tag{14}$$

where the two terms can be computed separately. We compute $P(I|J,\theta)$ using the probability maps $H_{j,c}$ generated by the Stacked Hourglass network for the tracked point j for camera c. For a single joint j seen by camera c, we model the likelihood of observing that particular point using $P(H_{j,c}|l_{j,c})$, which can be directly read from the probability maps as the pixel intensity. Ignoring the dependency between the cameras, we write the overall likelihood as the product of the individual likelihood terms

$$P(I|L,\theta) = P(H|L) \propto \prod_{i=1}^{n}\prod_{c=1}^{7} P(H_{j,c}|l_{i,c})\,,$$

which can be read directly from the probability maps as pixel intensities and represent the network's confidence that a particular keypoint is located at a particular pixel. When a point is not visible from a particular camera, we assume the probability map only contains a constant non-zero probability, which does not effect the final solution. We express $P(L|\theta)$ as

$$P(L|\theta) = P(L|\pi,S) = \prod_{(i,j)\in E} P\left(\bar{l}_i,\bar{l}_j|S_{i,j}\right)\prod_{j=1}^{n}\prod_{c=1}^{7} e_{c,j}\|\pi_c(\bar{l}_j) - l_{c,j}\|_2^{-1}\,,$$

where pairwise dependencies $P\left(\bar{l}_i,\bar{l}_j|S_{i,j}\right)$ between two variables respect the segment length constraint when the variables are connected by a limb. The length of segments defined by pairs of connected 3D points follows a normal distribution. Specifically, we model $P\left(\bar{l}_i,\bar{l}_j|S_{i,j}\right)$ as $S_{i,j}(\bar{l}_i,\bar{l}_j) = \mathcal{N}(\|\bar{l}_i - \bar{l}_j\| - \mu_{i,j},\sigma_{i,j})$. We model the reprojection error for a particular point j as $\prod_{c=1}^{7} e_{c,j}\|\pi_c(\bar{l}_j) - l_{c,j}\|_2^{-1}$ which is set to zero using the variable $e_{c,j}$ denoting the visibility of the point j from camera c. If a 2D observation for a particular camera is manually set by a user with the DeepFly3D GUI, we take it to be the only possible candidate for that particular image and we set $P(L_j|H)$ to 1, where j denotes the manually assigned pixel location.

Solving the MAP problem using the Max-Sum algorithm. For general graphs, MAP estimation with pairwise dependencies is NP-hard and therefore intractable. However, in the specific case of non-cyclical graphs, it is possible to solve the inference problem using belief propagation [179]. Since the fly's skeleton has a root and contains no loops, we can use a message passing approach [178]. It is closely related to Viterbi recurrence and propagates the unary probabilities $P(L_j|L_i)$ between the edges of the graph starting from the root and ending at the leaf nodes. This first propagation ends with the computation of the marginal distribution for the leaf node variables. During the subsequent backward iteration, as $P(L_j)$ for leaf node is computed, the point L_j with maximum posterior probability is selected in $O(k)$ time, where k is the upper bound on the number of proposals for a single tracked point. Next, the distribution $P(L_i|L_j)$ is calculated, adjacent nodes for the leaf node. Continuing this process on all of the remaining points results in a MAP solution for the overall distribution $P(L)$, as shown in 20, with overall $O(k^2)$ computational complexity.

Learning the parameters. We learn the parameters for the set of pairwise distributions $S_{i,j}$ using a maximum likelihood process and assuming the distributions to be Gaussian. We model the segment length $S_{i,j}$ as the euclidean distance between the points $\bar{l}_j$ and $\bar{l}_j$. We then solve for $\text{argmax}_S\, P(S|L,\theta)$, assuming segments have a Gaussian distribution resulting from the Gaussian noise in point observations L. This gives us the mean and variance, defining each distribution $S_{i.j}$. We exclude the same points that we removed from the calibration procedure, that exhibit high reprojection error.

In practice, we observe a large variance for pretarsus values. This is because occlusions occasionally shorten visible tarsal segments. To eliminate the resulting bias, we treat these limbs differently from the others and model the distribution of tibia-tarsus and tarsus-tip points as a Beta distribution, with parameters found using a similar Maximum Likelihood Estimator (MLE) formulation. Assuming the observation errors to be Gaussian and zero-centered, the bundle adjustment procedure can also be understood as an MLE of the calibration parameters [176]. Therefore, the entire set of parameters for the formulation can be learned using MLE. Thus, prior information about potentially occluded targets can be used to guide inference. For example, in a head-fixed rodent, the left eye may not always be visible from the right-side of the animal. This information can be incorporated into DeepFly3D's inference system in the file, 'skeleton.py', by editing the function 'camera_see_joint'. Afterwards, predictions from occluded cameras will not be used to triangulate a given 3D point. If no such information is provided, every prediction will be used to triangulate a given 3D point.

The pictorial structure formulation can be further expanded using temporal information, penalizing large movements of a single tracked point between two consecutive frames. However, we abstained from using temporal information more extensively for several reasons. First, temporal dependencies would introduce loops in our pictorial structures, thus making exact inference

NP-hard as discussed above. This can be handled using loopy belief propagation algorithms [180] but requires multiple message passing rounds, which prevents real-time inference without any theoretical guarantee of optimal inference. Second, the rapidity of *Drosophila* limb movements makes it hard to assign temporal constraints, even with fast video recording. Finally, we empirically observed that the current formulation, enforcing structured poses in a single temporal frame, already eliminates an overwhelming majority of false-positives inferred during the pose estimation stage of the algorithm.

Modifying DeepFly3D to study other animals. As DeepFly3D does not assume a circular camera arrangement or that there is one degree of freedom in the camera network, it could easily be adapted for 3D pose estimation in other animals, ranging from rodents to primates and humans (23). We illustrate this flexibility by using DeepFly3D to capture human 3D pose in the Human 3.6M Dataset (http://vision.imar.ro/human3.6m/description.php), a very popular, publicly available computer vision benchmarking dataset generated using four synchronized cameras [29, 181].

Generally, for any new dataset, the user first needs to provide an initial set of manual annotations. The user would describe the number of tracked points and their relationships to one another in a python setup file. Then, in a configuration file, the user specify the number of cameras along with the resolutions of input images and output probability maps. DeepFly3D will then use these initial manual annotations to (i) train the 2D Stacked Hourglass network, (ii) perform camera calibration without an external calibration pattern, (iii) learn the epipolar geometry to perform outlier detection, and (iv) learn the segment length distributions $S_{i,j}$. After this initial bootstrapping, DeepFly3D can be then used with pictorial structures and active learning to iteratively improve pose estimation accuracy.

The initial manual annotations can be performed using the DeepFly3D Annotation GUI. Afterwards, these annotations can be downloaded from the Annotation GUI as a CSV file using the *Save* button (18). Once the CSV file is placed in the images folder, DeepFly3D will automatically read and display the annotations. To train the Stacked Hourglass network, use the *csv-path* flag while running *pose2d.py* (found in *deepfly/pose2d*). DeepFly3D will then train the Stacked Hourglass network by performing transfer learning using the large MPII dataset and the smaller set of user manual annotations.

To perform camera calibration, the user should select the *Calibration* button on the GUI 21. DeepFly3D will then perform bundle adjustment (13) and save the camera parameters in *calibration.pickle* (found in the images folder). The path of this file should then be added to *Config.py* to initialize calibration. These initial calibration parameters will then be used in further experiments for fast and accurate convergence. If the number of annotations is insufficient for accurate calibration, or if bundle adjustment is converging too slowly, an initial rough estimate of the camera locations can be set in *Config.py*. As long as a calibration is set in *Config.py*, DeepFly3D will use it as a projection matrix to calculate the epipolar geometry between cameras. This step is necessary to perform outlier detection on further calibration operations.

DeepFly3D will also learn the distribution $S_{i,j}$, whose non-zero entries are found in *skeleton.py*. One can easily calculate these segment length distribution parameters using the functions provided with DeepFly3D. *CameraNetwork* class (found under deepfly/GUI/), will then automatically load the points and calibration parameters from the images folder. The function *CameraNetwork.triangulate* will convert 2D annotation points into 3D points using the calibration parameters. The $S_{i,j}$ parameters can then be saved using the *pickle* library (the save path can be set in *Config.py*). The *calcBoneParams* method will then output the segment lengths' mean and variance. These values will then be used with pictorial structures (14).

We provide further technical details for how to adapt DeepFly3D to other multi-view datasets online [1].

4.2.3 Experimental setup

We positioned seven Basler acA1920-155um cameras (FUJIFILM AG, Niederhaslistrasse, Switzerland) 94 mm away from the tethered fly, resulting in a circular camera network with the animal in the center (24). We acquired 960 × 480 pixel video data at 100 FPS under 850 nm infrared ring light illumination (Stemmer Imaging, Pfäffikon Switzerland). Cameras were mounted with 94 mm W.D. / 1.00x InfiniStix lenses (Infinity Photo-Optical GmbH, Göttingen). Optogenetic stimulation LED light was filtered out using 700 nm longpass optical filters (Edmund Optics, York UK). Each camera's depth of field was increased using 5.8 mm aperture retainers (Infinity Photo-Optical GmbH). To automate the timing of optogenetic LED stimulation and camera acquisition triggering, we use an Arduino (Arduino, Sommerville MA USA) and custom software written using the Basler camera API.

We assessed the optimal number of cameras for DeepFly3D and concluded that increasing the number of cameras increases accuracy by stabilizing triangulation. Specifically, we observed the following. (i) Calibration is not a significant source of error: calibrating with fewer than 7 cameras does not dramatically increase estimation error. (ii) Having more cameras improves triangulation. Reducing the number of cameras down to four, even having calibrated with 7 cameras, results in an increase of 0.05 mm triangulation error. This may be because the camera views are sufficiently different, having largely non-overlapping 2D-detection failure cases. Thus, the redundancy provided by having more cameras mitigates detection errors by finding a 3D pose that is consistent across at least two camera views.

***Drosophila* transgenic lines.** *UAS-CsChrimson* [182] animals were obtained from the Bloomington Stock Center (Stock #55135). *MDN-1-Gal4* [183] (*VT44845-DBD*; *VT50660-AD*) was provided by B. Dickson (Janelia Research Campus, Ashburn USA). *aDN-Gal4* [184](*R76F12-AD*; *R18C11-DBD*), was provided by J. Simpson (University of California, Santa Barbara USA). Wild-type, *PR* animals were provided by M. Dickinson (California Institute of Technology, Pasadena USA).

---

[1] https://github.com/NeLy-EPFL/DeepFly3D

Optogenetic stimulation experiments. Experiments were performed in the late morning or early afternoon Zeitgeber time (Z.T.), inside a dark imaging chamber. An adult female animal 2-3 days-post-eclosion (dpe), was mounted onto a custom stage [32] and allowed to acclimate for 5 minutes on an air-supported spherical treadmill [32]. Optogenetic stimulation was performed using a 617 nm LED (Thorlabs, Newton, NJ USA) pointed at the dorsal thorax through a hole in the stage, and focused with a lens (LA1951, 01" f = 25.4 mm, Thorlabs, Newton, NJ USA). Tethered flies were otherwise allowed to behave spontaneously. Data were acquired in 9 s epochs: 2 s baseline, 5 s with optogenetic illumination, and 2 s without stimulation. Individual flies were recorded for 5 trials each, with one-minute intervals. Data were excluded from analysis if flies pushed their abdomens onto the spherical treadmill—interfering with limb movements—or if flies struggled during optogenetic stimulation, pushing their forelimbs onto the stage for prolonged periods of time.

4.2.4 Unsupervised behavioral classification

To create unsupervised embeddings of behavioral data, we mostly followed the approach taken by [163, 185]. We smoothed 3D pose traces using a 1euro filter. Then we converted them into angles to achieve scale and translational invariance [186]. Angles were calculated by taking the dot product from sets of three connected 3D positions. For the antenna, we calculated the angle of the line defined by two antennal points with respect to the ground-plane. This way, we generated four angles per leg (two body-coxa, one coxa-femur, and one femur-tibia), two angles for the abdomen (top and bottom abdominal stripes), and a single angle for the antennae (head tilt with respect to the axis of gravity). In total, we obtained a set of 20 angles, extracted from 38 3D points.

We transformed angular time series using a Continous Wavelet Transform (CWT) to create a posture-dynamics space. We used the Morlet Wavelet as the mother wavelet, given its suitability to isolate periodic chirps of motion. We chose 25 wavelet scales to match dyadically spaced center frequencies between 5Hz and 50Hz. Then, we calculatd spectrograms for each postural time-series by taking the magnitudes of the wavelet coefficients. This yields a $20 \times 25 = 500$-dimensional time-series, which was then normalized over all frequency channels to unit length, at each time instance. Then, we could treat each feature vector from each time instance as a distribution over all frequency channels.

Later, from the posture-dynamics space, we computed a two-dimensional representation of behavior by using the non-linear embedding algorithm, t-SNE [187]. t-SNE embedded our high-dimensional posture-dynamics space onto a 2D plane, while preserving the high-dimensional local structure, while sacrificing larger scale accuracy. We used the Kullback–Leibler (KL) divergence as the distance function in our t-SNE algorithm. KL assesses the difference between the shapes of two distributions, justifying the normalization step in the preceding step. By analyzing a multitude of plots generated with different perplexity values, we empirically found perplexity 35 to best suit the features of our posture-dynamics space.

From this generated discrete space, we created a continuous 2D distribution, that we could then segment into behavioral clusters. We started by normalizing the 2D t-SNE projected space into a 1000×1000 matrix. Then, we applied a 2D Gaussian convolution with a kernel of size $\sigma = 10$px. Finally, we segmented this space by inverting it and applying a Watershed algorithm that separated adjacent basins, yielding a behavioral map.

4.3 Evaluation

4.3.1 DeepFly3D

The input to DeepFly3D is video data from seven cameras (24). These images are used to identify the 3D positions of 38 landmarks per animal: (i) five on each limb – the thorax-coxa, coxa-femur, femur-tibia, and tibia-tarsus joints as well as the pretarsus, (ii) six on the abdomen - three on each side, and (iii) one on each antenna - useful for measuring head rotations. Our software incorporates a number of innovations designed to ensure automated, high-fidelity, and reliable 3D pose estimation:

Calibration without an external calibration pattern: Estimating 3D pose from multiple images requires calibrating the cameras to achieve a level of accuracy that is commensurate with the target size—a difficult challenge when measuring leg movements for an animal as small as *Drosophila*. Therefore, instead of using a typical external calibration grid, DeepFly3D uses the fly itself as a calibration target. It detects arbitrary points on the fly’s body and relies on bundle-adjustment [188] to simultaneously assign 3D locations to these points and to estimate the positions and orientations of each camera (19). To increase robustness, it enforces geometric constraints that apply to tethered flies with respect to limb segment lengths and ranges of motion.

Geometrically consistent reconstructions: Starting with a state-of-the-art deep network for 2D keypoint detection in individual images [9], DeepFly3D enforces geometric consistency constraints across multiple synchronized camera views. When triangulating 2D detections to produce 3D joint locations, it relies on pictorial structures and belief propagation message passing [178] to detect and further correct erroneous pose estimates (20).

Self-supervision and active learning: We also use multiple view geometry as a basis for active learning. Thanks to the redundancy inherent in obtaining multiple views of the same animal, we can detect erroneous 2D predictions for correction (22) that would most efficiently train the 2D pose deep network. This approach greatly reduces the need for time-consuming manual labeling [173]. We also use pictorial structure corrections to fine-tune the 2D pose deep network. Self-supervision constitutes 85% of our training data.

4.3.2 2D pose performance and improvement using pictorial structures

We validated our approach using a challenging dataset of 2063 image frames manually annotated using the DeepFly3D annotation tool (18) and sampled uniformly from each camera. Images for testing and training were 480 × 960 pixels. The test dataset included challenging frames and occasional motion blur to increase the difficulty of pose estimation. For training, we used a final training dataset of 37,000 frames, an overwhelming majority of which were first automatically corrected using pictorial structures. On test data, we achieved a Root Mean Square Error (RMSE) of 13.9 pixels. Compared with a ground truth RMSE of 12.4 pixels – via manual annotation of 210 images by a new human expert – our Network Annotation / Manual Annotation ratio of 1.12 (13.9 pixels / 12.4 pixels) is similar to another state-of-the-art network [174]: 1.07 (2.88 pixels / 2.69 pixels). Setting a 50 pixel threshold (approximately one third the length of a femur) for PCK (percentage of correct keypoints) computation, we observed a 98.2% general accuracy before applying pictorial structures. Notably, if we reduced our threshold to 30 or 20 pixels, we could still achieve 95% or 89% accuracy, respectively (25*A*).

To test the performance of the network in a low data regime, we trained a two-stacked network using ground-truth annotations data from 7 cameras (25*B*). We compared the results to an asymptotic prediction error (i.e., the error observed when the network is trained using the full dataset of 40,000 annotated images) and to the variability observed in human annotations of 210 randomly selected images. We measured an asymptotic MAE (mean absolute error) of 10.5 pixels and a human variability MAE of 9.2 pixels. With 800 annotations, our network achieved a similar accuracy to manual annotation and was near the asymptotic prediction error. Further annotation yielded diminishing returns.

Although our network achieves high accuracy, the error is not isotropic (25*C*). The tarsus tips exhibited larger error than the other joints, perhaps due to occlusions from the spherical treadmill, and higher positional variance. Increased error observed for body-coxa joints might be due to the difficulty of annotating these landmarks from certain camera views.

To correct the residual errors, we applied pictorial structures. This strategy fixed 59% of the remaining erroneous predictions, increasing the final accuracy to 99.2%, from 98.2%. These improvements are illustrated in 26. Pictorial structure failures were often due to pose ambiguities resulting from heavy motion blur. These remaining errors were automatically detected with multi-view redundancy using 12, and earmarked for manual correction using the DeepFly3D GUI (21).

4.3.3 3D pose permits robust unsupervised behavioral classification

Unsupervised behavioral classification approaches enable the unbiased quantification of animal behavior by processing data features—image pixel intensities [185, 189], limb markers [163], or 2D pose [124]—to cluster similar behavioral epochs without user intervention and to automatically distinguish between otherwise similar actions. However, with this sensitivity may come a susceptibility to features unrelated to behavior. These may include changes in image size or perspective resulting from differences in camera angle across experimental systems, variable mounting of tethered animals, and inter-animal morphological variability. In theory, each of these issues can be overcome—providing scale and rotational invariance—by using 3D joint angles rather than 2D pose for unsupervised embedding.

To test this possibility, we performed unsupervised behavioral classification on video data taken during optogenetic stimulation experiments that repeatedly and reliably drove specific actions. Specifically, we optically activated CsChrimson [190] to elicit backward walking in MDN>CsChrimson animals (28sv1) [183], or antennal grooming in aDN>CsChrimson animals (28sv2) [184]. We also stimulated control animals lacking the UAS-CsChrimson transgene (28sv3)(MDN-GAL4/+ and aDN-GAL4/+). First, we performed unsupervised behavioral classification using 2D pose data from three adjacent cameras containing keypoints for three limbs on one side of the body. Using these data, we generated a behavioral map (27*A*). In this map each individual cluster would ideally represent a single behavior (e.g., backward walking, or grooming) and be populated by nearly equal amounts of data from each of the three cameras. This was not the case: data from each camera covered non-overlapping regions and clusters (27*B-D*). This effect was most pronounced when comparing regions populated by cameras 1 and 2 versus camera 3. Therefore, because the underlying behaviors were otherwise identical (data across cameras were from the same animals and experimental time points), we concluded that unsupervised behavioral classification of 2D pose data is highly sensitive to corruption by viewing angle differences.

By contrast, performing unsupervised behavioral classification using DeepFly3D-derived 3D joint angles resulted in a map (28) with clear segregation and enrichment of clusters for different GAL4 drivers lines and their associated behaviors (i.e., backward walking, grooming, and forward walking. Thus, 3D pose overcomes serious issues arising from the unsupervised embedding of 2D pose data, enabling more reliable and robust behavioral data analysis.

4.4 Conclusion

We have developed DeepFly3D, a deep learning-based 3D pose estimation system that is optimized for quantifying limb and appendage movements in tethered, behaving *Drosophila*. By using multiple synchronized cameras and exploiting multiview redundancy, our software delivers robust and accurate pose estimation at the sub-millimeter scale. Ultimately, we may work solely with monocular images by lifting the 2D detections [98] to 3D or by directly regressing to 3D [170] as has been achieved in human pose estimation studies. Our approach relies on supervised deep learning to train a neural network that detects 2D joint locations in individual camera images. Importantly, our network becomes increasingly competent as it runs: By leveraging the redundancy inherent to a multiple-camera setup, we iteratively reproject 3D pose to automatically detect and correct 2D errors, and then use these corrections to further train the network without user intervention.

None of the techniques we have put together—an approach for multiple-camera calibration that uses the animal itself rather than an external apparatus, an iterative approach to inferring 3D pose using graphical models as well as optimization based on dynamic programming and belief propagation, and a graphical user interface and active learning policy for interacting with, annotating, and correcting 3D pose data—are fly-specific. They could easily be adapted to other limbed creatures, from mice to primates and humans. The only thing that would have to change significantly are the dimensions of the setup. This would remove the need to deal with the very small scales that *Drosophila* requires and would, in practice, make things easier. In the Methods section, we explain in detail how organism-specific features of DeepFly3D—bone segment length, number of legs, and camera focal distance—can be modified to study, for example, humans (23), primates, rodents, or other insects.

As in the past, we anticipate that the development of new technologies for quantifying behavior will open new avenues and enhance existing lines of investigation. For example, deriving 3D pose using DeepFly3D can improve the resolution of studies examining how neuronal stimulation influences animal behavior [189, 191], the precision and predictive power of efforts to define natural action sequences [191, 192], the assessment of interventions that target models of human disease [193, 194], and the linking of neural activity with animal behavior—when coupled with recording technologies like 2-photon microscopy [31, 32]. Importantly, 3D pose dramatically improves the robustness of unsupervised behavioral classification approaches. Therefore, DeepFly3D is a critical step toward the ultimate goal of achieving fully-automated, high-fidelity behavioral data analysis.

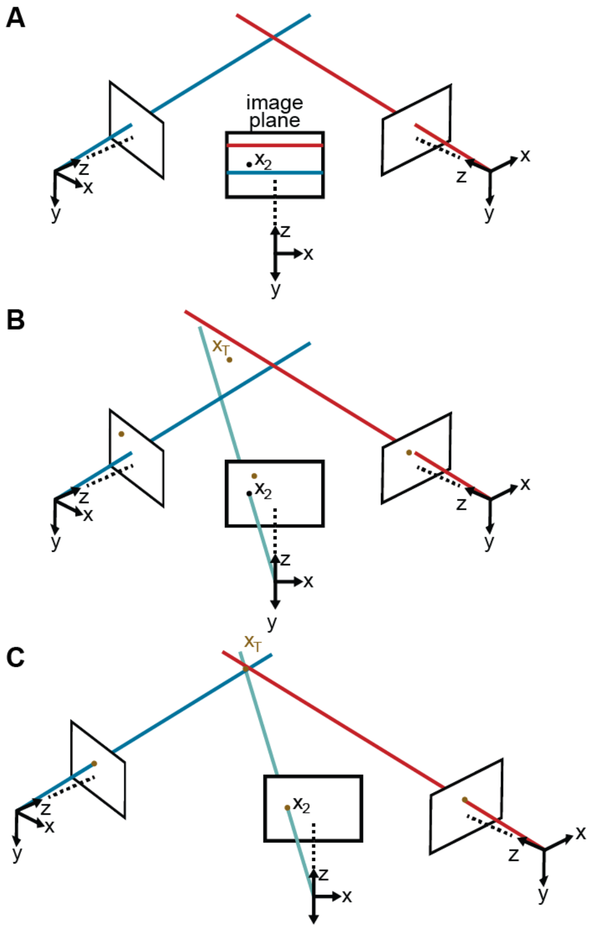


Figure 19 – Camera calibration. **(A)** Correcting erroneous 2D pose estimations by using epipolar relationships. Only 2D pose estimates without large epipolar errors are used for calibration. x_2 represents a 2D pose estimate from the middle camera. Epipolar lines are indicated as blue and red lines on the image plane. **(B)** The triangulated point, X_T, uses the initial camera parameters. However, due to the coarse initialization of each camera's extrinsic properties, observations from each camera do not agree with one another and do not yield a reasonable 3D position estimate. **(C)** The camera locations are corrected, generating an accurate 3D position estimate by optimizing Equation 13 using only the pruned 2D points.

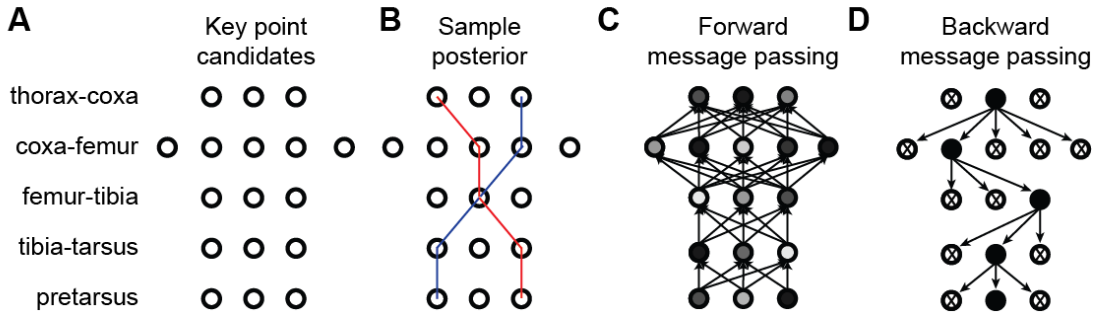


Figure 20 – 3D pose correction for one leg using the MAP solution and pictorial structures. **(A)** Candidate 3D pose estimates for each keypoint are created by triangulating local maxima from probability maps generated by the Stacked Hourglass deep network. **(B)** For a selection of these candidate estimates, we can assign a probability using Equation 14. However, calculating this probability for each pair of points is computationally intractable. **(C)** By exploiting the chain structure of Equation 14, we can instead pass a probability distribution across layers using a belief propagation algorithm. Messages are passed between layers as a function of parent nodes, describing the belief of the child nodes on each parent node. Grayscale colors represent the calculated belief of each node where darker colors indicate higher belief. **(D)** Corrected pose estimates are obtained during the second backward iteration, by selecting the nodes with largest belief. We discard nodes (x's) that have non-maximal belief during backwards message passing. Note that beliefs have been adjusted after forward message passing.

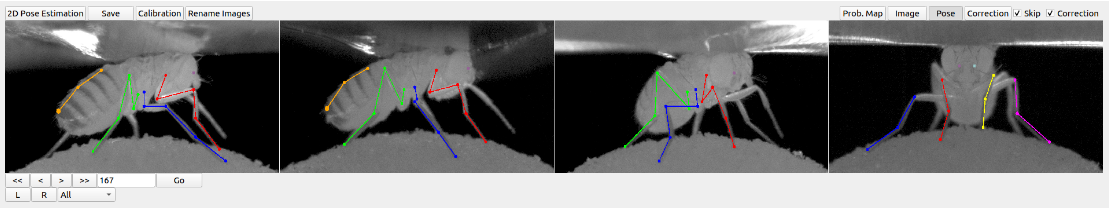


Figure 21 – DeepFly3D graphical user interface (GUI). The top-left buttons enable operations like 2D pose estimation, camera calibration, and saving the final results. The top-right buttons can be used to visualize the data in different ways: as raw images, probability maps, 2D pose, or the corrected pose following pictorial structures. The bottom-left buttons permit frame-by-frame navigation. A full description of the GUI can be found in the online documentation: https://github.com/NeLy-EPFL/DeepFly3D

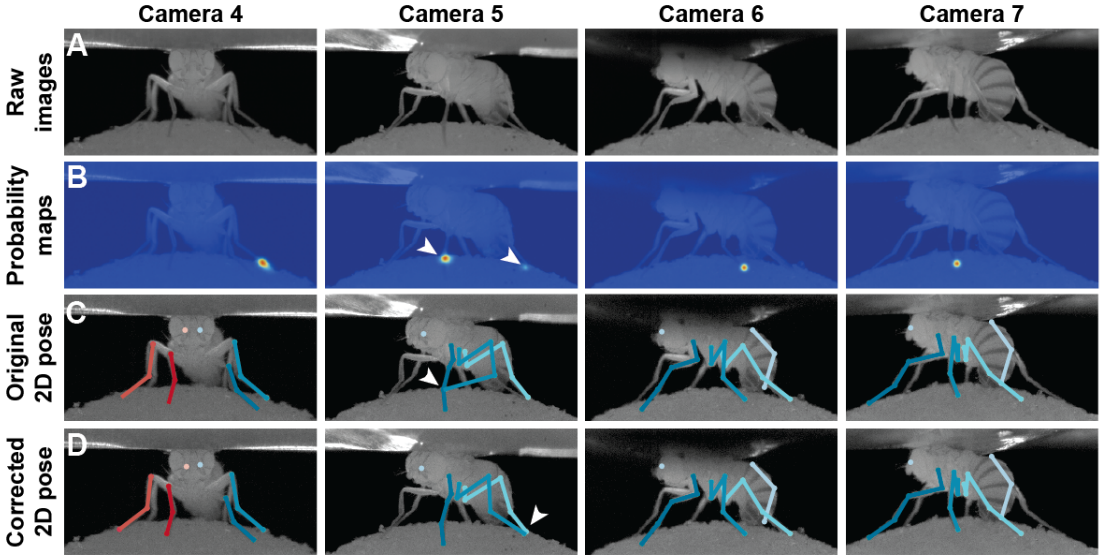


Figure 22 – Pose correction using pictorial structures. **(A)** Raw input data from four cameras, focusing on the pretarsus of the middle left leg. **(B)** Probability maps for the pretarsus output from the Stacked Hourglass deep network. Two maxima (white arrowheads) are present on the probability maps for camera 5. The false-positive has a larger unary probability. **(C)** Raw predictions of 2D pose estimation without using pictorial structures. The pretarsus label is incorrectly applied (white arrowhead) in camera 5. By contrast, cameras 4, 6, and 7 are correctly labeled. **(D)** Corrected pose estimation using pictorial structures. The false-positive is removed due to the high error measured in Equation 14. The newly corrected pretarsus label for camera 5 is shown (white arrowhead).

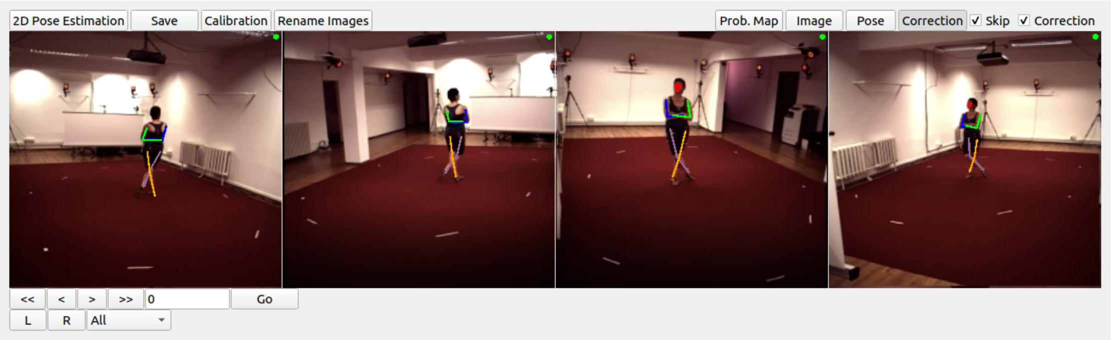


Figure 23 – DeepFly3D graphical user interface (GUI) used with the Human3.6M dataset [29]. To use the DeepFly3D GUI on any new dataset (*Drosophila* or otherwise), users can provide an initial small set of manual annotations. Using these annotations, the software calculates the epipolar geometry, performs camera calibration, and trains the 2D pose estimation deep network. A description of how to adopt DeepFly3D for new datasets can be found in the Methods section and, in greater detail, online: https://github.com/NeLy-EPFL/DeepFly3D

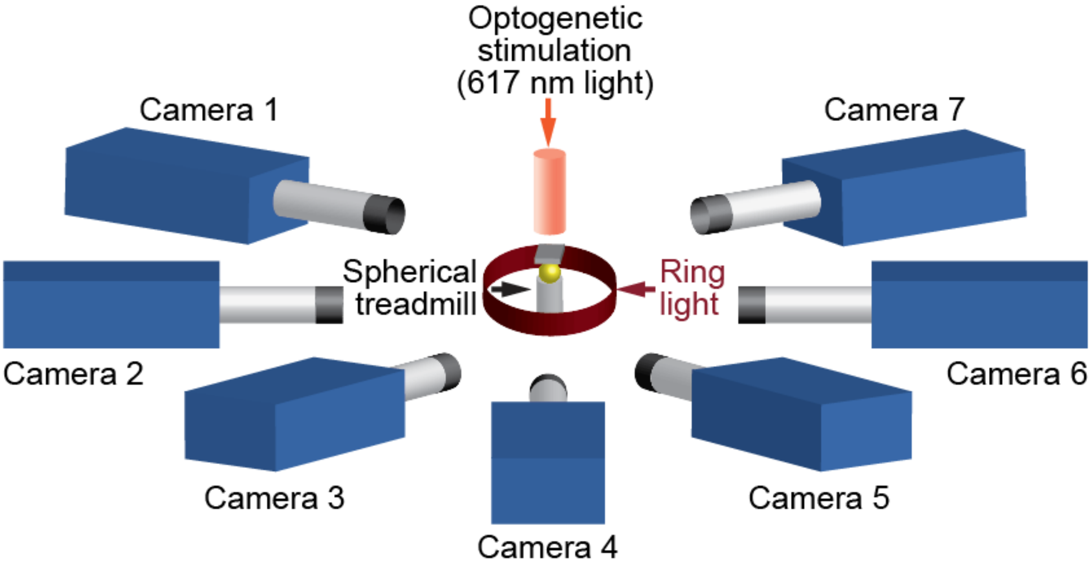


Figure 24 – A schematic of the seven camera spherical treadmill and optogenetic stimulation system that was used in this study.

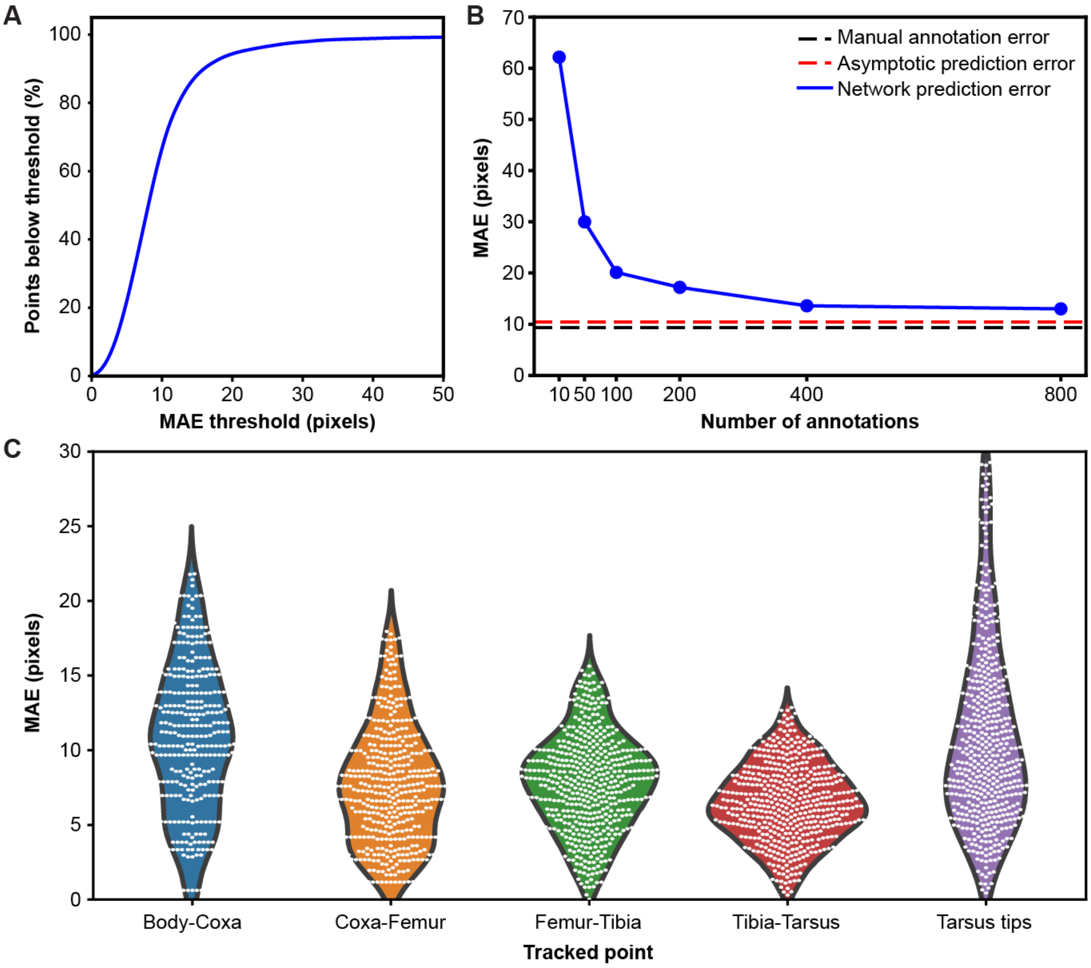


Figure 25 – Mean Absolute Error distribution. **(A)** PCK (percentage of keypoints) accuracy as a function of mean absolute error (MAE) threshold. **(B)** Evaluating network prediction error in a low data regime. The Stacked Hourglass network (blue circles) shows near asymptotic prediction error (red dashed line), even when trained with only 400 annotated images. After 800 annotations, there are minimal improvements to the MAE. **(C)** MAE for different limb landmarks. Violin plots are overlaid with raw data points (white circles).

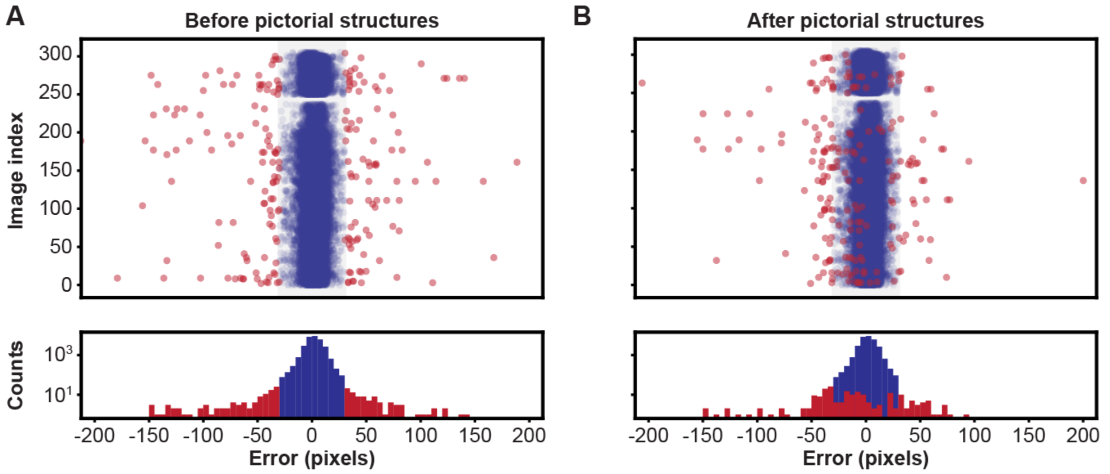


Figure 26 – Pose estimation accuracy before and after using pictorial structures. Shown are pixel-wise 2D pose errors/residuals (top) and their respective distributions (bottom) **(A)** before, or **(B)** after applying pictorial structures. Residuals larger than 35 pixels (red circles) represent incorrect keypoint detections. Those below this threshold (blue circles) represent correct keypoint detections.

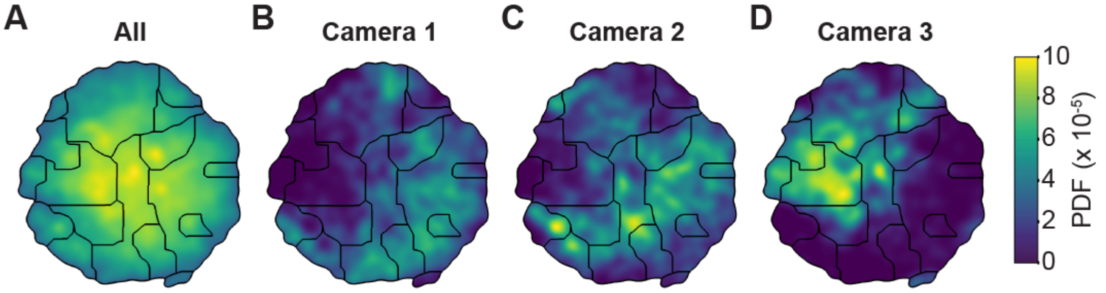


Figure 27 – Unsupervised behavioral classification of 2D pose data is sensitive to viewing angle. **(A)** Behavioral map derived using 2D pose data from three adjacent cameras (Cameras 1, 2, and 3) but the same animals and experimental time points. Shown are clusters (black outlines) with enriched (yellow), or sparsely (blue) populated data. Different clusters are enriched for data from either **(B)** camera 1, **(C)** camera 2, or **(D)** camera 3. Behavioral embeddings were derived using 1 million frames during 4 s of optogenetic stimulation of MDN>CsChrimson (n=6 flies, n=29 trials), aDN>CsChrimson (n=6 flies, n=30 trials), and wild-type control animals (MDN-GAL4/+: n=4 flies, n=20 trials. aDN-GAL4/+: n=4 flies, n=23 trials).

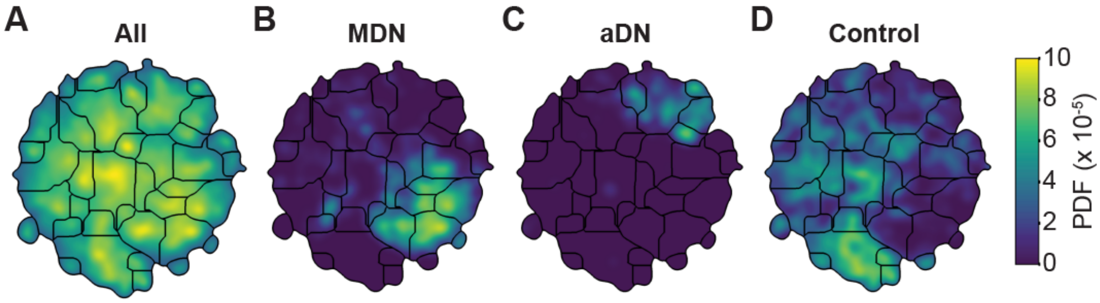


Figure 28 – Unsupervised behavioral classification of 3D joint angle data. Behavioral embeddings were calculated using 3D joint angles from the same 1 million frames used in Figure 3. **(A)** Behavioral map combining all data during 4 s of optogenetic stimulation of MDN>CsChrimson (n=6 flies, n=29 trials), aDN>CsChrimson (n=6 flies, n=30 trials), and wild-type control animals (For MDN-Gal4/+, n=4 flies, n=20 trials. For aDN-Gal4/+ n=4 flies, n=23 trials). The same behavioral map is shown with only the data from **(B)** MDN>CsChrimson stimulation, **(C)** aDN>CsChrimson stimulation, or **(D)** control animal stimulation. Associated videos reveal that these distinct map regions are enriched for backward walking, antennal grooming, and forward walking, respectively.
The following video supplements are available:

Single-view 3D pose estimaton Part 5

Adam Gosztolai*, Semih Günel*, Victor Lobato Rios, Marco Pietro Abrate, Daniel Morales, Helge Rhodin, Pascal Fua, Pavan Ramdya; LiftPose3D, a deep learning-based approach for transforming 2D to 3D pose in laboratory animals, Nature Methods, 2021.

I worked on conceptualization, methodology, software, formal analysis, data curation, writing—original draft, writing—review and editing.

5.1 Introduction

To identify how actions arise from neural circuit dynamics, one must first make accurate measurements of behavior in laboratory experiments. Recent innovations in 3-dimensional (3D) pose estimation promise to accelerate the discovery of these neural control principles. 3D pose estimation is typically accomplished by triangulating 2-dimensional (2D) poses acquired using multiple, synchronized cameras and deep learning-based tracking algorithms [9, 14, 108, 111, 124, 174, 195–197]. Triangulation requires that every tracked keypoint (body landmark) be visible from at least two synchronized cameras [198] and that each camera is calibrated. These requirements are often difficult to meet in space-constrained experimental systems that also house sensory stimulation devices [30, 31, 199], or when imaging untethered, freely behaving animals like fur-covered rodents [200] for whom keypoints can sometimes be occluded.

Because of these challenges, most animal studies have favored 2D pose estimation using one camera [108, 124, 167, 174, 201]. Nevertheless, 3D poses are desirable because they eliminate a problematic camera-angle dependence that can arise during behavioral analyses [111]. Recent work on human pose estimation has performed “lifting" of 2D poses by regressing them to a library of 3D poses [202–205]. However, high accuracy has only recently been achieved using deep learning [10, 104, 206–216]. These techniques have not been adapted to the study of animals due to the lack of large and diverse training datasets.

Here, we introduce LiftPose3D, a tool for 3D pose estimation of tethered and freely behaving laboratory animals from a single camera view. Our method builds on a neural network architecture designed to lift human poses [10]. We develop data transformations and network training augmentation methods that enable accurate 3D pose estimation across a wide range of animals, camera angles, experimental systems, and behaviors using relatively little data. We applied LiftPose3D to a series of use cases to demonstrate that (i) a library of 3D poses can be used to train our network to lift 2D poses from one camera, with minimal constraints on camera hardware and positioning and, consequently, no calibration, (ii) by aligning animal poses, our network can overcome occlusions and outliers in ground truth data, and (iii) pretrained networks can generalize across experimental setups using linear domain adaptation.

5.2 Method

5.2.1 Theoretical basis for LiftPose3D

LiftPose3D estimates the 3D pose $\mathbf{X} = (\mathbf{X}_1, \ldots, \mathbf{X}_n)$, i.e., an ensemble of keypoints, by learning a nonlinear mapping between triangulated ground truth 3D poses and corresponding 2D poses $\mathbf{x}_c = (\mathbf{x}_{c,1}, \ldots, \mathbf{x}_{c,n})$. Formally, this operation is encoded in a *lifting* function f mapping a 2D pose from any camera c to their corresponding 3D pose in camera-centered coordinates, $\mathbf{Y}_c = f(\mathbf{x}_c)$, and a camera transformation ϕ_c, encoding a rotation and translation operation (see Eq. (16)), mapping from camera-centered coordinates to world coordinates $\mathbf{X} = \phi_c^{-1}(\mathbf{Y}_c)$. The lifting function f can be approximated using a deep neural network $F(\mathbf{x}_c; \Theta)$, where Θ represents the network weights controlling the behavior of F. In a specific application, Θ are trained by minimizing the discrepancy between 3D poses predicted by lifting from any camera and ground truth 3D poses,

$$\mathcal{J}_1(\Theta) := \sum_c \sum_{j=1}^{n} \chi_{V_c}(j) ||(F(\mathbf{x}_c; \Theta))_j - \mathbf{Y}_{c,j}||_2^2, \tag{15}$$

where $\chi_{V_c}(j)$ is an indicator function of the set V_c of visible points from camera c. For $F(\mathbf{x}_c; \Theta)$, we adapted a network architecture from [10] composed of fully connected layers regularized by batch-norm and dropout [217] and linked with skip connections (**Figure 29B**). This network was developed to perform human-pose estimation following training on approximately 10^6 fully annotated 2D-3D human pose pairs for many different behaviors.

Our approach implicitly assumes that the network learns two operations: lifting the 2D pose $\mathbf{x}_c$ to camera-centered 3D coordinates $\mathbf{Y}_c$ by predicting the depth component of the pose, and learning perspective effects encoded in the animal-to-camera distance and the intrinsic camera matrix (see Eqs. (16)–(19)). The intrinsic camera matrix is camera-specific, suggesting that a trained network can only lift poses from cameras used during training and that application to new settings with strong perspective effects (short focal lengths) may require camera calibration. We show that this is not necessarily the case and that one can generalize pre-trained networks to new settings by weakening perspective effects. This can be accomplished by either using a large focal length camera, or by increasing the animal-to-camera distance and normalizing the scale of 2D poses.

5.2.2 Obtaining 3D pose ground truth data by triangulation

Triangulated 3D positions served as ground truth data for assessing the accuracy of LiftPose3D. If a keypoint j of interest is visible from at least two cameras, with corresponding 2D coordinates $\mathbf{x}_{c,j} \in \mathbb{R}^2$ in camera c and camera parameters (extrinsic and intrinsic matrices), then its 3D coordinates $\mathbf{X}_j \in \mathbb{R}^3$ in a global world reference frame can be obtained by triangulation. Let us express $\mathbf{X}_j = (x_j^1, x_j^2, x_j^3)$ in homogeneous coordinates as $\widehat{\mathbf{X}}_j = (x_j^1, x_j^2, x_j^3, 1)$. The projection from the 3D points in the global coordinate system to 2D points in a local coordinate system centered on camera c is performed by the function $\pi_c : \mathbb{R}^4 \to \mathbb{R}^3$ defined as $\widehat{\mathbf{x}}_{c,j} = \pi_c(\widehat{\mathbf{X}}_j)$. This function can be expressed as a composition $\pi_c = \mathrm{proj}_{1,2} \circ \phi_c$ of an affine transformation $\phi_c : \mathbb{R}^4 \to \mathbb{R}^4$ from global coordinates to camera-centered coordinates and a projection $\mathrm{proj}_{1,2}: \mathbb{R}^4 \to \mathbb{R}^3$ to the first two coordinates. Both functions can be parametrized using the pinhole camera model [198]. On the one hand, we have

$$\phi_c(\mathbf{X}_j) := \mathbf{C}_c \widehat{\mathbf{X}}_j^T = \widehat{\mathbf{Y}}_{c,j}\,, \tag{16}$$

where $\mathbf{C}_c$ is the extrinsic camera matrix corresponding to the ϕ_c and can be written as

$$\mathbf{C}_c = \left(\begin{array}{c|c} \mathbf{R}_c & \mathbf{T}_c \\ \hline 0 & 1 \end{array}\right) \tag{17}$$

where $\mathbf{R}_c \in \mathbb{R}^{3\times3}$ is a matrix corresponding to rotation around the origin and $\mathbf{T}_c \in \mathbb{R}^3$ is a translation vector representing the distance of the origin of the world coordinate system to the camera center. Likewise, the projection function can be expressed as

$$\mathrm{proj}_{1,2}\widehat{\mathbf{Y}}_{c,j} := \mathbf{K}\widehat{\mathbf{Y}}_{c,j} = \widehat{\mathbf{x}}_{c,j}\,, \tag{18}$$

where $\mathbf{K}$ is the intrinsic camera transformation

$$\mathbf{K} = \begin{pmatrix} f_x & 0 & c_x & 0 \\ 0 & f_y & c_y & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix}, \tag{19}$$

where f_x, f_y denote the focal lengths and c_x, c_y denote the image center. The coordinates projected to the camera plane can be obtained by converting back to Euclidean coordinates $\mathbf{x}_{c,j} = (\widehat{\mathbf{x}}^1_{c,j}/\widehat{\mathbf{x}}^3_{c,j}, \widehat{\mathbf{x}}^2_{c,j}/\widehat{\mathbf{x}}^3_{c,j})$.

Triangulation of the coordinate $\mathbf{X}_j$ of joint j with respect to π_c is obtained by minimizing the reprojection error, that is, the discrepancy between the 2D camera coordinate, $\mathbf{x}_{c,j}$, and the 3D coordinate projected to the camera frame, $\pi_c(\mathbf{X}_j)$. Let V_c be the set of visible joints from camera c. The reprojection error for joint j is taken to be

$$e_{\mathsf{RP}}\left(j; \{\pi_c\}\right) = \sum_c \chi_{V_c}(j)\,||\mathbf{x}_{c,j} - \pi_c(\mathbf{X}_j)||_2^2\,, \tag{20}$$

where $\chi_{V_c}(\cdot)$ is the indicator function of set V_c of visible keypoints from camera c. The camera projection functions π_c are initially unknown. To avoid having to use a calibration grid, we jointly minimize with respect to the 3D location of all joints and to the camera parameters, a procedure known as bundle adjustment [198]. Given a set of 2D observations, we seek

$$\min_{\pi_c, \mathbf{X}_j} \sum_j e_{\mathsf{RP}}\left(j; \{\pi_c\}\right). \tag{21}$$

using a second-order optimization method. For further details, we refer the interested reader to [111].

5.2.3 LiftPose3D network architecture and optimization

The core LiftPose3D network architecture is similar to the one of [10] and is depicted in **Figure 29B**. Its main module includes two linear layers of dimension 1024 rectified linear units (ReLU [218]), dropout [217] and residual connections [219]. The inputs and outputs of each block are connected during each forward pass using a skip connection. The model contains 4×10^6 trainable parameters, which are optimized by stochastic gradient descent using the Adam optimizer [220]. We also perform batch normalization [221].

In all cases, the parameters were set using Kaiming initialization [219] and the optimizer was run until convergence—typically within 30 epochs—with the following training hyperparameters: Batch-size of 64 and an initial learning rate of 10^{-3} that was dropped by 4% every 5000 steps. We implemented our network in PyTorch on a desktop workstation running on an Intel Core i9-7900X CPU with 32 GB of DDR4 RAM, and a GeForce RTX 2080 Ti Dual O11G GPU. Training time was less than 10 minutes for all cases studied.

5.2.4 Weak perspective augmentation

To project 2D poses from 3D poses, one needs to know the camera transformation ϕ_c (Eq. (16)), encoded by the extrinsic matix $\mathbf{C}_c$ (Eq. (17)) and the projection function $\text{proj}_{1,2}$ (Eq. (18)), encoded by the intrinsic matrix $\mathbf{K}$ (Eq. (19)). $\mathbf{K}$ may be unknown *a priori* at test time. Alternatively, one may want to use one of our pre-trained networks on a novel dataset without having to match the camera positioning (focal length, camera-to-animal distance) used to collect the training data. In this case, one may still be able to predict the 3D pose in a fixed camera-centered coordinate frame by assuming that either the camera-to-animal distance or the focal length are large enough to neglect perspective effects and by normalizing the scale of 2D poses. Following Ref. [6], we chose the Frobenius norm to perform normalization on the input 2D poses $\mathbf{x}_{c,j}/||\mathbf{x}_{c,j}||_F$, which is the diagonal distance of the smallest bounding box around the 2D pose. Note, that if the 2D poses are obtained via projections, one may use the unit intrinsic matrix Eq. (19) with $f_x = f_y$ and $c_x = c_y = 0$ before performing normalization. Here, using $c_x = c_y = 0$ assumes that the 2D poses are centered, which in each of our examples is achieved by considering coordinates relative to root joints placed at the origin. Importantly, the 2D poses must be normalized both at training and test times.

5.2.5 Camera-angle augmentation

The object-to-camera orientation is encoded by the extrinsic matrix $\mathbf{C}_c$ of Eq. (17). When it is unavailable, one can still use our framework by taking 3D poses from the ground truth library and, during training, performing virtual 2D projections around the approximate camera location or for all possible angles. To this end, we assume that the rotation matrix $\mathbf{R}$ is unknown, but that the intrinsic matrix $\mathbf{K}$ and the object-to-camera distance d are known such that we may take $\mathbf{T} = (0, 0, d)^T$. When $\mathbf{K}$ or d are also unknown, or dynamically changing, one can make the weak-perspective assumption as described in the next section. Then, instead of training the LiftPose3D network with pairs of 3D poses and 2D poses at fixed angles, we perform random 2D projections of the 3D pose to obtain virtual camera planes whose centers c_x, c_y lie on the sphere of radius d. To define the projections we require a parametric representation of the rotations. Rotating a point in 3D space can be achieved using three consecutive rotations around the three Cartesian coordinate axes x, y, z commonly referred to as Euler angles and denoted by ψ_x,ψ_y, and ψ_y. The rotation matrix can then be written as

$$\begin{aligned}\mathbf{R} = \mathbf{R}_{xyz} &= \mathbf{R}_x(\psi_x)\mathbf{R}_y(\psi_y)\mathbf{R}_z(\psi_z)\\ &= \begin{pmatrix}1 & 0 & 0\\ 0 & \cos\psi_x & -\sin\psi_x\\ 0 & \sin\psi_x & \cos\psi_x\end{pmatrix}\begin{pmatrix}\cos\psi_y & 0 & \sin\psi_y\\ 0 & 1 & 0\\ -\sin\psi_y & 0 & \cos\psi_y\end{pmatrix}\begin{pmatrix}\cos\psi_z & -\sin\psi_z & 0\\ \sin\psi_z & \cos\psi_z & 0\\ 0 & 0 & 1\end{pmatrix}.\end{aligned} \tag{22}$$

Given Eq. (16)–(19) we may then define a random projection $\widehat{\mathbf{x}}_j$ on the sphere of radius d of a keypoint with homogeneous coordinate $\widehat{\mathbf{X}}_j$ as

$$\widehat{\mathbf{x}}_j = \mathbf{K}\left(\begin{array}{c|c}\mathbf{R}_{xyz} & \mathbf{T}\\ \hline 0 & 1\end{array}\right)\widehat{\mathbf{X}}_j \tag{23}$$

where $\mathbf{T} = (0, 0, d)^T$. Likewise, the 3D pose in camera coordinates can be expressed as

$$\widehat{\mathbf{Y}}_j = \left(\begin{array}{c|c}\mathbf{R}_{xyz} & \mathbf{T}\\ \hline 0 & 1\end{array}\right)\widehat{\mathbf{X}}_j. \tag{24}$$

Before training, we fix d, f_x, f_y, c_y, c_y and define intervals for the Euler angle rotations. We then obtain the mean and standard deviation in each dimension for both 2D and 3D poses in the training dataset by performing random projections within these angle ranges. The obtained means and standard deviations are then used to normalize both the training and test datasets.

5.2.6 Linear domain adaptation

Here we describe the process of adapting a network trained on data from experiment A to lift 2D poses in experiment B. Domain adaptation is also useful if the camera parameters or the distance from the camera are not known and the weak perspective assumption cannot be invoked. Before performing domain adaptation, we first estimate 2D poses from ventral images in domain B, as before. This allowed us to circumvent the difficulties arising from differences in appearance and illumination that are present in the more general image domain adaptation problem [16, 116]. Thus, adapting poses became a purely geometric problem of adjusting proportions and postural differences across domains.

The basis for domain adaptation is to first find a function $d_2 : B|_2 \to A|_2$, where $A|_2$ and $B|_2$ are restrictions of 3D poses in the two domains, to the corresponding $2n$-dimensional spaces of 2D poses. This function maps poses in domain B to domain A and makes them compatible inputs for the network trained on poses in domain A. In the scenario that 3D data is available in domain B, we can also find a function $d_3 : B \to A$ where A and B are $3n$-dimensional spaces of 3D poses in the two experimental domains. After 3D poses have been obtained in domain A, we map these poses back to domain B by inverting this function.

We now describe how to obtain the functions d_2 and d_3, which we denote collectively as d. To find d, we assume that poses in domain B can be obtained by small perturbations of poses in domain A. This allows us to set up a matching between the two domains by finding nearest neighbor 2D poses in domain A for each 2D pose in domain B, $\mathbf{x}_i^B = (\mathbf{x}_{i,1}^B, \dots, \mathbf{x}_{i,n}^B)$. We use 2D rather than 3D poses to find a match because 3D poses may not always be available in domain B. Moreover, the nearest poses in 3D space will necessarily be among the nearest poses in 2D space. Specifically, for each $\mathbf{x}_i^B$, we find a set of k nearest poses in domain A, $\{\mathcal{N}(\mathbf{x}_i^B)_j\}_{j=1}^k$ such that $||\mathcal{N}(\mathbf{x}_i^B)_j - \mathbf{x}_i^B||_2 < ||\mathcal{N}(\mathbf{x}_i^B)_{j+1} - \mathbf{x}_i^B||_2$. We then use these poses to learn a linear mapping $\mathbf{W}_{BA} \in \mathbb{R}^{2n \times 2n}$ from domain B to A, where n is the number of keypoints, as before. We can find this linear mapping by first defining a set of p training poses in domain B, $\mathbf{x}_{\mathrm{tr}}^B = \mathbf{x}_1^B, \dots \mathbf{x}_p^B$ and writing $\mathbf{W}_{BA}\mathbf{x}_{\mathrm{tr}}^B = \mathbf{x}_{\mathrm{tr}}^A$, where $\mathbf{x}_{\mathrm{tr}}^B \in \mathbb{R}^{dn \times kp}$ and $\mathbf{x}_{\mathrm{tr}}^A \in \mathbb{R}^{dn \times kp}$ with $d = 2$ or 3 are matrices defined according to

$$\mathbf{W}_{BA}\left(\underbrace{\begin{matrix} | & & | \\ \mathbf{x}_1^B & \cdots & \mathbf{x}_1^B \\ | & & | \end{matrix}}_{k} \quad \cdots \quad \underbrace{\begin{matrix} | & & | \\ \mathbf{x}_p^B & \cdots & \mathbf{x}_p^B \\ | & & | \end{matrix}}_{k}\right) =$$

$$\left(\underbrace{\begin{matrix} | & & | \\ \mathcal{N}(\mathbf{x}_1^B)_1 & \cdots & \mathcal{N}(\mathbf{x}_1^B)_k \\ | & & | \end{matrix}}_{k} \quad \cdots \quad \underbrace{\begin{matrix} | & & | \\ \mathcal{N}(\mathbf{x}_p^B)_1 & \cdots & \mathcal{N}(\mathbf{x}_p^B)_k \\ | & & | \end{matrix}}_{k}\right). \tag{25}$$

Transposing this linear equation yields the linear problem $(\mathbf{x}_{\mathrm{tr}}^B)^T \mathbf{W}_{BA}^T = (\mathbf{x}_{\mathrm{tr}}^A)^T$. Given that the p training poses are different, $\mathbf{x}_{\mathrm{tr}}^B$ has linearly independent columns and this problem is overdetermined as long as $kp > dn$. Thus, by least-squares minimization, we obtain $\mathbf{W}_{BA}^T = ((\mathbf{x}_{\mathrm{tr}}^B)^T \mathbf{x}_{\mathrm{tr}}^B)^{-1} (\mathbf{x}_{\mathrm{tr}}^B)^T (\mathbf{x}_{\mathrm{tr}}^A)^T$.

5.2.7 Experimental systems and conditions

All adult *Drosophila melanogaster* experiments were performed on female flies raised at 25°C on a 12 h light/dark cycle at 2-3 days post-eclosion (dpe). Before each experiment, wild-type (*PR*) animals were anaesthetized using CO_2 or in ice-cooled vials and left to acclimate for 10 min. DeepFly3D tethered fly data were taken from [111]. OpenMonkeyStudio macaque data were taken from [14]. LocoMouse mouse data were taken from [200]. CAPTURE rat data were taken from [15]. FlyLimbTracker freely-behaving fly data were taken from [167]. See these publications for detailed experimental procedures. For more information on the datasets including the number of keypoints, poses, animals, resolution, framerate we refer the reader to **Supplementary Table 2**.

Freely behaving *Drosophila* recorded from two high-resolution views using one camera and a right-angle prism mirror

We constructed a transparent arena coupled to a right-angle prism mirror [222, 223]. The enclosed arena consists of three vertically stacked layers of 1/16” thick acrylic sheets laser-cut to be 15 mm long, 3 mm wide, and 1.6 mm high. The arena ceiling and walls were coated with Sigmacote (Sigma-Aldrich, Merck, Darmstadt, Germany) to discourage animals from climbing onto the walls and ceilings. One side of the enclosure was physically coupled to a right-angled prism (Thorlabs PS915). The arena and prism were placed on a kinematic mounting platform (Thorlabs KM100B/M), permitting their 3D adjustment with respect to a camera (Basler acA1920-150um) outfitted with a lens (Computar MLM3X-MP, Cary, NC USA). Data were acquired using the Basler Pylon software (pylon Application 1.2.0.8206, pylon Viewer 6.2.0.8206). The camera was oriented vertically upwards below the arena to provide two views of the fly: a direct ventral view, and an indirect, prism mirror-reflected side view. The arena was illuminated by four Infrared LEDs (Thorlabs, fibre-coupled LED M850F2 with driver LEDD1B T-Cube and collimator F810SMA-780): two from above and two from below. To elicit locomotor activity, the platform was acoustically and mechanically stimulated using a mobile phone speaker. Flies were then allowed to behave freely, without optogenetic stimulation.

Freely behaving *Drosophila* recorded from one ventral view at low-resolution

We constructed a square arena consisting of three vertically stacked layers of 1/16" thick acrylic sheets laser-cut to be 30 mm long, 30 mm wide, and 1.6 mm high. This arena can house multiple flies at once, increasing throughput at the expense of spatial resolution (26 px mm^{-1}). Before each experiment the arena ceiling was coated with 10 uL Sigmacote (Sigma-Aldrich, Merck, Darmstadt, Germany) to discourage animals from climbing onto the ceiling. A camera (pco.panda 4.2 M-USB-PCO, Gloor Instruments, Switzerland, with a Milvus 2/100M ZF.2 lens, Zeiss, Switzerland) was oriented with respect to a 45 ° mirror below the arena to capture a ventral view of the fly. An 850 nm infrared LED ring light (CCS Inc. LDR2-74IR2-850-LA) was placed above the arena to provide illumination. Although the experiment contained optogenetically elicited behaviors interspersed with periods of spontaneous behavior, here we focused only on spontaneously generated forward walking.

The positions and orientations of individual flies were tracked using custom software including a modified version of Tracktor [224]. Using these data, a 138 × 138 px image was cropped around each fly and registered for subsequent analyses.

5.2.8 2D pose estimation

In the prism-mirror setup, we split the data acquired from a single camera into ventral and side view images. We hand-annotated the location of all 30 leg joints (five joints per leg) on 640 images from the ventral view and up to 15 visible unilateral joints on 640 images of the side view. We used these manual annotations to train two separate DeepLabCut [174] 2D pose estimation networks (root-mean-squared errors for training and testing were 0.02 mm and 0.04 mm for ventral and side views, respectively). We ignored frames in which flies were climbing the enclosure walls (thus exhibiting large yaw and roll orientation angles). We also removed keypoints with < 0.95 DeepLabCut confidence and higher than 10 px mismatch along the x-coordinate of ventral and side views. FlyLimbTracker data [167] was manually annotated. Images acquired in the new low-resolution ventral view setup were annotated using DeepLabCut [174] trained on 160 hand-annotated images. Due to the low resolution of images, the coxa-femur joints were not distinguishable. Therefore, we treated the thorax-coxa and coxa-femur joints as a single entity.

5.2.9 Training the LiftPose3D network

An important step in constructing LiftPose3D training data is to choose r root joints (see the specific use cases below for how these root joints were selected), and a target set corresponding to each root joint. The location of joints in the target set are predicted relative to the root joint to ensure translation invariance of the 2D poses.

The training dataset consisted of input-output pose pairs ($\mathbf{x}_c^{\mathrm{tr}}, \mathbf{X}^{\mathrm{tr}}$) with dimensionality equal to the number of keypoints visible from a given camera c minus the number of root joints r, namely $\mathbf{x}_c^{\mathrm{tr}} \in \mathbb{R}^{2(|V_c|-r)}$ and $\mathbf{X}^{\mathrm{tr}} \in \mathbb{R}^{3(|V_c|-r)}$. Then, the training data was standardized with respect to the mean and standard deviation of a given keypoint across all poses.

Tethered *Drosophila melanogaster*

Of the 38 original keypoints in Ref. [111], here we focused on the 30 leg joints. Specifically, for each leg we estimated 3D position for the thorax-coxa, coxa-femur, femur-tibia, and tibia-tarsus joints and the tarsal tips (claws). Thus, the training data consisted of input-output coordinate pairs for 24 joints (30 minus six thorax-coxa root joints) from all cameras. The training convergence is shown on **Extended Data Figure 2A**).

Freely behaving macaque monkeys

The OpenMonkeyStudio dataset [14] consists of images of freely behaving macaques inside a $2.45 \times 2.45 \times 2.75$ m arena in which 62 cameras are equidistant horizontally at two heights along the arena perimeter. We extracted all five available experiments (7, 9, 9a, 9b and 11) for training and testing. Since 2D pose annotations were not available for all cameras, we augmented this dataset during training by projecting triangulated 3D poses onto cameras lacking 2D annotation using the provided camera matrix. We removed fisheye lens-related distortions of 2D poses using the provided radial distortion parameters. We normalized each 2D pose to unit length, by dividing it by its Euclidean norm as well as the 3D pose with respect to bone lengths to reduce the large scale variability of the OpenMonkeyStudio annotations (animals ranged between 5.5 and 12 kg). We set the neck as the root joint during training. We compare our absolute errors to the total body length, calculated as the sum of the mean lengths of the nose-neck, neck-hip, hip-knee, knee-foot joints pairs. Over multiple epochs, we observed rapid convergence of our trained network (**Extended Data Figure 2B**).

Freely behaving mice and *Drosophila* recorded from two views using a right-angle mirror

Freely behaving mouse data [200] consisted of recordings of animals traversing a 66.5 cm long, 4.5 cm wide, and 20 cm high glass corridor. A 45° mirror was used to obtain both ventral and side views with a single camera beneath the corridor. 2D keypoint positions were previously tracked using the LocoMouse software [200]. We considered six major keypoints—the four paws, the proximal tail, and the nose. Keypoint positions were taken relative to a virtual root keypoint placed on the ground midway between the nose and the tail. The networks were trained on partial ground truth data following pose alignment, as described in the main text. The networks for *Drosophila* and mouse training data converged within 30 and 10 training epochs (**Extended Data Figure 2C,D**).

Freely behaving rat in a naturalistic enclosure

The CAPTURE dataset contains recordings of freely behaving rats in a 2-foot diameter cylindrical enclosure video recorded using six cameras. Motion capture markers on the animal were tracked using a commercial motion capture acquisition program [15] to obtain 2D poses. Out of 20 possible joints, we limited our scope to the 15 joints that were not redundant and provided most of the information about the animal's pose. The dataset includes 4 experiments recording 3 rats from two different camera setups. Before using LiftPose3D, we removed the distortion from 2D poses using radial distortion parameters provided by the authors. The CAPTURE dataset has many missing 3D pose instances which we handle by not computing the loss corresponding to these keypoints during back-propagation. We selected the neck joint as the single root joint and predicted all of the other joints with respect to this root joint. We observed that LiftPose3D converged within 15 training epochs (**Extended Data Figure 2E**).

Freely behaving adult *Drosophila melanogaster* recorded from one ventral camera view

For both the newly acquired low-resolution and previously published high-resolution [167] images of freely behaving flies taken using one ventral view camera, we trained a LiftPose3D network on partial ground truth data acquired from the prism mirror system. For the high-resolution data, we considered the thorax-coxa joints as roots. For the low resolution data, the coxa-femur joints were imperceptible. Therefore, the thorax-coxa joints were selected as roots. The training dataset consisted of coordinate pairs ($\mathbf{x}^{\mathsf{tr}}_{\mathsf{ventral}} + \eta, \mathbf{z}^{\mathsf{tr}}_{\mathsf{side}}$) where $\mathbf{x}^{\mathsf{tr}}_{\mathsf{ventral}}$, $\mathbf{z}^{\mathsf{tr}}_{\mathsf{side}}$ were chosen to represent the annotated ventral coordinates and z-axis depth for the visible joints, as before. Meanwhile, η was a zero-mean Gaussian noise term with a joint-independent standard deviation of 4 px. The role of this noise term was to account for the keypoint position degeneracy inherent in the transformation from high-resolution prism training data to lower-resolution testing data. For the high resolution dataset this noise term was set to zero.

5.2.10 Comparing joint angles derived from lifted 3D and 2D poses

To illustrate the benefits of using lifted 3D coordinates versus 2D coordinates for kinematic analyis, we derived the joint angles obtained from 3D coordinates along with projected 2D coordinates. Consider the (2D or 3D) coordinates of three consecutive joints in the kinematic chain of one leg with coordinates $\mathbf{u}$, $\mathbf{v}$, $\mathbf{w}$. Then, vectors $\mathbf{s}_1 = \mathbf{u}-\mathbf{v}$ and $\mathbf{s}_2 = \mathbf{u}-\mathbf{w}$ describe adjacent bones. Their enclosed angle is found by the cosine rule, $\cos^{-1}(\mathbf{s}_1 \cdot \mathbf{s}_2/(||\mathbf{s}_1||\,||\mathbf{s}_2||))$. Due to the uncertainty of 2D and 3D pose estimation, we assumed that keypoint coordinates are Gaussian distributed around the estimated coordinate. As a proxy for the variance we took the variation of bone lengths $||\mathbf{s}_1||$ and $||\mathbf{s}_2||$ because they are expected to remain approximately constant owing to the low mechanical compliance of the fly's exoskeleton (with the exception of the flexible tarsal segments). This allowed us to predict 3D joint angles by Monte Carlo sampling (using 5×10^3 samples), drawing one sample from each of three distributions and then computing the corresponding joint angle by the cosine rule.

5.3 Evaluation

5.3.1 Predicting 3D poses with a single camera at an arbitrary position

Rather than taking independent 2D keypoints as inputs, as for triangulation-based 3D pose estimation, LiftPose3D uses a deep-neural network to regress an ensemble of 2D keypoints viewed from a camera—the 2D pose—to a ground truth library of 3D poses. Considering all keypoints simultaneously allows the network to learn geometric relationships intrinsic to animal poses.

First, we illustrate how this approach can reduce the number of cameras needed for 3D pose estimation on a tethered adult *Drosophila* dataset [111]. Here, 15 keypoints are visible from three synchronized cameras on each side of the animal (**Figure 29A**). These keypoints were annotated and triangulated using DeepFly3D [111]. Using this dataset as a 3D pose library we trained a LiftPose3D network that lifts half-body 2D poses from any side camera without knowing the camera's orientation **Figure 29B,C**). First, we ensured that the output of LiftPose3D was translation invariant by predicting the keypoints of the respective legs relative to six "root" immobile thorax-coxa joints (green circles, **Figure 29B,C**). Second, to avoid the network having to learn perspective distortion, we assumed that the focal length (intrinsic matrix) of the camera and the animal-to-camera distance were known, or that one of them is large enough to assume weak perspective effects. In the latter case, we normalized 2D input poses by their Frobenius norm during both training and testing. Third, to facilitate lifting from any angle, we assumed that extrinsic matrices, which could be obtained by calibration, might also be unknown. Instead, we parametrized them by Euler angles ψ_z, ψ_y, ψ_x representing ordered rotations around the z, y and x axes of a coordinate system centered around the fly (**Figure 29D**). During training, we took as inputs 2D poses (from 3D poses randomly projected to virtual camera planes, rather than 2D pose estimates), and as outputs 3D poses triangulated from three cameras. To measure lifting accuracy, we tested the network on software-annotated 2D poses (**Figure 29B**) from two independent animals and computed the mean absolute error (MAE), e_j^{te}, for each joint j as well as the MAE across all joints $e^{\mathrm{te}} = (1/n)\sum_j e_j^{\mathrm{te}}$ relative to triangulated 3D poses.

We found that LiftPose3D could predict 3D poses using only one camera per side (**Figure 29C**). When the virtual projections during training were performed using known intrinsic and extrinsic matrices, the network's accuracy was at least as good as triangulation using two cameras per keypoint (**Figure 29E**). The accuracy did not suffer when the network was trained using virtual 2D projections around an approximate camera location (**Figure 29E**) rather than with known instrinsic matrices, or using normalized 2D poses rather than with known intrinsic matrices. Accuracy remained excellent when virtual projections extended to all possible angles around the meridian (**Figure 29E**). Lifting could be performed for optogenetically induced backward walking (**Supplementary Video 1**), antennal grooming (**Supplementary Video 2**), and spontaneous, irregular limb movements (**Supplementary Video 3**). Because the network predicts joint coordinates with respect to thoracic root joints, the MAE was larger for distal joints that move within a larger kinematic volume. By contrast, the error for triangulation depended only on the accuracy of 2D annotations because it treats each keypoint independently. We also assessed camera-angle dependence for our wide angle-range network by lifting virtual 2D poses projected onto the meridian of the unit sphere, or 2D poses captured from each of the six cameras (**Figure 29F**). The test MAE was low (< 0.05 mm) and had no camera-angle dependence. Because we make no assumptions about camera placement when training our angle-invariant networks, these pretrained networks might also be used to predict accurate 3D poses for tethered *Drosophila* recorded in other laboratories.

We next explored how the similarity between animal behaviors used for training and testing might influence lifting accuracy. Our tethered *Drosophila* dataset contained optogenetically induced antennal grooming (*aDN*), and backward walking (*MDN*), as well as spontaneous behaviors like forward walking (control). We trained a network using poses from only one behavior (not including rest frames) and evaluated it on all three behaviors while keeping the amount of training data fixed (2.5×10^4 poses). As expected, the MAE was higher when test data included untrained behaviors than when test data included trained behaviors (**Figure 29G**). Furthermore, training on all three behaviors led to comparable or lower MAE (**Figure 29E**) than training and testing on one single behavior (**Figure 29G**). Thus, higher training data diversity improves lifting accuracy.

To illustrate the advantage of using lifted 3D poses versus 2D poses in downstream analyses, we derived joint angles during forward walking from lifted 3D poses and from 2D poses projected from 3D poses in the ventral plane (**Extended Data Figure 1**). Joint angles derived from lifted and triangulated 3D poses were in close agreement. On the other hand, when viewed from a projected plane, we found spurious dynamics in the distal joints likely due to rotations upstream in the kinematic chain (proximal joints) that cause movements of the whole leg. Thus, 3D poses predicted by LiftPose3D can help to decouple underlying physical degrees-of-freedom.

We also tested LiftPose3D in freely behaving animals where the effective camera angle dynamically changes, and in animals without exoskeletons whose neighboring keypoints are less constrained. Specifically, we considered freely behaving macaque monkeys [14] where 3D poses were triangulated using 2D poses from 62 synchronized cameras (**Figure 29H**). After training LiftPose3D with only 6,571 3D poses, we could lift 3D poses from test images with diverse animal poses (**Supplementary Video 4**), acquired from any camera (**Figure 29I**), and with relatively low body length-normalized MAE (**Figure 29J**).

Taken together, these results demonstrate that, using simple data preprocessing and a relatively small but diverse training dataset, LiftPose3D can reduce the number of cameras required to perform accurate 3D pose estimation.

5.3.2 Predicting 3D poses despite occluded keypoints

In freely behaving animals, keypoints are often missing from certain camera angles due to self-occlusions and, therefore, only partial 3D ground truth can be obtained by triangulation. We asked how the global nature of lifting—all keypoints are lifted simultaneously—might be leveraged to reconstruct information lost by occlusions, allowing one to predict full 3D poses.

To address this question, we built an experimental system similar to others used for flies and mice [200, 222, 223] that consisted of a transparent enclosure coupled to a right-angle prism mirror and with a camera beneath to record ventral and side views of a freely behaving fly (**Figure 30A**). Due to the right-angle prism and the long focal length camera (i.e., negligible perspective effects), the ventral and side views are orthographic projections of the true 3D pose. Triangulation thus consisted of estimating the z-axis depth of keypoints from the side view. Although keypoints closer to the prism were simultaneously visible in both views and could be triangulated, other joints had only ventral 2D information. We therefore aligned flies in the same reference frame in the ventral view (**Figure 30B**), turning lifting into a regression problem similar to that for tethered animals. During training we took ventral view 2D poses as inputs, but trained only those keypoints with complete 3D information, i.e., those having both ventral and side views (**Figure 30C**). By also aligning these data, we found that the network training converged despite the unseen coordinates (**Extended Data Figure 2**), which were implicitly augmented during training by learning geometric relationships between keypoints. The network could predict 3D positions for every joint at test time, including those occluded in the side view

(**Figure 30D** and **Supplementary Video 5**). Notably, owing to the high spatial resolution of this setup, the accuracy, based on available triangulation-derived 3D positions (**Figure 30E**), was better than that obtained for tethered flies triangulated using four cameras (**Figure 29E**). Thus, LiftPose3D can estimate 3D poses from 2D images in cases where keypoints are occluded and cannot be triangulated.

These results suggested an opportunity to apply lifting to potentially correct inaccurate 3D poses obtained using other tracking approaches. To test this, we used a dataset consisting of freely behaving mice traversing a narrow corridor [200] and tracked using the LocoMouse software from ventral and side views [200]. We triangulated and aligned incomplete 3D ground truth poses as we did for *Drosophila* and then trained a LiftPose3D network using ventral 2D poses as inputs. Predictions were in good agreement with the LocoMouse's side view tracking (**Figure 30E** and **Supplementary Video 6**) and could recover expected cycloid-like kinematics between strides (**Figure 30F**). LiftPose3D predictions could also correct poorly labeled or missing side-view poses (**Figure 30F**). However, lifting accuracy depended on the fidelity of input 2D poses: incorrect ventral 2D poses generated false side view predictions (**Figure 30F**). These errors were always localized to a single joint and were relatively infrequent. Overall, LiftPose3D and LocoMouse performed similarly compared with manual human annotation (**Figure 30G**) demonstrating that LiftPose3D can be used to test the consistency of ground truth datasets.

To assess how well spatial relationships learned by LiftPose3D could generalize to animals with more complex behaviors and larger variations in body proportions, we next considered the CAPTURE dataset which was taken using six cameras that recorded freely behaving rats within a circular arena [15] (**Figure 30H**). Animal joints were intermittently self-occluded during a variety of complex behaviors (**Figure 30I**). Therefore, to allow the network to learn the skeletal geometry, we aligned animals in the camera-coordinate frame and replaced missing input data with zeros. Furthermore, to make the network robust to bone length variability within and across animals (**Figure 30J**) we assumed that bone lengths were normally distributed and generated, for each triangulated 3D pose, rescaled 3D poses by sampling from bone-length distributions while preserving joint angles. Then, we obtained corresponding 2D poses via a virtual projection within the Euler angle range of $\pm 10°$ with respect to the known camera locations (to augment the range of camera-to-animal angles). Finally, we normalized 2D poses by their Frobenius norm, as before, assuming a large enough camera-to-animal distance.

To show that the network generalizes across new experimental setups, we used two experiments from this dataset (i.e., two animals and two camera arrangements) for training and tested with a third experiment (a different animal, camera focal length, and animal-to-camera distance). By replacing low confidence or missing coordinates with zeros, LiftPose3D could accurately predict the nonzero coordinates (**Figure 30H, K** and **Supplementary Video 7**). Thus, this is a viable way to correct for erroneous input keypoints and makes our network directly applicable to other rat movement studies.

5.3.3 Lifting diverse experimental data without 3D ground truth

Although our angle-invariant networks for lifting 3D poses in tethered flies (**Figure 29D-F**) and freely behaving rats (**Figure 30**H-K) can be used in similar experimental systems without the need for additional training data, small variations resulting from camera distortions or postural differences may limit the accuracy of lifted poses. Therefore, we explored how domain adaptation might enable pretrained networks to lift poses in new experimental systems despite small postural variations.

We assessed the possibility of domain adaptation by training a network in domain A—tethered flies—and predicting 3D poses in domain B—freely-moving flies (**Figure 31A**). To do so, we identified two linear transformations d_2 and d_3. d_2 is used to map 2D poses from domain B as inputs to the pre-trained network in domain A, while d_3^{-1} is used to transform lifted 3D poses back to domain B. These linear transformations were found as best-fit mappings from every pose in a training dataset *B'* to their k nearest neighbors *A'* (**Figure 31B**). They are expected to generalize as long as the poses in domain A are rich enough to cover the pose repertoire in domain B and are sufficiently similar between domains. We found by 10-fold cross-validation that the error associated with the transformations converged after training with less than 500 poses (**Figure 31C**). The final lifted poses were also in good agreement with the triangulated poses in domain B (**Figure 31D**) having accuracies comparable to a network lifting purely in domain A (**Figure 31E**).

To demonstrate the full potential of linear domain adaptation, we next lifted *Drosophila* 2D poses from a single ventral camera. This experimental system is common due to its simplicity, low cost, and increased throughput and has been used to study *C. elegans* [225], larval zebrafish [226], larval *Drosophila* [227], adult *Drosophila* [228], and mice [229]. Because depth sensors [230, 231] cannot resolve small laboratory animals, 3D pose estimation from a single 2D view remains unsolved, but has the potential to enrich behavioral datasets and improve downstream analysis.

We developed an experimental system with a square-shaped arena in which multiple freely-behaving flies were recorded ventrally using a single camera (**Figure 31F**, left) at four-fold lower spatial resolution (26 px mm^{-1}) than in our prism-mirror system. We pretrained a network using prism-mirror training data for keypoints present in both datasets and then augmented these data using a Gaussian noise term with standard deviation of ~ 4. We adapted annotated 2D poses into the network's domain before lifting (**Figure 31B**). We found that the network could predict physiologically realistic 3D poses in this dataset using only ventral 2D poses (**Figure 31G** and **Supplementary Video 8**). This is remarkable because ventrally-viewed swing and stance phases are difficult to distinguish, particularly at lower resolution. During walking, 2D tracking of the tarsal claws traced out stereotypical trajectories in the x-y plane [166] and circular movements in the unmeasured x-z plane (**Figure 31H**). The amplitudes of these movements were consistent with real kinematic measurements during forward walking [232].

Another possibility offered by LiftPose3D is to 'resurrect' previously published 2D pose data for 3D kinematic analyses. We applied our network that was trained on prism-mirror data to lift video data of a fly walking through a capsule-shaped arena [167] (**Figure 31I**). Using a similar processing pipeline as before (**Figure 31B,F,G**), including registration and domain adaptation but not noise perturbations (the target data were of similarly high resolution as the training data), LiftPose3D could predict 3D poses from this dataset (**Figure 31J**). We again observed physiologically realistic cyclical movements of the pretarsi during forward walking (**Figure 31K**, bottom; **Supplementary Video 9**). These data illustrate that linear domain-adaptation and LiftPose3D can be combined to lift 3D poses from previously published 2D video data for which 3D triangulation would be otherwise impossible.

5.3.4 *Drosophila* LiftPose3D station

These domain adaptation results suggested that one could make 3D pose acquisition more accessible by designing a "*Drosophila* LiftPose3D station"—an open-source hardware system including a 3D printed rig supporting a rectangular arena (**Extended Data Figure 3**, **Supplementary Note 1**). A common hardware solution like this overcomes potential variability across different experimental systems that arise from camera distortions and perspective effects. Using pre-trained DeepLabCut and LiftPose3D networks we found that one can effectively lift *Drosophila* 3D poses with this system (**Supplementary Video 10**). We envision that a similar approach might, in the future, also facilitate cross-laboratory 3D lifting of mouse 2D poses from a single camera.

5.4 Conclusion

Here we have introduced LiftPose3D, a deep learning-based tool that simplifies 3D pose estimation across a wide variety of laboratory contexts. LiftPose3D can take as inputs 2D poses from any of a variety of annotation softwares[111, 174]. Through input data preprocessing, training augmentation, and domain adaptation one can train a lifting network [10] with several orders of magnitude less data as well as incomplete or innacurate ground truth poses. LiftPose3D is invariant to camera hardware and positioning, making it possible to use the same networks across laboratories and experimental systems. We provide an intuitive Python notebook that serves as an interface for data preprocessing, network training, 3D predictions, and data visualization.

Several factors must be considered when optimizing LiftPose3D for new experimental systems. First, because predicting depth from a 2D projection depends on comparing the lengths of body parts, input poses must be sufficiently well-resolved to discriminate between 3D poses with similar 2D projections. Second, prediction accuracy depends on training data diversity: previously untrained behaviors may not be as accurately lifted. Further work may improve LiftPose3D by constraining 3D poses using body priors [114, 233–236] and temporal information [213].

Using our domain adaptation methodology, networks with the largest and most diverse training data, like those for the tethered fly, may be sufficiently robust to accurately lift 2D to 3D pose in other laboratories. In the future, similarly robust lifting networks might be generated for other animals through a cross-laboratory aggregation of diverse 3D pose ground truth datasets. In summary, LiftPose3D can accelerate 3D pose estimation by reducing the need for complex and expensive synchronized multi-camera systems, and arduous calibration procedures. This enables the acquisition of rich behavioral data and can accelerate our understanding of the neuromechanical control of behavior.

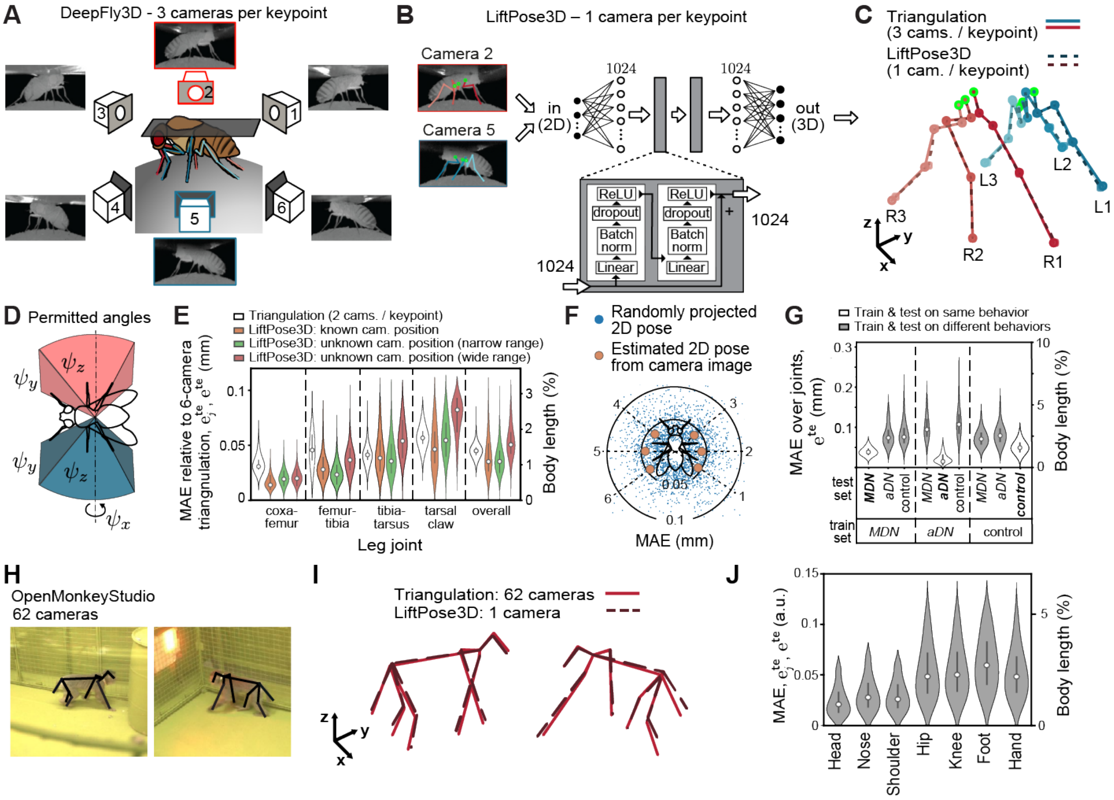


Figure 29 – **LiftPose3D predicts 3D poses with a single, flexibly positioned camera** **A** Ground truth 3D poses of tethered *Drosophila* are triangulated using six camera views (3 cameras per keypoint). **B** LiftPose3D predicts 3D poses using deep network-derived 2D poses from only two cameras (red and blue, 1 camera per keypoint). The coordinates are considered relative to a set of root joints (green). The inputs are scaled up and passed twice through the main processing unit (gray rectangle) comprising batch norm, dropout and ReLU wrapped by a skip connection. **C** The output, 3D half-body poses (blue/red), are compared with triangulated 3D poses. Limbs are labeled by left (L)/right (R) and front (1), mid (2), or hind (3) positions. **D** LiftPose3D can be trained using virtual camera projections of 3D poses to lift from cameras within the angles ψ_z, ψ_y, ψ_x (representing ordered yaw, roll, pitch rotations). **E** Error of 3D poses relative to triangulation using three cameras per keypoint. We compare triangulation error using 2 cameras per keypoint (white), test error for a network trained with known camera parameters (orange) and two angle-invariant networks with narrow (green, $\psi_z = \pm 10°$, $\psi_y = \pm 5°$, $\psi_x = \pm 5°$ with respect to a known camera orientation), or wide ranges (red, $\psi_z = \pm 180°$, $\psi_y = \pm 5°$, $\psi_x = \pm 5°$). **F** Error of lifted 3D poses at different virtual camera orientations of the wide-range lifter network and a network with known camera parameters. Blue dots represent lifting errors for a given projected 2D pose. Orange circles represent averages over the test dataset for a given camera. **G** Error of estimated 3D poses for a network trained and tested on different combination of behavioral data including optogenetically induced backward walking (*MDN*, left), antennal grooming (*aDN*, middle), or spontaneous, unstimulated behaviors (*control*, right). **H** Two representative images from the OpenMonkeyStudio dataset. 2D poses are superimposed (black). **I** 3D poses obtained by triangulating up to 62 cameras (red lines), or using a single camera and LiftPose3D (dashed black lines). **J** Absolute errors for different body parts with respect to total body length. Violin plots represent Gaussian kernel density estimates with bandwidth 0.5, truncated at the 99th percentile and superimposed with the median (gray dot), 25th, and 50th percentiles (black line).

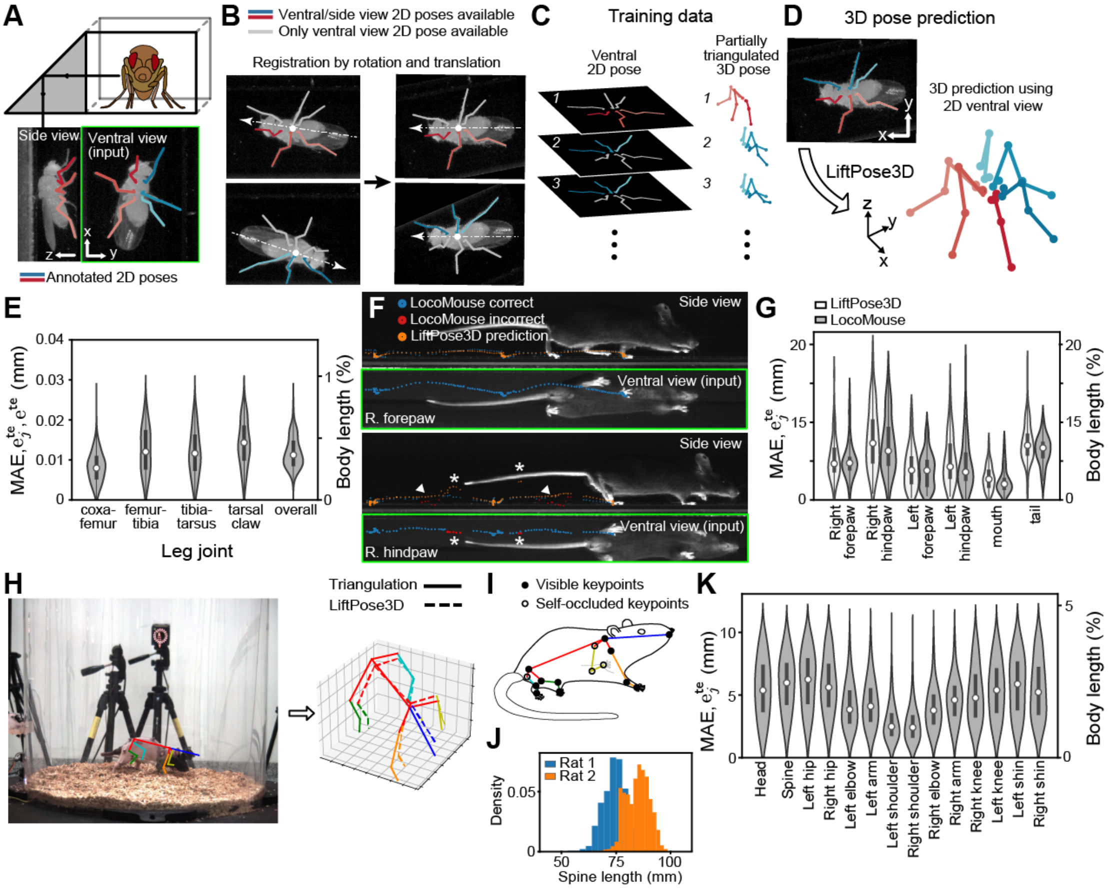


Figure 30 – **LiftPose3D performs 3D poses estimation on freely behaving animals with occluded keypoints.** **A** *Drosophila* behaving freely within a narrow, transparent enclosure. Using a right-angle prism mirror, ventral (top) and side (bottom) views are recorded with one camera. Afterwards, 2D poses are annotated (colored lines). Ventral 2D poses (green box) are used to lift 3D poses. **B** Keypoints near the prism mirror (red and blue) can be tracked in both views and triangulated. Other keypoints (gray) are only visible ventrally and thus have no 3D ground truth. Unilateral ground truth for both sides are obtained by registering the orientation and position of ventral images of the fly. **C** Training data consist of full ventral 2D poses and their corresponding partial 3D poses. **D** Following training, LiftPose3D can predict 3D poses for new ventral view 2D poses. **E** Joint-wise and overall absolute errors of the network's 3D pose predictions for freely behaving *Drosophila*. **F** A similar data preprocessing approach is used to lift ventral view 2D poses of mice (green boxes) walking within a narrow enclosure and tracked using LocoMouse software. LocoMouse ground truth (blue and red) and LiftPose3D (orange) pose trajectories are shown for the right forepaw (top) and hindpaw (bottom) during one walking epoch. Arrowheads indicate where LiftPose3D lifting of the ventral view can be used to correct LocoMouse side view tracking errors (red). Asterisks indicate where inaccuracies in the LocoMouse ventral view ground truth (red) disrupt LiftPose3D's side view predictions (orange). **G** Absolute errors of LiftPose3D and LocoMouse side view predictions for six keypoints with respect to manually annotated ground truth data. **H** Camera image from the CAPTURE dataset superimposed with the annotated 2D pose (left). LiftPose3D uses this 2D pose to recover the full 3D pose (right). **I** LiftPose3D can be trained to lift 3D poses of a freely moving rat with occluded keypoints (open circles). **J** Histograms of the measured lengths of the spinal segment for two different animals. **K** Error distribution over all keypoints for the CAPTURE dataset.

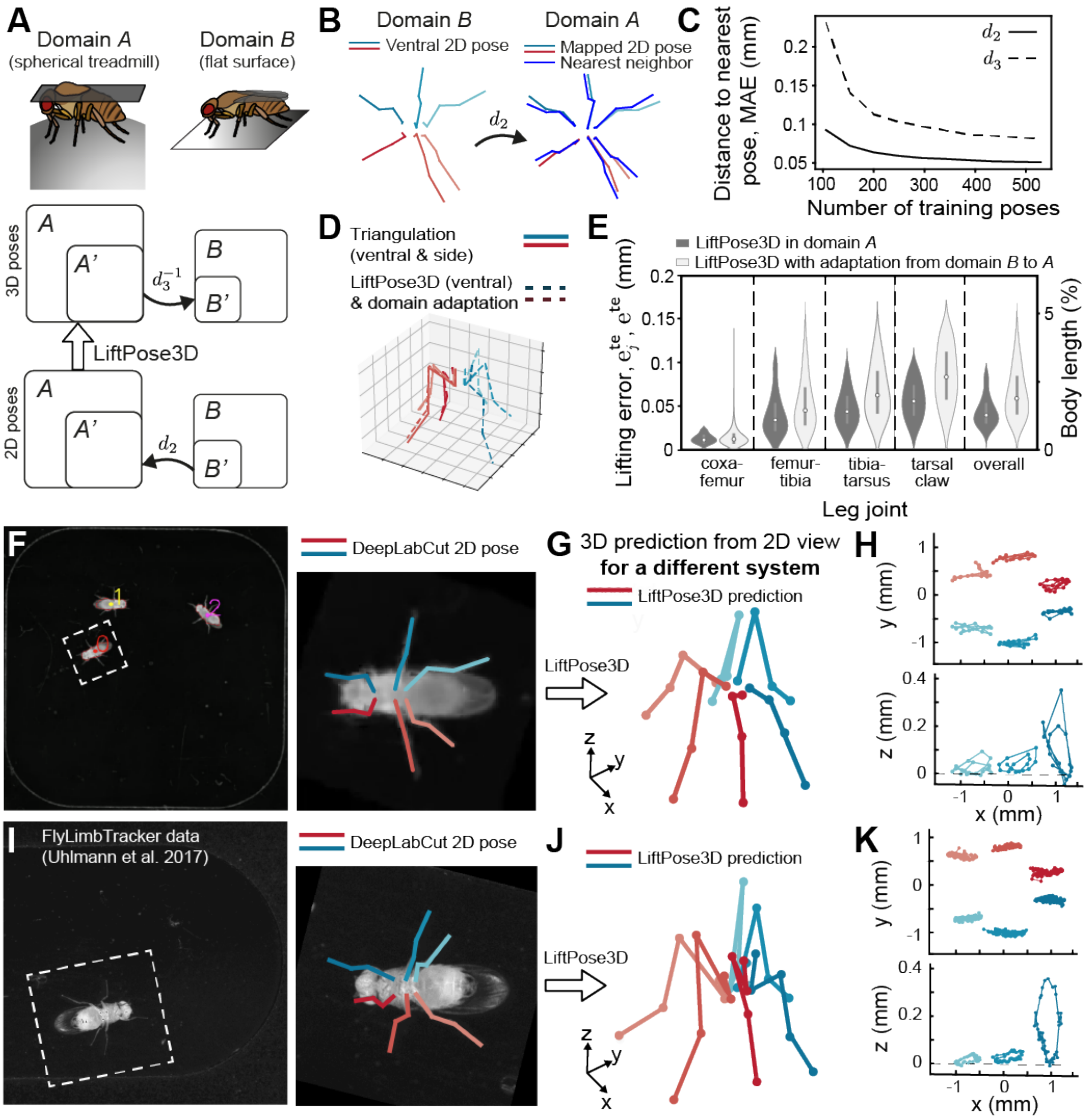


Figure 31 – **A pretrained LiftPose3D network predicts 3D poses for diverse data and when triangulation is impossible.** **A** Linear domain adaptation between domain A (fly on a spherical treadmill) and domain B (fly on a flat surface). 2D poses in B are mapped to A by a linear transformation d_2 then lifted with a network trained only on domain A poses. After lifting, the 3D poses are mapped back to B by another linear transformation d_3. **B** A typical 2D pose in domain B mapped into domain A by the best-fit linear transformation d_2 between poses in B and their nearest neighbors in A. **C** Error between mapped pose and k nearest neighbor poses for d_2, d_3 against the number of poses used to train them ($k = 1$ for d_2 and $k = 2$ for d_3). **D** Lifted 3D pose following domain adapation of a ventral domain B 2D pose and lifting with a network trained on domain A data. The prediction is superimposed with the incomplete ground truth 3D pose in domain B. **E** Lifting error following domain adaptation of domain B poses compared with lifting error in the domain A with no domain adaptation. **F** Freely behaving flies were recorded from below using a low-resolution camera. Following body tracking, the region-of-interest containing the fly was cropped and registered. 2D pose estimation was then performed for 24 visible joints. **G** 2D poses are adapted to the prism-mirror domain. These are then lifted to 3D poses using a network pre-trained with prism-mirror data and coarse-grained to match the lower resolution 2D images in the new experimental system. **H** These 3D poses permit the analysis of claw movements in an otherwise unobserved $x-z$ plane (bottom). **I** Freely behaving fly recorded from below using one high-resolution camera. 2D pose estimation was performed for all 30 joints. Following tracking, a region-of-interest containing the fly was cropped and registered. The same LiftPose3D network trained in panel B—but without coarse-graining—was used to predict **J** 3D poses and **K** unobserved claw movements in the $x-z$ plane (bottom).

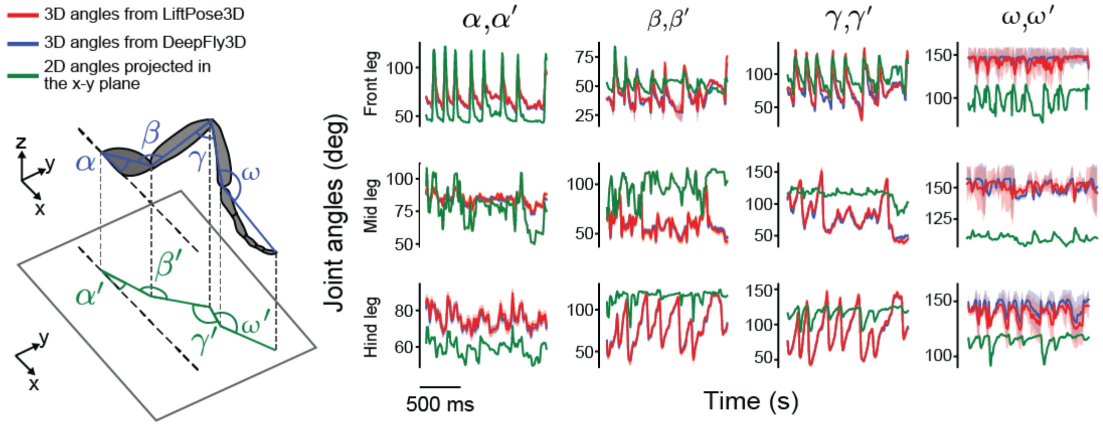


Figure 32 – **Joint angles resulting from lifting compared with 3D triangulated ground truth and 2D projections.** Joint angles α, β, γ, and ω for the front, mid, and hind left legs during forward walking. Shown are angles computed from 3D triangulation using DeepFly3D (blue), LiftPose3D predictions (red), and ventral 2D projections α', β', γ', and ω' (green). The mean (solid lines) and standard deviation of joint error distributions (transparency) are shown. Joint angles were computed by Monte Carlo sampling and errors were computed by taking the fluctuation in bone lengths.

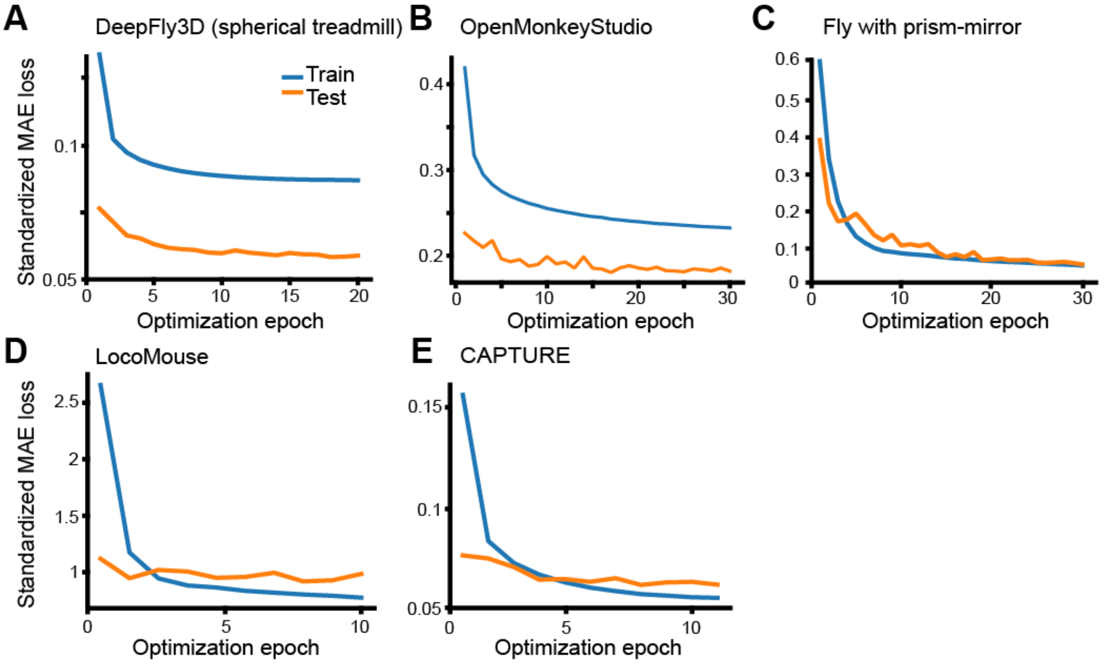


Figure 33 – **Training and test loss convergence of the LiftPose3D network applied to a variety of datasets.** Shown are the absolute test errors of LiftPose3D for all joints as a function of optimization epoch. Note that the test error is sometimes lower than the training error because we do not apply dropout at test time. **A** Two-camera data of *Drosophila* on a spherical treadmill (each color denotes a different pair of diametrically opposed cameras). **B** OpenMonkeyStudio dataset (each color denotes a different training run). **C** Single-camera data of *Drosophila* behaving freely in the right-angle prism mirror system. **D** LocoMouse dataset. **E** CAPTURE dataset.

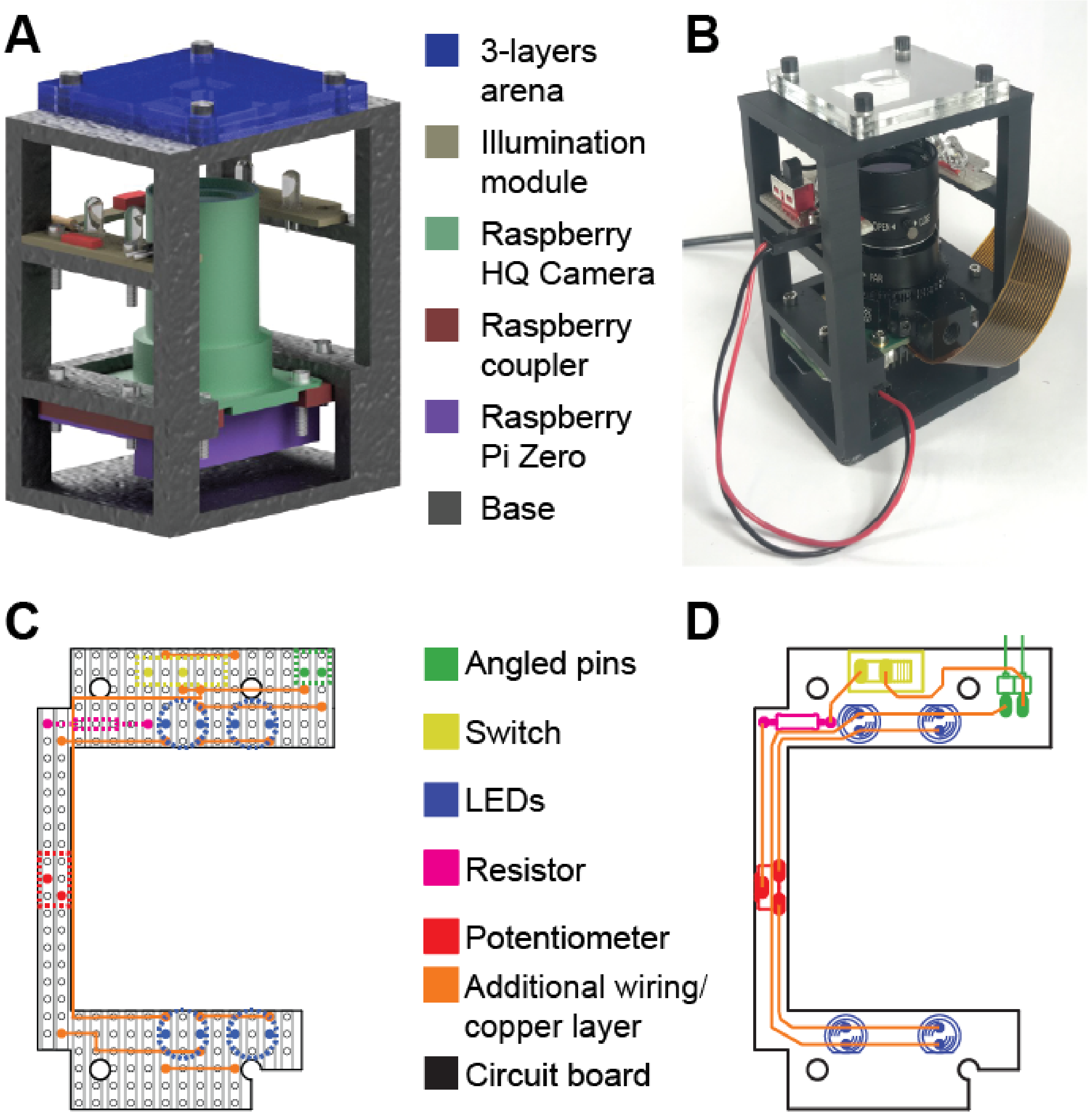


Figure 34 – ***Drosophila*** **LiftPose3D station. A** CAD drawing of the LiftPose3D station indicating major components (color-coded). **B** Photo of the LiftPose3D station. **C** Electronic circuit for building the illumination module on a pre-fabricated prototyping board, electronic components, and additional wiring are indicated (color-coded). **D** Printed circuit board provided as an alternative to the pre-fabricated board for building the illumination module.

Part 6

Learning Neural Action Representations

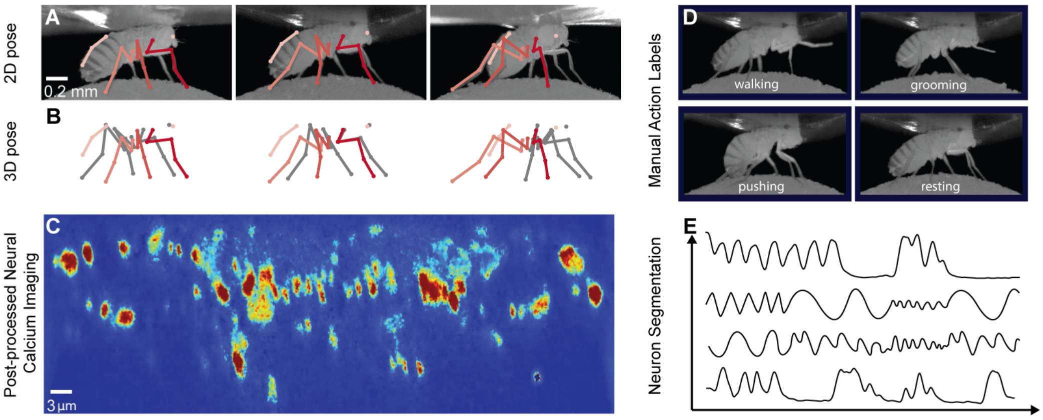


Figure 35 – **Our Motion Capture and Two-Photon (MC2P) Dataset.** A tethered fly (*Drosophila melanogaster*) is recorded using six multi-view infrared cameras and a two-photon microscope. The resulting dataset includes the following. **(A)** 2D poses extracted from all views (only three are shown), calculated on grayscale images. **(B)** 3D poses triangulated from the 2D views. **(C)** Synchronized, registered, and denoised single-channel fluorescence calcium imaging data using a two-photon microscope. Shown are color-coded activity patterns for populations of descending neurons from the brain. These carry action information (red is active, blue is inactive). **(D)** Annotations for eight animals of eight different behaviors, four of which are shown here. **(E)** Manual neural segmentation has been performed to extract neural activity traces for each neuron. We will release our MC2P publicly.

Semih Günel, Florian Aymanns, Sina Honari, Pavan Ramdya, Pascal Fua; Overcoming the Domain Gap in Contrastive Learning of Neural Action Representations. arXiv, 2022.

6.1 Introduction

neural decoding of actions, the accurate prediction of animal behavior from brain activity, is a fundamental challenge in neuroscience with important applications in the development of robust brain machine interfaces. Recent technological advances have enabled simultaneous recordings of neural activity and behavioral data in experimental animals and humans [30–36]. Nevertheless, our understanding of the complex relationship between behavior and neural activity remains limited.

A major reason is that it is difficult to obtain many long recordings from mammals and a few subjects are typically not enough to perform meaningful analyses [237]. This is less of a problem when studying the fly *Drosophila melanogaster*, for which long neural and behavioral datasets can be obtained for many individual animals **(Fig. 35)**. Nevertheless, current supervised approaches for performing neural action decoding [3, 238] still do not generalize well across animals because each nervous system is unique **(Fig. 36A)**. This creates a significant domain-gap that necessitates tedious and difficult manual labeling of actions. Furthermore, a different model must be trained for each individual animal, requiring more annotation and overwhelming the resources of most laboratories.

Another problem is that experimental neural imaging data often has unique temporal and spatial properties. The slow decay time of fluorescence signals introduces temporal artifacts. Thus, neural imaging frames include information about an animal's previous behavioral state. This complicates decoding and requires specific handling that standard machine learning algorithms do not provide.

To address these challenges, we propose to learn neural action representations—embeddings of behavioral states within neural activity patterns—in an unsupervised fashion. To this end, we leverage the recent development of computer vision approaches for automated, markerless 3D pose estimation [18, 111] to provide the required supervisory signals without human intervention. We first show that using contrastive learning to generate latent vectors by maximizing the mutual information of simultaneously recorded neural and behavioral data modalities is not sufficient to overcome the domain gap between animals and to generalize to unlabeled animals at test time **(Fig. 36C)**. To address this problem, we introduce two sets of techniques:

1. To close the domain gap between animals, we leverage 3D pose information. Specifically, we use pose data to find sequences of similar actions between a source and multiple target animals. Given these sequences, we input to our model neural data from one animal and behavioral data composed of multiple animals. This allows us to train our decoder to ignore animal identity and close the domain gap.
2. To mitigate the slowly decaying calcium data impact from past actions on neural images, we add simulated randomized versions of this effect to our training neural images in the form of a temporally exponentially decaying random action. This trains our decoder to learn the necessary invariance and to ignore the real decay in neural calcium imaging data. Similarly, to make the neural encoders robust to imaging noise resulting from low image spatial resolution, we mix random sequences into sequences of neural data to replicate this noise.

The combination of these techniques allowed us to bridge the domain gap across animals in an unsupervised manner, making it possible to perform action recognition on unlabeled animals **(Fig. 36E)** better than earlier techniques, including those requiring supervision [3, 58, 76].

To test the generalization capacity of neural action decoding algorithms, we record and use MC2P dataset, which we will make publicly available. It includes two-photon microscope recordings of multiple spontaneously behaving *Drosophila*, and associated behavioral data together with action labels.

Finally, to demonstrate that our technique generalizes beyond this one dataset, we tested it on two additional ones: One that features neural ECoG recordings and 2D pose data for epileptic patients [239, 240] along with the well-known H36M dataset [29] in which we treat the multiple views as independent domains. Our method markedly improves across-subject action recognition in all datasets.

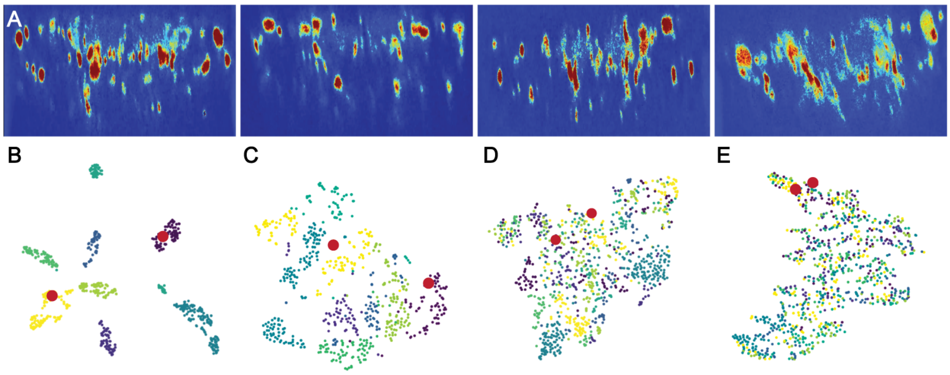


Figure 36 – **Domain gap across nervous systems and learned representations. (Top row) (A)** Neural imaging data from four different animals in our MC2P dataset. Images differ in terms of total brightness, the location and number of visible neurons, and the shape and size of axons. **(Bottom row)** t-SNE embeddings of different neural representation algorithms on MC2P dataset. Each color denotes a different fly. The two red dots are embeddings of the same action label for two different animals. t-SNE embeddings of **(B)** Raw neural data. **(C)** Contrastive SimCLR [5] representation trained on behavioral and neural pairs, **(D)** SimCLR and domain adaptation using a two-layer MLP discriminator and a Gradient Reversal Layer [241]. **(E)** Ours. The identify of the animals is discarded and the semantic structure is preserved, as evidenced by the fact that the two red dots are very close to one another.

We hope our work will inspire the use and development of more general unsupervised neural feature extraction algorithms in neuroscience. These approaches promise to accelerate our understanding of how neural dynamics give rise to complex animal behaviors and can enable more robust neural action decoding algorithms to be used in brain-machine interfaces.

6.2 Method

Our goal is to be able to interpret neural data such that, given a neural image, one can generate latent representations that are useful for downstream tasks. This is challenging, as there is a wide domain-gap in neural representations for different animals **(Fig. 36A)**. We aimed to leverage unsupervised learning techniques to obtain rich features that, once trained, could be used on downstream tasks including action recognition to predict the behaviors of unlabeled animals.

Our data is composed of two-photon microscopy neural images synchronized with 3D behavioral data, where we do not know where each action starts and ends. We leveraged contrastive learning to generate latent vectors from both modalities such that their mutual information would be maximized and therefore describe the same underlying action. However, this is insufficient to address the domain-gap between animals **(Fig. 36C)**. To address this issue, we perform swapping augmentation: we replace the original pose or neural data of an animal with one from another set of animals for which there is a high degree of 3D pose similarity at each given instance in time.

Unlike behavioral data, neural data has unique properties. Neural calcium data contains information about previous actions because it decays slowly across time and it involves limited spatial resolution. To teach our model the invariance of these artifacts of neural data, we propose two data augmentation techniques: (i) Neural Calcium augmentation - given a sequence of neural data, we apply an exponentially decaying neural snapshot to the sequence, which imitates the decaying impact of previous actions, (ii) Neural Mix augmentation - to make the model more robust to noise, we applied a mixing augmentation which merges a sequence of neural data with another randomly sampled neural sequence using a mixing coefficient.

Together, these augmentations enable a self-supervised approach to (i) bridge the domain gap between animals allowing testing on unlabeled ones, and (ii) imitate the temporal and spatial properties of neural data, diversifying it and making it more robust to noise. In the following section, we describe steps in more detail.

6.2.1 Problem Definition

We assume a paired set of data $\mathscr{D}_s = \left\{\left(\mathbf{b}_\mathbf{i}^s, \mathbf{n}_\mathbf{i}^s\right)\right\}_{i=1}^{n_s}$, where $\mathbf{b}_\mathbf{i}^s$ and $\mathbf{n}_\mathbf{i}^s$ represent behavioral and neural information respectively, with n_s being the number of samples for animal $s \in \mathscr{S}$. We quantify behavioral information $\mathbf{b}_\mathbf{i}^s$ as a set of 3D poses $\mathbf{b}_k^s$ for each frame $k \in \mathbf{i}$ taken of animal s, and neural information $\mathbf{n}_\mathbf{i}^s$ as a set of two-photon microscope images $\mathbf{n}_k^s$, for all frames $k \in \mathbf{i}$ capturing the activity of neurons. The data is captured such that the two modalities are always synchronized (paired) without human intervention, and therefore describe the same set of events. Our goal is to learn an unsupervised parameterized image encoder function f_n, that maps a set of neural images $\mathbf{n}_\mathbf{i}^s$ to a low-dimensional representation. We aim for our learned representation to be representative of the underlying action label, while being agnostic to both modality and the identity. We assume that we are not given action labels during unsupervised training. Also note that we do not know at which point in the captured data an action starts and ends. We just have a series of unknown actions performed by different animals.

6.2.2 Contrastive Representation Learning

For each input pair $\left(\mathbf{b}_\mathbf{i}^s, \mathbf{n}_\mathbf{i}^s\right)$, we first draw a random augmented version $(\tilde{\mathbf{b}}_\mathbf{i}^s, \tilde{\mathbf{n}}_\mathbf{i}^s)$ with a sampled transformation function $t_n \sim \mathscr{T}_n$ and $t_b \sim \mathscr{T}_b$, where $\mathscr{T}_n$ and $\mathscr{T}_b$ represent a family of stochastic augmentation functions for behavioral and neural data, respectively, which are described in the following sections. Next, the encoder functions f_b and f_n transform the input data into low-dimensional vectors $\mathbf{h}_b$ and $\mathbf{h}_n$, followed by non-linear projection functions g_b and g_n, which further transform data into the vectors $\mathbf{z}_b$ and $\mathbf{z}_n$. During training, we sample a minibatch of N input pairs $\left(\mathbf{b}_\mathbf{i}^s, \mathbf{n}_\mathbf{i}^s\right)$, and train with the loss function

$$\mathscr{L}_{NCE}^{b\rightarrow n} = -\sum_{i=1}^{N} \log \frac{\exp\left(\left\langle \mathbf{z}_b^i, \mathbf{z}_n^i \right\rangle / \tau\right)}{\sum_{k=1}^{N} \exp\left(\left\langle \mathbf{z}_b^i, \mathbf{z}_n^k \right\rangle / \tau\right)} \tag{26}$$

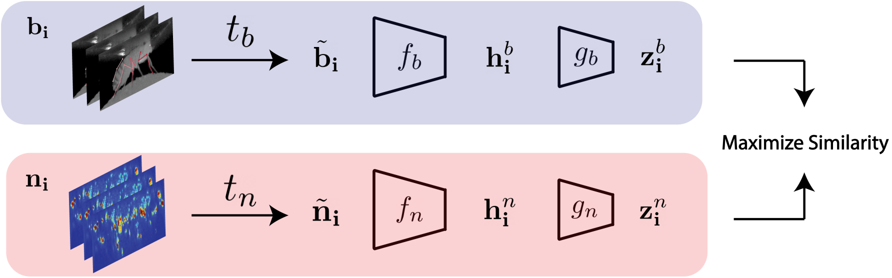

$\mathbf{b_i}$
t_b
$\tilde{\mathbf{b}}_\mathbf{i}$
f_b
$\mathbf{h}_\mathbf{i}^b$
g_b
$\mathbf{z}_\mathbf{i}^b$
Maximize Similarity
$\mathbf{n_i}$
t_n
$\tilde{\mathbf{n}}_\mathbf{i}$
f_n
$\mathbf{h}_\mathbf{i}^n$
g_n
$\mathbf{z}_\mathbf{i}^n$

Figure 37 – **Overview of our contrastive learning-based neural action representation learning approach.** First, we sample a synchronized set of behavioral and neural frames, $(\mathbf{b_i}, \mathbf{n_i})$. Then, we augment these data using randomly sampled augmentation functions t_b and t_n. Encoders f_b and f_n then generate intermediate representations $\mathbf{h}^b$ and $\mathbf{h}^n$, which are then projected into $\mathbf{z}_b$ and $\mathbf{z}_t$ by two separate projection heads g_b and g_n. We maximize the similarity between the two projections using an InfoNCE loss. At test time, the red branch and $\mathbf{h}_\mathbf{i}^n$ is used for neural decoding.

where $\langle \mathbf{z}_b^i, \mathbf{z}_n^i \rangle$ is the cosine similarity between behavioral and neural modalities and $\tau \in \mathbb{R}^+$ is the temperature parameter. An overview of our method for learning f_n is shown in **Fig 37**. Intuitively, the loss function measures classification accuracy of a N-class classifier that tries to predict $\mathbf{z}_n^i$ given the true pair $\mathbf{z}_b^i$. To make the loss function symmetric with respect to the negative samples, we also define

$$\mathscr{L}_{NCE}^{n \to b} = -\sum_{i=1}^{N} \log \frac{\exp\left(\langle \mathbf{z}_b^i, \mathbf{z}_n^i \rangle / \tau\right)}{\sum_{k=1}^{N} \exp\left(\langle \mathbf{z}_b^k, \mathbf{z}_n^i \rangle / \tau\right)}. \tag{27}$$

We take the combined loss function to be $\mathscr{L}_{NCE} = \mathscr{L}_{NCE}^{b \to n} + \mathscr{L}_{NCE}^{n \to b}$, as in [242, 243]. The loss function maximizes the mutual information between two modalities [244]. Although standard contrastive learning bridges the gap between different modalities, it does not bridge the gap between different animals **(Fig. 36C)**. This is a fundamental challenge that we address in this work through augmentations described in the following section, which are part of the neural and behavioral family of augmentations $\mathscr{T}_n$ and $\mathscr{T}_b$.

Swapping Augmentation. Given a set of consecutive 3D poses $\mathbf{b}_\mathbf{i}^s$, for each $k \in \mathbf{i}$, we stochastically replace $\mathbf{b}_k^s$ with one of its nearest pose neighbors in the set of domains $\mathscr{D}_\mathscr{S}$, where $\mathscr{S}$ is the set of all animals. To do this, we first randomly select a domain $\hat{s} \in \mathscr{S}$ and define a probability distribution $\mathbf{P}_{\mathbf{b}_k^s}^{\hat{s}}$ over the domain $\mathscr{D}_{\hat{s}}$ with respect to $\mathbf{b}_k^s$,

$$\mathbf{P}_{\mathbf{b}_k^s}^{\hat{s}}(\mathbf{b}_l^{\hat{s}}) = \frac{\exp(-\|\mathbf{b}_l^{\hat{s}} - \mathbf{b}_k^s\|_2)}{\sum_{\mathbf{b}_m^{\hat{s}} \in \mathscr{D}_{\hat{s}}} \exp(-\|\mathbf{b}_m^{\hat{s}} - \mathbf{b}_k^s\|_2)}. \tag{28}$$

We then replace each 3D pose $\mathbf{b}_k^s$ by first uniformly sampling a new domain $\hat{s}$, and then sampling from the above distribution $\mathbf{P}^{\hat{s}}_{\mathbf{b}^s_k}$, which yields in $\tilde{\mathbf{b}}^s_k \sim \mathbf{P}^{\hat{s}}_{\mathbf{b}^s_k}$. In practice, we calculate the distribution $\mathbf{P}$ only over the first $\mathbf{N}$ nearest neighbors of $\mathbf{b}^s_k$, in order to sample from a distribution of the most similar poses. We empirically set $\mathbf{N}$ to 128. Swapping augmentation reduces the identity information in the behavioral data without perturbing it to the extent that semantic action information is lost. Since each behavioral sample $\mathbf{b}^s_\mathbf{i}$ is composed of a set of 3D poses, and each 3D pose $\mathbf{b}^s_k, \forall k \in \mathbf{i}$ is replaced with a pose of a random domain, the transformed sample $\tilde{\mathbf{b}}^s_\mathbf{i}$ is now composed of multiple domains. This forces the behavioral encoding function f_b to leave identity information out, therefore merging multiple domains in the latent space **(Fig. 38)**.

Swapping augmentation is similar to the synonym replacement augmentation used in natural language processing [245], where randomly selected words in a sentence are replaced by their synonyms, therefore changing the syntactic form of the sentence without altering the semantics. To the best of our knowledge, we are the first to use swapping augmentation in the context of time-series analysis or for domain adaptation.

To keep swapping symmetric, we also swap the neural modality. To swap a set of neural images $\mathbf{n}^s_\mathbf{i}$, we take its behavioral pair $\mathbf{b}^s_\mathbf{i}$, and searched for similar sets of poses in other domains, with the assumption that similar sets of poses describe the same action. Therefore, once similar behavioral data is found, their neural data can be swapped. Note that, unlike behavior swapping, we do not calculate the distribution on individual 3D pose $\mathbf{b}^s_k$, but instead on the whole set of behavioral data $\mathbf{b}^s_\mathbf{i}$, because similarity in a single pose does not necessarily imply similar actions. More formally, given the behavioral-neural pair $\left(\mathbf{b}^s_\mathbf{i}, \mathbf{n}^s_\mathbf{i}\right)$, we swap the neural modality $\mathbf{n}^s_\mathbf{i}$ with $\mathbf{n}^{\hat{s}}_\mathbf{j}$, with the probability

$$\mathbf{P}^{\hat{s}}_{\mathbf{n}^s_\mathbf{i}}(\mathbf{b}^{\hat{s}}_\mathbf{j}) = \frac{\exp(-\|\mathbf{b}^{\hat{s}}_\mathbf{j} - \mathbf{b}^s_\mathbf{i}\|_2)}{\sum_{\mathbf{b}^{\hat{s}}_\mathbf{m} \in \mathcal{D}_{\hat{s}}} \exp(-\|\mathbf{b}^{\hat{s}}_\mathbf{m} - \mathbf{b}^s_\mathbf{i}\|_2)}. \tag{29}$$

This results in a new pair $\left(\mathbf{b}^s_\mathbf{i}, \tilde{\mathbf{n}}^s_\mathbf{i}\right)$, where the augmented neural data comes from a new animal $\hat{s} \in \mathcal{S} / s$.

Neural Calcium Augmentation. Our neural data was obtained using two-photon microscopy and fluorescence calcium imaging. The resulting images are only a function of the underlying neural activity, and have temporal properties that differ from the true neural activity. For example, calcium signals from a neuron change much more slowly than the neuron's actual firing rate. Consequently, a single neural image $\mathbf{n}_t$ includes decaying information concerning neural activity from the recent past, and thus carries information about previous behaviors. This makes it harder to decode the current behavioral state.

We aimed to prevent this overlap of ongoing and previous actions. Specifically, we wanted to teach our network to be invariant with respect to past behavioral information by augmenting the set of possible past actions. To do this, we generated new data $\tilde{\mathbf{n}}^s_\mathbf{i}$, that included previous neural activity $\mathbf{n}^s_k$. To mimic calcium indicator decay dynamics, given a neural data sample $\mathbf{n}^s_\mathbf{i}$ of multiple frames, we sample a new neural frame $\mathbf{n}^s_k$ from the same domain, where $k \notin \mathbf{i}$. We then

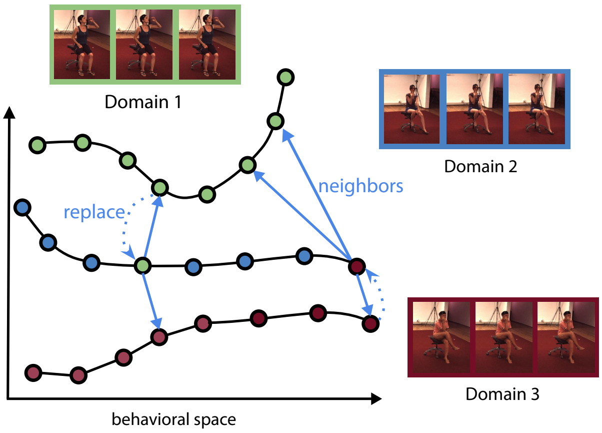


Figure 38 – **Swapping augmentation.** Each 3D pose in the motion sequence of Domain 2 is randomly replaced with its neighbors, from the set of domains $\hat{s} \in \mathcal{S} / s$, which includes Domain 1 and Domain 3. The swapping augmentation hides identity information, while keeping pose changes in the sequence minimal.

convolve $\mathbf{n}_k^s$ with the temporally decaying calcium convolutional kernel $\mathcal{K}$, therefore creating a set of images from a single frame $\mathbf{n}_k^s$, which we then add back to the original data sample $\mathbf{n}_\mathbf{i}^s$. This results in $\tilde{\mathbf{n}}_\mathbf{i}^s = \mathbf{n}_\mathbf{i}^s + \mathcal{K} * \mathbf{n}_k^s$ where $*$ denotes the convolutional operation. In the Supplementary Material, we explain calcium dynamics and our calculation of the kernel $\mathcal{K}$ in more detail.

Neural Mix Augmentation. Two-photon microscopy images often include multiple neural signals combined within a single pixel. This is due to the the fact that multiple axons can be present in a small tissue volume that is below the spatial resolution of the microscope. To mimic this noise-adding effect, given a neural image $\mathbf{n}_\mathbf{i}^s$, we randomly sample a set of frames $\mathbf{n}_\mathbf{k}^{\hat{s}}$, from a random domain $\hat{s}$. We then return the blend of these two videos, $\tilde{\mathbf{n}}_\mathbf{i}^s = \mathbf{n}_\mathbf{i}^s + \alpha \mathbf{n}_\mathbf{k}^{\hat{s}}$, to mix and hide the behavioral information. Unlike the CutMix [246] and Mixup [247] augmentations used for supervised training, we apply the augmentation in an unsupervised setup to make the model more robust to noise. We sample a random α for the entire set of samples in $\mathbf{n}_\mathbf{i}^s$.

Augmentations. Aside from the augmentations mentioned before, for the neural image transformation family $\mathcal{T}_n$, we used a sequential application of Poisson noise and Gaussian blur and color jittering. In contrast with recent work on contrastive visual representation learning, we only applied brightness and contrast adjustments in color jittering because neural images have a single channel that measures calcium indicator fluorescence intensity. We did not apply any cropping augmentation, such as cutout, because action representation is often highly localized and non-redundant (e.g., grooming is associated with the activity of a small set of neurons and thus with only a small number of pixels). We applied the same augmentations to each frame in single sample of neural data.

For the behavior transformation family $\mathcal{T}_b$, we used a sequential application of scaling, shear, and random temporal and spatial dropping. We did not apply rotation and translation augmentations because the animals were tethered (i.e., restrained from moving freely), and their direction and absolute location were fixed throughout the experiment. We did not use time warping because neural and behavioral information are temporally linked (e.g., fast walking has different neural representations than slow walking).

Swapping Parameters. We analyze the effects of swapping individual poses, instead of whole motion sequences, through our swapping augmentation in **Fig. 42**. We compare the distribution similarity across individuals when tested on single poses and windows of poses. We observe that the distribution similarity across individuals in behavioral modality is much larger in pose level when compared to the whole motion sequence, therefore making it easier to swap behavioral data in pose level. We quantify the distribution similarity using MMD (Mean Maximum Discrepancy) and Homogeneity metrics. Similarly, swapping individual poses decreases the overall change in the motion sequence, as quantified by the Mean Squared Distance. Yet, the degree to which identity information is hid does not strongly correlate with the window size of swapping. Therefore, overall, suggesting swapping in pose level is better than swapping whole motion sequences.

Implementation Details: For all methods, we initialized the weights of the networks randomly unless otherwise specified. To keep the experiments consistent, we always paired 32 frames of neural data with 8 frames of behavioral data. For the neural data, we used a larger time window because the timescale during which dynamic changes occur are smaller. For the paired modalities, we considered data synchronized if their center frames had the same timestamp. We trained contrastive methods for 200 epochs and set the temperature value τ to 0.1. We set the output dimension of $\mathbf{z}_b$ and $\mathbf{z}_n$ to 128. We used a cosine training schedule with three epochs of warm-up. For non-contrastive methods, we trained for 200 epochs with a learning rate of $1e-4$, and a weight decay of $1e-5$, using the Adam optimizer [220]. We ran all experiments using an Intel Core i9-7900X CPU, 32 GB of DDR4 RAM, and a GeForce GTX 1080. Training for a single SimCLR network for 200 epochs took 12 hours. To create train and test splits, we removed two trials from each animal and used them only for testing. We used the architecture shown in **Supplementary Table 1** for the neural image and behavioral pose encoder. Each layer except the final fully-connected layer was followed by Batch Normalization and a ReLU activation function [221]. For the self-attention mechanism in the behavioral encoder (**Supplementary Table 1**), we implement Bahdanau attention [248]. Given the set of intermediate behavioral representations $S \in \mathbb{R}^{T \times D}$, we first calculated,

$$\mathbf{r} = W_2 \tanh\left(W_1 S^\top\right) \quad \text{and} \quad \mathbf{a}_i = -\log\left(\frac{\exp\left(\mathbf{r}_i\right)}{\sum_j \exp\left(\mathbf{r}_j\right)}\right)$$

where W_1 and W_2 are a set of matrices of shape $\mathbb{R}^{12 \times D}$ and $\mathbb{R}^{1 \times 12}$ respectively. $\mathbf{a}_i$ is the assigned score i-th pose in the sequence of motion. Then the final representation is given by $\sum_i^T \mathbf{a}_i S_i$.

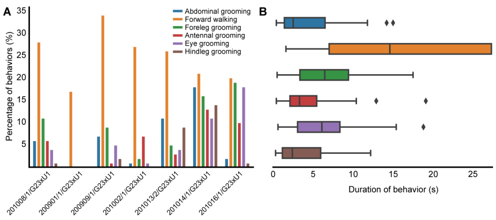


Figure 39 – **Motion Capture and two-photon dataset statistics.** Visualizing **(A)** the number of annotations per animal and **(B)** the distribution of the durations of each behavior across animals. Unlike scripted human behaviors, animal behaviors occur spontaneously. The total number of behaviors and their durations do not follow a uniform distribution, therefore making it harder to model.

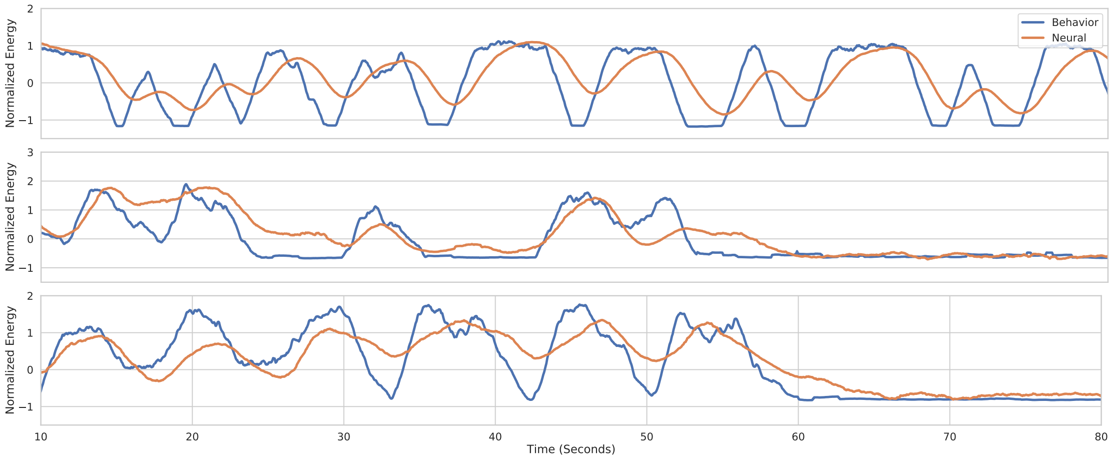


Figure 40 – **Visualizing the temporal correlation between behavioral and neural energies on multiple animals.** The behavioral and neural energies are calculated as the normalized distances between consecutive frames. The multi-modal energies show a similar temporal pattern. The slower neural energy decay is due to the calcium dynamics.

6.3 Evaluation

We test our method on three datasets. In this section, we describe these datasets, the set of baselines against which we compare our model, and finally the quantitative comparison of all models.

6.3.1 Datasets

We ran most of our experiments on a large dataset of fly neural and behavioral recordings that we acquired and describe below. To demonstrate our method's ability to generalize, we also adapted it to run on another multimodal dataset that features neural ECoG recordings and markerless motion capture[239, 240], as well as the well known H36M human motion dataset [29].

MC2P: Since there was no available neural-behavioral dataset with a rich variety of spontaneous behaviors from multiple individuals, we acquired our own dataset that we name *Motion Capture and Two-photon Dataset (MC2P)*. We will release this dataset publicly. MC2P features data acquired from tethered behaving adult flies, *Drosophila melanogaster* **(Fig. 35)**, It includes:

1. Infrared video sequences of the fly acquired using six synchronized and calibrated infrared cameras forming a ring with the animal at its center. The images are 480 × 960 pixels in size and recorded at 100 fps.
2. Neural activity imaging obtained from the axons of descending neurons that pass from the brain to fly's ventral nerve cord (motor system) and drive actions. The neural images are 480 × 736 pixels in size and recorded at 16 fps using a two-photon microscope [249] that measures the calcium influx which is a proxy for the neuron's actual firing rate.

We recorded 40 animals over 364 trials, resulting in 20.7 hours of recordings with 7,480,000 behavioral images and 1,197,025 neural images. We provide additional details and examples in the Supplementary Material. We give an example video of synchronized behavioral and neural modalities in **Supplementary Videos 1-2**.

To obtain quantitative behavioral data from video sequences, we extracted 3D poses expressed in terms of the 3D coordinates of 38 keypoints [111]. We provide an example of detected poses and motion capture in **Supplementary Videos 3-4**. For validation purposes, we manually annotated a subset of frames using eight behavioral labels: *forward walking*, *pushing*, *hindleg grooming*, *abdominal grooming*, *rest*, *foreleg grooming*, *antenna grooming*, and *eye grooming*. We provide an example of behavioral annotations in **Supplementary Video 5**.

ECoG dataset [239, 240]: This dataset was recorded from epilepsy patients over a period of 7-9 days. Each patient had 90 electrodes implanted under their skull. The data comprises human neural Electrocorticography (ECoG) recordings and markerless motion capture of upper-body 2D poses. The dataset is labeled to indicate periods of voluntary spontaneous motions, or rest. As for two-photon images in flies, ECoG recordings show a significant domain gap across individual subjects. We applied our multi-modal contrastive learning approach on ECoG and 2D pose data along with swapping-augmentation. Then, we applied an across-subject benchmark in which we do action recognition on a new subject without known action labels.

We apply multi-modal contrastive learning on windows of time series and RGB videos. We make the analogy that, similar to the neural data, RGB videos from different view angles show a domain gap although they are tied to the same 3D pose. Therefore, to test our method, we select three individuals with different camera angles where all actors perform the same three actions. We test domain adaptation using the Across-Subject benchmark, where we train our linear action classifier on labels of one individual and test it on the others. We repeat the same experiment three times and report the mean results. We show the results of Across-Subject and Identity Recognition in **Table 7**.

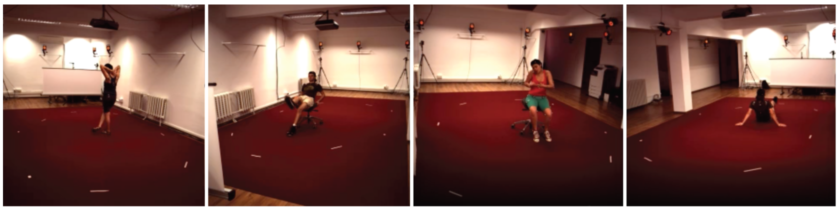


Figure 41 – **Domain Gap in the H3.6M dataset.** Similar to the domain gap across nervous systems, RGB images show a significant domain gap when the camera angle changes across individuals. We guide action recognition across cameras in RGB images using 3D poses and behavioral swapping.

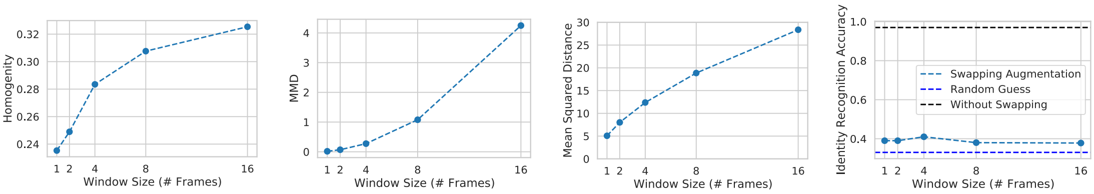


Figure 42 – **Changing window size for the swapping of behavioral modality on the MC2P dataset.** Statistics of the behavioral modality as a function of changing the window size. Decreasing the window size increases clustering homogeneity and Mean Maximum Discrepancy (MMD) when applied to the raw data, therefore suggesting higher quality swapping in individual poses instead of sequences of poses. Swapping augmentation with a smaller window size lowers the degree of perturbation, quantified by Mean Squared Distance. However, identity recognition accuracy does not change considerably when swapping is done with different window sizes.

For preprocessing, we remove global translation and rotation from 3D poses by subtracting the root joint and then rotating the skeletons to point in the same direction. We use resnet18 for the RGB encoder and a 4 layer convolutional network for the 3D pose encoder. We use S1, S5 and S7 and all their behaviors for training, except for the three behaviors which we used for testing. For each number, we report three-fold cross-validation results.

H3.6M [29]: H3.6M is a multi-view motion capture dataset that is not inherently multimodal. However, to test our approach in a very different context than the other two cases, we treated the videos acquired by different camera angles as belonging to separate domains. Since videos are tied to 3D poses, we used these two modalities and applied swapping augmentation together with multimodal contrastive learning to reduce the domain gap across individuals. Then, we evaluated the learned representations by performing action recognition on a camera angle that we do not have action labels for. This simulates our across-subject benchmark used in the MC2P dataset. For each experiment we selected three actions, which can be classified without examining large window sizes. We give additional details in the Supplementary Material.

Dataset Collection. Here we provide a more detailed technical explanation of the experimental dataset. Transgenic female *Drosophila melanogaster* flies aged 2-4 days post-eclosion were selected for experiments. They were raised on a 12h:12h day, night light cycle and recorded in either the morning or late afternoon Zeitgeber time. Flies expressed both GCaMP6s and tdTomato in all brain neurons as delineated by otd-Gal4 expression,

$(;\frac{Otd-nls{:}FLPo(attP40)}{P20XUAS-IVS-GCaMP6sattP40};\frac{R57C10-GAL4,tub>GAL80>}{Pw[+mC]=UAS-tdTom.S3}.)$ The fluorescence of GCaMP6s proteins within the neuron increases when it binds to calcium. There is an increase in intracellular calcium when neurons become active and fire action potentials. Due to the relatively slow release (as opposed to binding) of calcium by GCaMP6s molecules, the signal decays exponentially. We also expressed the red fluorescent protein, tdTomato, in the same neurons as an anatomical fiduciary to be used for neural data registration. This compensates for image deformations and translations during animal movements. We recorded neural data using a two-photon microscope (ThorLabs, Germany; Bergamo2) by scanning the cervical connective. This neural tissue serves as a conduit between the brain and ventral nerve cord (VNC) [32]. The brain-only GCaMP6s expression pattern in combination with restrictions of recording to the cervical connective allowed us to record a large population of descending neuron axons while also being certain that none of the axons arose from ascending neurons in the VNC. Because descending neurons are expected to drive ongoing actions [189], this imaging approach has the added benefit of ensuring that the imaged cells should, in principle, relate to paired behavioral data.

Neural Pre-Processing. For neural data processing, data were synchronized using a custom Python package [250]. We then estimated the motion of the neurons using images acquired on the red (tdTomato) PMT channel. The first image of the first trial was selected as a reference frame to which all other frames were registered. For image registration, we estimated the vector field describing the motion between two frames. To do this, we numerically solved the optimization problem in **Eq. 30**, where w is the motion field, $\mathscr{I}_t$ is the image being transformed, $\mathscr{I}_r$ is the reference image, and Ω is the set of all pixel coordinates [32, 251].

$$\begin{aligned}\hat{w} = argmin_w \sum_{x\in\Omega} \|\mathscr{I}_t(x+w(x)) - \mathscr{I}_r(x)\|_2^2 \\ -\lambda \sum_{x\in\Omega} \|\nabla w(x)\|_2^2\end{aligned} \tag{30}$$

A smoothness promoting parameter λ was empirically set to 800. We then applied $\hat{w}$ to the green PMT channel (GCaMP6s). To denoise the motion corrected green signal, we trained a DeepInterpolation network [252] for nine epochs for each animal and applied it to the rest of the frames. We only used the first 100 frames of each trial and used the first and last trials as validation data. The batch size was set to 20 and we used 30 frames before and after the current frame as input. In order to have a direct correlation between pixel intensity and neuronal activity we applied the following transformation to all neural images $\frac{F-F_0}{F_0} \times 100$, where F_0 is the baseline fluorescence in the absence of neural activity. To estimate F_0, we used the pixel-wise minimum of a moving average of 15 frames.

Neural Fluorescence Signal Decay. The formal relationship between the neural image $\mathbf{n}_t$ and neural activity (underlying neural firings) $\mathbf{s}_t$ can be modeled as a first-order autoregressive process

$$\mathbf{n}_t = \gamma \mathbf{n}_{t-1} + \alpha \mathbf{s}_t,$$

where $\mathbf{s}_t$ is a binary variable indicating an event at time t (e.g. the neuron firing an action potential). The amplitudes γ and α determine the rate at which the signal decays and the initial response to an event, respectively. In general, $0 < \gamma < 1$, therefore resulting in an exponential decay of information pertaining to $\mathbf{s}_t$ to be inside of $\mathbf{n}_t$. A single neural image $\mathbf{n}_t$ includes decaying information from previous neural activity, and hence carries information from previous behaviors. For more detailed information on calcium dynamics, see [28, 253]. Assuming no neural firings, $\mathbf{s}_t = 0$, $\mathbf{n}_t$ is given by $\mathbf{n}_t = \gamma^t \mathbf{n}_0$. Therefore, we define the calcium kernel $\mathcal{K}$ as $\mathcal{K}_t = \gamma^t$.

Dataset Analysis. We show the distribution of annotations across 7 animals and action duration distribution in **Supplementary Fig. 39**. Unlike scripted actions in human datasets, the animal behavior is spontaneous, therefore does not follow a uniform distribution. The average duration of behaviors can also change across behaviors. Walking is the most common behavior and lasts longer than other behaviors. We visualize the correlation between the neural and behavioral energy in **Supplementary Fig. 40**. We quantify the energy as the Euclidean distance between consecutive, vectorized 3D poses. Similarly, for the neural energy, we calculate the Euclidean distance between consecutive images. To be able to compare corresponding energies, we first synchronize neural and behavioral modalities. We then smooth the corresponding time series using Gaussian convolution with kernel size of 11 frames. We observe that there is a strong correlation between the modalities, suggesting large mutual information.

6.3.2 Baselines

We evaluated our method using two supervised baselines, Neural Linear and Neural MLP. These directly predict action labels from neural data without any unsupervised pretraining using cross-entropy loss. We also compared our approach to three regression methods that attempt to regress behavioral data from neural data, which is a common neural decoding technique. These include a recent neural decoding algorithm, BehaveNet [58], as well as to two other regression baselines with recurrent and convolutional approaches: Regression (Recurrent) and Regression (Convolution). In addition, we compare our approach to recent self-supervised representation learning methods, including SeqCLR [68] and SimCLR [5]. We also combine convolutional regression-based method (Reg. (Conv)) or the self-supervised learning algorithm SimCLR with the common domain adaptation techniques Gradient Reversal Layer (GRL)[241], or Mean Maximum Discrepancy [254]. This yields four domain adaptation models. Finally, we apply a recent multi-modal domain adaptation network for action recognition, MM-SADA[82] on MC2P dataset. For all of these methods, we used the same backbone architecture. We describe the

Tasks →		**Single-Subject ↑**		**Across-Subject ↑**		**Identity Recog. ↓**	
Percentage of Data →		**0.5**	**1.0**	**0.5**	**1.0**	**0.5**	**1.0**
Random Guess		16.6	16.6	16.6	16.6	12.5	12.5
Neural (Linear)	**Sup.**	29.3	32.5	18.4	18.4	100.0	100.0
Neural (MLP)		–	–	18.4	18.4	100.0	100.0
SeqCLR [68]	**Self-Supervised**	39.5	42.1	21.9	28.4	93.0	96.5
Ours (Neural Only)		42.0	44.8	21.3	30.6	94.1	96.8
SimCLR[5]		54.3	57.6	46.9	50.6	69.9	80.3
Regression (Recurrent)		53.6	59.7	49.4	51.2	89.5	91.8
Regression (Convolution)		52.6	59.6	50.6	55.8	88.7	92.5
BehaveNet [58]		54.6	60.2	50.5	56.8	80.2	83.4
Ours		**57.9**	**63.3**	**54.8**	**61.9**	**12.5**	**12.5**
SimCLR [5] + MMD	**Domain Ada.**	53.6	57.8	50.1	53.1	18.4	21.2
SimCLR [5] + GRL		53.5	56.3	49.9	52.3	16.7	19.1
Reg. (Conv.) + MMD		54.5	60.7	52.6	55.4	18.2	19.5
Reg. (Conv.) + GRL		55.5	60.2	51.8	55.7	17.2	17.3
MM-SADA [82]		53.1	56.2	49.2	52.1	13.8	15.2

Table 5 – **Action Recognition Accuracy.** Single- and Across-Subject action recognition results on the MC2P dataset. Neural MLP results for the single-subject task are removed because single subject often do not have enough labels for every action. Smaller numbers are better for Identity Recognition. Our method performs better than previous neural decoding methods and other self-supervised learning methods in all benchmarks, while at the same time closing the domain gap between animals, as shown by the identity recognition task.

backbone architecture and the baseline methods in more detail in the Supplementary Material.

Supervised:A feedforward network trained with manually annotated action labels using cross-entropy loss, having neural data as input. We discarded datapoints that did not have associated behavioral labels. For the MLP baseline, we trained a simple three layer MLP with a hidden layer size of 128 neurons with ReLU activation and without batch normalization.

Regression (Convolutional):A fully-convolutional feedforward network trained with MSE loss for behavioral reconstruction task, given the set of neural images. To keep the architectures consistent with the other methods, the average pooling is followed by a projection layer, which is used as the final representation of this model.

Regression (Recurrent): This is similar to the one above but the last projection network was replaced with a two-layer GRU module. The GRU module takes as an input the fixed representation of neural images. At each time step, the GRU module predicts a single 3D pose with a total of eight steps to predict the eight poses associated with an input neural image. This model is trained with an MSE loss. We take the input of the GRU module as the final representation of neural encoder.

BehaveNet [58]: This uses a discrete autoregressive hidden Markov model (ARHMM) to decompose 3D motion information into discrete “behavioral syllables." As in the regression baseline, the neural information is used to predict the posterior probability of observing each discrete syllable. Unlike the original method, we used 3D poses instead of RGB videos as targets. We skipped compressing the behavioral data using a convolutional autoencoder because, unlike RGB videos, 3D poses are already low-dimensional.

SimCLR [5]: We trained the original SimCLR module without the calcium imaging data and swapping augmentations. As in our approach, we took the features before the projection layer as the final representation.

Gradient Reversal Layer (GRL) [241]: Together with the contrastive loss, we trained a two-layer MLP domain discriminator per modality, D_b and D_n, which estimates the domain of the neural and behavioral representations. Discriminators were trained by minimizing

$$\mathscr{L}_D = \textstyle\sum_{x \in \{\mathbf{b},\mathbf{n}\}} -d \log\left(D_m\left(f_m(x)\right)\right) \tag{31}$$

where d is the one-hot identity vector. Gradient Reversal layer is inserted before the projection layer. Given the reversed gradients, the neural and behavioral encoders f_n and f_b learn to fool the discriminator and outputs invariant representations across domains, hence acting as a domain adaptation module. We kept the hyperparameters of the discriminator the same as in previous work [82]. We froze the weights of the discriminator for the first 10 epochs, and trained only the $\mathscr{L}_{NCE}$. We trained the network using both loss functions, $\mathscr{L}_{NCE} + \lambda_D \mathscr{L}_D$, for the remainder of training. We set the hyperparameters λ_D to 10 empirically.

Maximum Mean Discrepancy (MMD) [254]: We replaced adversarial loss in GRL baseline with a statistical test that minimizes the distributional discrepancy from different domains.

MM-SADA [82]:A recent multi-modal domain adaptation model for action recognition that minimizes cross-entropy loss on target labels, adverserial loss for domain adaptation, and contrastive losses to maximize consistency between multiple modalities. As we do not assume any action labels during the contrastive training phase, we removed the cross-entropy loss.

SeqCLR [68]: This approach learns a uni-modal self-supervised contrastive model. Hence, we only apply it to the neural imaging data, without using the behavioral modality. As this method was previously applied on datasets with Electroencephalography (ECoG) imaging technique, we removed ECoG specific augmentations.

Maximum Mean Discrepancy (MMD): We replaced adversarial loss in GRL baseline with a statistical test to minimize the distributional discrepancy from different domains [254]. Similar to previous work, we applied MMD only on the representations before the projection layer independently on both modalities [80, 82]. Similar to the GLR baseline, we first trained 10 epochs only using the contrastive loss, and trained using the combined losses $\mathscr{L}_{NCE} + \lambda_{MMD}\mathscr{L}_{MMD}$ for the remainder. We set the hyperparameters λ_{MMD} as 1 empirically. For the domain adaptation methods GRL and MMD, we reformulated the denominator of the contrastive loss function. Given a domain function dom which gives the domain of the data sample, we replaced one side of L_{NCE} in Eq. 26 with,

$$\log \frac{\exp\left(\left\langle \mathbf{z}_b^i, \mathbf{z}_n^i \right\rangle / \tau\right)}{\sum_{k=1}^{N} \mathbf{1}_{[dom(i)=dom(k)]} \exp\left(\left\langle \mathbf{z}_b^i, \mathbf{z}_n^k \right\rangle / \tau\right)}, \tag{32}$$

where selective negative sampling prevents the formation of trivial negative pairs across domains, therefore making it easier to merge multiple domains. Negative pairs formed during contrastive learning try to push away inter-domain pairs, whereas domain adaptation methods try to merge multiple domains to close the domain gap. We found that the training of contrastive and domain adaptation losses together could be quite unstable, unless the above changes were made to the contrastive loss function.

6.3.3 Benchmarks

Since our goal is to create useful representations of neural images in an unsupervised way, we focused on single- and across-subject action recognition. Specifically, we trained our neural decoder f_n along with the others without using any action labels. Then, freezing the neural encoder parameters, we trained a linear model on the encoded features, which is an evaluation protocol widely used in the field [5, 73, 75, 255]. We used either half or all action labels. We mention the specifics of the train-test split in the Supplementary Material.

Single-Subject Action Recognition. For each subject, we trained and tested a simple linear classifier *independently* on the learned representations to predict action labels. We assume that we are given action labels on the subject we are testing. In **Table 5** we report aggregated results.

Across-Subject Action Recognition. We trained linear classifiers on N-1 subjects simultaneously and tested on the left-out one. Therefore, we assume we do not have action labels for the target subject. We repeated the experiment for each individual and report the mean accuracy in **Table 5** and **Table 7**.

Identity Recognition. As a sanity check, we attempted to classify subject identity among the individuals given the learned representations. We again used a linear classifier to test the domain invariance of the learned representations. In the case that the learned representations are domain (subject) invariant, we expect that the linear classifier will not be able to detect the domain of the representations, resulting in a lower identity recognition accuracy. Identity recognition results are rerported in **Table 5** and **Table 7**.

6.3.4 Results

Single-Subject Action Recognition on M2CP. For the Single-Subject baseline, joint modeling of common latent space out-performed supervised models by a large margin, even when the linear classifier was trained on the action labels of the tested animal. Our swapping and neural augmentations resulted in an accuracy boost when compared with a simple contrastive learning method, SimCLR[5]. Although regression-based methods can extract behavioral information from the neural data, they do not produce discriminative features. When combined with the proposed set of augmentations, our method performs better than previous neural decoding models because it extracts richer features thanks to a better unsupervised pretraining step. Domain adaptation techniques do not result in a significant difference in the single-subject baseline; the domain gap in a single animal is smaller than between animals.

Across-Subject Action Recognition on M2CP. We show that supervised models do not generalize across animals, because each nervous system is unique. Before using the proposed augmentations, the contrastive method SimCLR performed worse than convolutional and recurrent regression-based methods including the current state-of-art BehaveNet [58]. This was due to large domain gap between animals in the latent embeddings **(Fig. 36C)**. Although the domain adaptation methods MMD (Maximum Mean Discrepancy) and GRL (Gradient Reversal Layer) close the domain gap when used with contrastive learning, they do not position semantically similar points near one another **(Fig. 36D)**. As a result, domain adaptation-based methods do not result in significant improvements in the across-subject action recognition task. Although regression-based methods suffer less from the domain gap problem, they do not produce representations that are as discriminative as contrastive learning-based methods. Our proposed set of augmentations close the domain gap, while improving the action recognition baseline for self-supervised methods, for both single-subject and across-subject tasks **(Fig. 36E)**.

Action Recognition on ECoG Motion vs Rest. As shown at the bottom of **Table 7**, our approach significantly lowers the identity information in ECoG embeddings, while significantly increasing across-subject action recognition accuracy compared to the regression and multi-modal SimCLR baselines. Low supervised accuracy confirms a strong domain gap across individuals. Note that uni-modal contrastive modeling of ECoG recordings (SimCLR (ECoG)) does not yield strong across-subject action classification accuracy because uni-modal modeling cannot deal with the large domain gap in the learned representations.

Human Action Recognition on H3.6M. We observe in **Table 7** that, similar to the previous datasets, the low performance of the supervised baseline and the uni-modal modeling of RGB images (SimCLR (RGB)) are due to the domain-gap in the across-subject benchmark. This observation is confirmed by the high identity recognition of these models. Our swapping augmentation strongly improves compared to the regression and multi-modal contrastive (SimCLR) baselines. Similar to the previous datasets, uni-modal contrastive training cannot generalize across subjects, due to the large domain gap.

6.3.5 Ablation Study

We compare the individual contributions of different augmentations proposed in our method. We report these results in **Table 8**. We observe that all augmentations contribute to single- and across-subject benchmarks. Our swapping augmentation strongly affects the across-subject benchmark, while at the same time greatly decreasing the domain gap, as quantified by the identity recognition result. Other augmentations have minimal effects on the domain gap, as they only slightly affect the identity recognition benchmark.

6.4 Conclusion

We have introduced an unsupervised neural action representation framework for neural imaging and behavioral videography data. We extended previous methods by incorporating a new swapping based domain adaptation technique which we have shown to be useful on three very different multimodal datasets, together with a set of domain-specific neural augmentations. Two of these datasets are publicly available. We created the third dataset, which we call MC2P, by recording video and neural data for *Drosophila melanogaster* and will release it publicly to speed-up the development of self-supervised methods in neuroscience. We hope our work will help the development of effective brain machine interface and neural decoding algorithms. In future work, we plan to disentangle remaining long-term non-behavioral information that has a global effect on neural data, such as hunger or thirst, and test our method on different neural recording modalities. As a potential negative impact, we assume that once neural data is taken without consent, our method can be used to extract private information.

(a) First part of the Neural Encoder f_n

Layer	# filters	K	S	Output
input	1	-	-	$T \times 128 \times 128$
conv1	2	(3,3)	(1,1)	$T \times 128 \times 128$
mp2	-	(2,2)	(2,2)	$T \times 64 \times 64$
conv3	4	(3,3)	(1,1)	$T \times 64 \times 64$
mp4	-	(2,2)	(2,2)	$T \times 32 \times 32$
conv5	8	(3,3)	(1,1)	$T \times 32 \times 32$
mp6	-	(2,2)	(2,2)	$T \times 16 \times 16$
conv7	16	(3,3)	(1,1)	$T \times 16 \times 16$
mp8	-	(2,2)	(2,2)	$T \times 8 \times 8$
conv9	32	(3,3)	(1,1)	$T \times 8 \times 8$
mp10	-	(2,2)	(2,2)	$T \times 4 \times 4$
conv11	64	(3,3)	(1,1)	$T \times 4 \times 4$
mp12	-	(2,2)	(2,2)	$T \times 2 \times 2$
fc13	128	(1,1)	(1,1)	$T \times 1 \times 1$
fc14	128	(1,1)	(1,1)	$T \times 1 \times 1$

(a) Second part of the Neural Encoder f_n

Layer	# filters	K	S	Output
input	60	-	-	$T \times 128$
conv1	64	(3)	(1)	$T \times 128$
conv2	80	(3)	(1)	$T \times 128$
mp2	-	(2)	(2)	$T/2 \times 128$
conv2	96	(3)	(1)	$T/2 \times 128$
conv2	112	(3)	(1)	$T/2 \times 128$
conv2	128	(3)	(1)	$T/2 \times 128$
attention6	-	(1)	(1)	1×128
fc7	128	(1)	(1)	1×128

(a) Behavioral Encoder f_b

Layer	# filters	K	S	Output
input	60	-	-	$T \times 60$
conv1	64	(3)	(1)	$T \times 64$
conv2	80	(3)	(1)	$T \times 80$
mp2	-	(2)	(2)	$T/2 \times 80$
conv2	96	(3)	(1)	$T/2 \times 96$
conv2	112	(3)	(1)	$T/2 \times 112$
conv2	128	(3)	(1)	$T/2 \times 128$
attention6	-	(1)	(1)	1×128
fc7	128	(1)	(1)	1×128

Table 6 – **Architecture details.** Shown are half of the neural encoder f_n and behavior encoder f_b functions. How these encoders are used is shown in **Fig. 3**. Neural encoder f_n is followed by 1D convolutions similar to the behavioral encoder f_b, by replacing the number of filters. Both encoders produce 128 dimensional output, while first half of the neural encoder do not downsample on the temporal axis. *mp* denotes a max-pooling layer. Batch Normalization and ReLU activation are added after every convolutional layer.

Dataset	Tasks →	A.S. ↑	A.S. ↑	I.R. ↓
	Percentage of Data →	**0.5**	**1.0**	**1.0**
H3.6M Walking, Sitting, Posing	Random Gu.	33.0	33.0	33.0
	Supervised	46.6	48.3	100.0
	SimCLR [5] (RGB)	33.2	33.5	99.5
	SimCLR[5]	53.3	55.7	99.2
	Regression (Conv.)	65.2	68.8	68.4
	Ours	**72.4**	**73.6**	**42.3**
H3.6M Walking Together, Directions, Eating	Random Gu.	33.3	33.3	33.3
	Supervised	31.2	30.9	100.0
	SimCLR [5] (RGB)	34.6	34.4	100.0
	SimCLR[5]	52.3	53.2	94.8
	Regression (Conv.)	44.8	48.7	62.1
	Ours	**63.2**	**68.3**	**44.8**
ECoG Moving, Rest	Random Gu.	50.0	50.0	33.3
	Supervised	54.2	53.8	100.0
	SimCLR [5] (ECoG)	52.3	55.1	98.0
	SimCLR[5]	64.6	72.1	81.1
	Regression (Conv.)	64.1	71.8	74.3
	Ours	**75.8**	**81.9**	**53.0**

Table 7 – Across-subject (A.S.) and identity recognition (I.R.) results on H3.6M dataset[29] using RGB and 3D pose, and on ECoG Move vs Rest [239] using neural ECoG recordings and 2D pose. For Ours, we remove calcium imaging specific augmentations and only use swapping augmentation. Swapping augmentation closes the domain gap for the contrastive learning and strongly improves across-subject action recognition on both datasets.

Method	Single Subj.↑	Across Subj.↑	Identity Recog. ↓
w/ Swapping Augmentation	▲ + 2.7	▲ + 7.9	▼ -63.0
+ w/ Calcium Augmentation	▲ + 2.1	▲ + 2.7	■ +1.2
+ w/ Mix Augmentation	▲ + 1.1	▲ + 1.2	■ -0.8

Table 8 – **Ablation Study.** Showing the effect of different augmentations on single-subject, across-subject and identity recognition benchmarks.

Concluding Remarks Part 7

This thesis presents several solutions to the problems of unsupervised neural decoding and 3D pose estimation on laboratory animals. These challenging problems find several applications in neuroscience and brain-machine interfaces. In the following, we first summarize the accomplishments and contributions presented in the thesis. We then discuss some limitations of our approaches and identify directions for future research.

7.1 Summary

Chapter 3 presented a novel unsupervised markerless 2D pose estimation system. Extracting pose information from laboratory animals requires significant annotation labor. Our model requires only unrealistic renderings of the target animal without requiring any manual pose labeling. While deep learning models cannot be trained on unrealistic renderings of laboratory animals due to the domain gap, we propose a novel deformation-based generative network that transforms the unrealistic renderings into real images while at the same time keeping the pose information intact. This enables us to train a pose estimation network on generated images with annotations.

Chapter 4 proposed a markerless self-calibrating motion capture system on tethered animals. Although the 2D pose is enough for many applications, it suffers from the loss of information due to projection operation. Main-stream extraction of 3D pose requires multi-view calibration. However, this tedious setup is not ideal for many laboratory setups. Therefore, we model a self-calibrating system that uses the animal as the calibration pattern. To make the system even more reliable, we explain a which uses animal shape and skeleton as the 3D prior, therefore fixing the deep learning based motion capture system on the go.

Chapter 5 proposed a monocular 3D motion capture algorithm for laboratory animals. Although 3D pose includes all motion information, multi-camera systems require hardware. To ease the motion capture on laboratory animals, we explain a monocular system that can still recover 3D pose. We test our network on multiple lab animals, including flies, mice, rats, and macaques. We show that our monocular motion capture system performs reliably and on-par with multi-camera motion capture systems.

Chapter 6 proposed a novel unsupervised domain adaptation method for neural decoding, using the previously mentioned animal markerless motion capture systems for domain adaptation. We show that our neural decoding algorithm can successfully generalize across individuals and can provide better neural decoding performance on low data regimes when compared to supervised algorithms and performs better than previous unsupervised neural decoding algorithms.

7.2 Limitations and Future Work

This section discusses the main limitations of the proposed methods and suggests potential directions for future work.

3D Animal Body Shape Estimation: In this thesis, we have shown that the 2D and 3D animal pose estimation problem is solvable in terms of predicting joint locations. Although this captures the location of body parts, it does not fully encode the shape information. One exciting direction for future research is to predict the full 3D shape of an animal by fitting a detailed 3D body model to the initial 3D pose predictions, using realistic models such as [117, 235].

Multiple Animal Pose Estimation: In Chapters 3, 4, and 5, we have assumed that we have a single animal in the given image and estimated the 3D pose of a single animal. Our algorithms can still be used for multi-animal 3D pose estimation by operating on individually cropping the images around individual animals. To both benefit from scene context and interactions, one could jointly predict the pose of multiple animal from the entire image with a single-shot method. Therefore, our work is open to further extensions by incorporating the top-down multi-animal pose estimation approaches, such as [256].

Self-Supervised Learning Motion-Capture for Generalizability: For many animals, in the wild pose annotations do not exist. Present annotations are recorded only in restricted lab settings. This makes animal pose estimation hard to generalize across animals. Self-supervised and weakly-supervised methods can be incorporated into animal pose estimation to make it more reliable across animals and setups.

Domain Generalization: In Chapter 6, we have shown success in transductive domain adaptation across individuals and animals for neural decoding. However, our algorithm still requires target individuals without action labels to be visible during the training. Therefore, our algorithm must be re-trained given a new domain. However, this requirement is challenging and not feasible in many real-world applications. We hope the future directions will provide truly non-transductive domain adaptation for neural decodings [257].

General Neural Representations: In Chapter 6, we showed we can generate neural representations that can be used for downstream tasks. However, we have only tested our representations under the action recognition setting. One obvious next step is to test our embeddings on other downstream tasks such as motion prediction.

A An appendix

7.0.1 Dataset sources and splits.

The worm dataset in Chapter 3 stems from the OpenWorm initiative [155, 156]. We used three videos after subsampling to 8x speed. The OpenWorm videos are referred by strain type and timestamp. We used the three videos specified in Table A.1, downloaded from YouTube at subsampled framerate (8x speed compared to the original recording).

Strain	Strain description	Time stamp
OW940	zgIs128[P(dat-1)::alpha-Synuclein::YFP]	2014-03-14T13:39:36+01:00
OW940	zgIs128[P(dat-1)::alpha-Synuclein::YFP]	2014-03-06T09:11:51+01:00
OW939	zgIs113[P(dat-1)::alpha-Synuclein::YFP]	2014-02-22T14:13:49+01:00

Table A.1 – **OpenWorm videos.** Strain type and timestamp of the used videos published by [155, 156].

The worm is tracked in each video to be roughly centered. The only transformation done is scaling the original frames to resolution 128×128 pixels. We randomly picked 100 frames of these three videos for test and then picked 1000 frames out of all remaining frames for unpaired training. We manually annotated every 10th frame (100 frames) from the unpaired training examples with two keypoints (head and tail) to train the supervised baseline, and the entire test set (100 frames) for quantifying pose estimation accuracy.

For the zebrafish larva experiments, we used Video 3 (672246_file04.avi) published in the supplemental of [123] (biorxiv.org/content/10.1101/672246v1.supplementary-material). We crop the original video from 1920×1080 pixels to the region with top left corner $(500, 10)$ and bottom right $(1290, 800)$, and scale it to 128×128 pixels. We deleted some repetitive frames where the zebrafish is not moving to increase the percentage of frames where zebrafish is bending. In total, we retained 600 frames. We selected the last 100 frames for test and 500 left for unpaired training.

Besides the test images, we also manually annotated every 5th (100 frames) from the 500 training images as the training data for the supervised baseline.

Bibliography

[1] O. G. Sani, Y. Yang, M. B. Lee, H. E. Dawes, E. F. Chang, *et al.*, “Mood variations decoded from multi-site intracranial human brain activity”, *Nature Biotechnology*, 2018.

[2] K. Utsumi, K. Takano, Y. Okahara, T. Komori, O. Onodera, *et al.*, “Operation of a p300-based brain-computer interface in patients with duchenne muscular dystrophy”, *Scientific Reports*, 2018.

[3] J. I. Glaser, A. S. Benjamin, R. H. Chowdhury, M. G. Perich, L. E. Miller, *et al.*, “Machine learning for neural decoding”, *eNeuro*, 2020.

[4] X. Gu, Z. Cao, A. Jolfaei, P. Xu, D. Wu, *et al.*, “Eeg-based brain-computer interfaces (bcis): A survey of recent studies on signal sensing technologies and computational intelligence approaches and their applications”, *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 2021.

[5] T. Chen, S. Kornblith, M. Norouzi, and G. Hinton, “A simple framework for contrastive learning of visual representations”, in *Proceedings of the International Conference on Machine Learning (ICML)*, 2020.

[6] B. Wandt, M. Rudolph, P. Zell, H. Rhodin, and B. Rosenhahn, “CanonPose: Self-supervised monocular 3D human pose estimation in the wild”, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[7] D. Mehta, O. Sotnychenko, F. Mueller, W. Xu, S. Sridhar, *et al.*, “Single-shot multi-person 3d pose estimation from monocular rgb”, in *3dv*, 2018.

[8] D. Tome, C. Russell, and L. Agapito, “Lifting from the deep: Convolutional 3d pose estimation from a single image”, in *arXiv*, 2017.

[9] A. Newell, K. Yang, and J. Deng, “Stacked hourglass networks for human pose estimation”, *Proceedings of the International Conference on Computer Vision (ECCV)*, 2016.

[10] J. Martinez, R. Hossain, J. Romero, and J. J. Little, “A simple yet effective baseline for 3D human pose estimation”, in *IEEE International Conference on Computer Vision (ICCV)*, The University of British Columbia, Vancouver, Canada, 2017.

[11] S. Gunel, H. Rhodin, D. Morales, J. Compagnolo, P. Ramdya, *et al.*, “Deepfly3d, a deep learning-based approach for 3d limb and appendage tracking in tethered, adult drosophila”, in *eLife*, 2019.

[12] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*. 2000.

[13] G. Pavlakos, X. Zhou, and K. Daniilidis, “Ordinal depth supervision for 3d human pose estimation”, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[14] P. C. Bala, B. R. Eisenreich, S. Bum Michael Yoo, H. Y. Hayden Benjamin Soo Park, and J. Zimmermann, “Automated markerless pose estimation in freely moving macaques with OpenMonkeyStudio”, *Nature Communications*, 2020.

[15] J. D. Marshall, D. E. Aldarondo, T. W. Dunn, W. L. Wang, G. J. Berman, *et al.*, “Continuous whole-body 3D kinematic recordings across the rodent behavioral repertoire”, *Neuron*, 2021.

[16] J. Cao, H. Tang, H.-S. Fang, X. Shen, C. Lu, *et al.*, “Cross-domain adaptation for animal pose estimation”, *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[17] A. Mathis, T. Biasi, S. Schneider, M. Yuksekgonul, B. Rogers, *et al.*, “Pretraining boosts out-of-domain robustness for pose estimation”, in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, 2021.

[18] T. Nath, A. Mathis, A. C. Chen, A. Patel, M. Bethge, *et al.*, “Using deeplabcut for 3d markerless pose estimation across species and behaviors”, *Nature Protocols*, 2019.

[19] D. A. Moses, S. L. Metzger, J. R. Liu, G. K. Anumanchipalli, J. G. Makin, *et al.*, “Neuroprosthesis for decoding speech in a paralyzed person with anarthria”, *New England Journal of Medicine*, 2021.

[20] “Progress towards restoring upper limb movement and sensation through intracortical brain-computer interfaces”, *Current Opinion in Biomedical Engineering*, 2018.

[21] S. Niketeghad and N. Pouratian, “Brain machine interfaces for vision restoration: The current state of cortical visual prosthetics”, *Neurotherapeutics*, 2019.

[22] X. Zabulis, P. Koutlemanis, H. Baltzakis, and D. Grammenos, “Multiview 3d pose estimation of a wand for human-computer interaction”, in *Advances in Visual Computing*, G. Bebis, R. Boyle, B. Parvin, D. Koracin, S. Wang, *et al.*, Eds., 2011.

[23] S. Prokudin, M. J. Black, and J. Romero, “Smplpix: Neural avatars from 3d human models”, in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, 2021, pp. 1810–1819.

[24] C. Yang, X. Wang, and S. Mao, “Rfid-based 3d human pose tracking: A subject generalization approach”, *Digital Communications and Networks*, 2021.

[25] S. Günel, F. Aymanns, S. Honari, P. Ramdya, and P. Fua, “Overcoming the domain gap in neural action representations”, in *arXiv*, 2022.

[26] X. Wang, A. Farhadi, R. P. N. Rao, and B. W. Brunton, “Ajile movement prediction: Multimodal deep learning for natural human neural recordings and video”, in *Association for the Advancement of Artificial Intelligence (AAAI)*, 2018.

[27] S. Raghu, N. Sriraam, E. D. Gommer, D. M. Hilkman, Y. Temel, *et al.*, “Cross-database evaluation of eeg based epileptic seizures detection driven by adaptive median feature baseline correction”, *Clinical Neurophysiology*, 2020.

[28] E. A. Pnevmatikakis, J. Merel, A. Pakman, and L. Paninski, “Bayesian spike inference from calcium imaging data”, *arXiv*, 2013.

[29] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu, “Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments”, 2014.

[30] D. A. Dombeck, A. N. Khabbaz, F. Collman, T. L. Adelman, and D. W. Tank, “Imaging large-scale neural activity with cellular resolution in awake, mobile mice”, *Neuron*, 2007.

[31] J. D. Seelig, M. E. Chiappe, G. K. Lott, A. Dutta, J. E. Osborne, *et al.*, “Two-photon calcium imaging from head-fixed drosophila during optomotor walking behavior”, *Nature methods*, 2010.

[32] C.-L. Chen, L. Hermans, M. C. Viswanathan, D. Fortun, F. Aymanns, *et al.*, “Imaging neural activity in the ventral nerve cord of behaving adult drosophila”, *Nature communications*, 2018.

[33] C. Pandarinath, D. J. O’Shea, J. Collins, R. Jozefowicz, S. D. Stavisky, *et al.*, “Inferring single-trial neural population dynamics using sequential auto-encoders”, *Nature Methods*, 2018.

[34] A. S. Ecker, P. Berens, G. A. Keliris, M. Bethge, N. K. Logothetis, *et al.*, “Decorrelated neuronal firing in cortical microcircuits”, *Science*, 2010.

[35] U. Topalovic, Z. M. Aghajan, D. Villaroman, S. Hiller, L. Christov-Moore, *et al.*, “Wireless programmable recording and stimulation of deep brain activity in freely moving humans”, *Neuron*, 2020.

[36] A. E. Urai, B. Doiron, A. M. Leifer, and A. K. Churchland, “Large-scale neural recordings call for new insights to link brain and behavior”, *arXiv*, 2021.

[37] W. G. Gonzalez, H. Zhang, A. Harutyunyan, and C. Lois, “Persistence of neuronal representations through time and damage in the hippocampus”, *Science*, 2019.

[38] T. Yoshida and K. Ohki, “Natural images are reliably represented by sparse and variable populations of neurons in visual cortex”, *Nature Communications*, 2020.

[39] T. Ebina, Y. Masamizu, Y. R. Tanaka, A. Watakabe, R. Hirakawa, *et al.*, “Two-photon imaging of neuronal activity in motor cortex of marmosets during upper-limb movement tasks”, *Nature Communications*, 2018.

[40] Y. Ziv, L. D. Burns, E. D. Cocker, E. O. Hamel, K. K. Ghosh, *et al.*, “Long-term dynamics of ca1 hippocampal place codes”, *Nature Neuroscience*, vol. 16, no. 3, pp. 264–266, 2013.

[41] J. N. Betley, S. Xu, Z. F. H. Cao, R. Gong, C. J. Magnus, *et al.*, "Neurons for hunger and thirst transmit a negative-valence teaching signal", *Nature*, pp. 180–185, 2015.

[42] E. A. Pnevmatikakis, D. Soudry, Y. Gao, T. A. Machado, J. Merel, *et al.*, "Simultaneous denoising, deconvolution, and demixing of calcium imaging data.", *Neuron*, 2016.

[43] Z. T. D. Góis and A. B. L. Tort, "Characterizing speed cells in the rat hippocampus.", *Cell Rep*, 2018.

[44] E. A. Pnevmatikakis, "Analysis pipelines for calcium imaging data.", *Curr Opin Neurobiol*, 2019.

[45] S. Jewell and D. Witten, "Exact spike train inference via l(0) optimization.", *Ann Appl Stat*, 2018.

[46] J. Nassar, S. W. Linderman, M. Bugallo, and I.-S. Park, "Tree-structured recurrent switching linear dynamical systems for multi-scale modeling", *arXiv*, 2019.

[47] S. Linderman, A. Nichols, D. Blei, M. Zimmer, and L. Paninski, "Hierarchical recurrent state space models reveal discrete and continuous dynamics of neural activity in c. elegans", *bioRxiv*, 2019.

[48] S. Linderman, M. Johnson, A. Miller, R. Adams, D. Blei, *et al.*, "Bayesian learning and inference in recurrent switching linear dynamical systems", in *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2017.

[49] Y. Gao, E. Archer, L. Paninski, and J. Cunningham, "Linear dynamical neural population models through nonlinear embeddings", in *Advances in Neural Information Processing Systems (NeurIPS)*, 2016.

[50] H. Abbaspourazad, M. Choudhury, Y. T. Wong, B. Pesaran, and M. M. Shanechi, "Multiscale low-dimensional motor cortical state dynamics predict naturalistic reach-and-grasp behavior", *Nature Communications*, 2021.

[51] K. V. Shenoy and J. C. Kao, "Measurement, manipulation and modeling of brain-wide neural population dynamics", *Nature Communications*, 2021.

[52] P. D. Ganzer, S. C. Colachis, M. A. Schwemmer, D. A. Friedenberg, C. F. Dunlap, *et al.*, "Restoring the sense of touch using a sensorimotor demultiplexing neural interface", *Cell*, 2020.

[53] K. Lacourse, B. Yetton, S. Mednick, and S. C. Warby, "Massive online data annotation, crowdsourcing to generate high quality sleep spindle annotations from eeg data", *Scientific Data*, 2020.

[54] D. Wu, Y. Xu, and B. Lu, "Transfer learning for eeg-based brain-computer interfaces: A review of progresses since 2016", *arXiv*, 2020.

[55] G. Huang, G. Liu, J. Meng, D. Zhang, and X. Zhu, "Model based generalization analysis of common spatial pattern in brain computer interfaces", *Cognitive neurodynamics*, 2010.

[56] S. Wen, A. Yin, P.-H. Tseng, L. Itti, M. A. Lebedev, *et al.*, “Capturing spike train temporal pattern with wavelet average coefficient for brain machine interface”, *Scientific Reports*, 2021.

[57] C. K. S. Lau, M. Jelen, and M. D. Gordon, “A closed-loop optogenetic screen for neurons controlling feeding in drosophila”, *G3 (Bethesda)*, 2021.

[58] E. Batty, M. Whiteway, S. Saxena, D. Biderman, T. Abe, *et al.*, “Behavenet: Nonlinear embedding and bayesian neural decoding of behavioral videos”, in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019.

[59] O. G. Sani, H. Abbaspourazad, Y. T. Wong, B. Pesaran, and M. M. Shanechi, “Modeling behaviorally relevant neural dynamics enabled by preferential subspace identification”, *Nature Neuroscience*, 2021.

[60] T. D. Sanger, “Probability density estimation for the interpretation of neural population codes.”, *J Neurophysiol*, 1996.

[61] K Zhang, I Ginzburg, B. L. McNaughton, and T. J. Sejnowski, “Interpreting neuronal population activity by reconstruction: Unified framework with application to hippocampal place cells.”, *J Neurophysiol*, 1998.

[62] F. Pereira, T. Mitchell, and M. Botvinick, “Machine learning classifiers and fmri: A tutorial overview.”, *Neuroimage*, 2009.

[63] A. A. Robie, J. Hirokawa, A. W. Edwards, L. A. Umayam, A. Lee, *et al.*, “Mapping the neural substrates of behavior”, *Cell*, 2017.

[64] S. Musall, M. T. Kaufman, A. L. Juavinett, S. Gluf, and A. K. Churchland, “Single-trial neural dynamics are dominated by richly varied movements”, *Nature Neuroscience*, 2019.

[65] C. Stringer, M. Pachitariu, N. Steinmetz, C. B. Reddy, M. Carandini, *et al.*, “Spontaneous behaviors drive multidimensional, brainwide activity”, *Science*, 2019.

[66] K. Volkova, M. A. Lebedev, A. Kaplan, and A. Ossadtchi, “Decoding movement from electrocorticographic activity: A review”, *Frontiers in Neuroinformatics*, 2019.

[67] D. P. McMullen, G. Hotson, K. D. Katyal, B. A. Wester, M. S. Fifer, *et al.*, “Demonstration of a semi-autonomous hybrid brain-machine interface using human intracranial eeg, eye tracking, and computer vision to control a robotic upper limb prosthetic.”, *IEEE Trans Neural Syst Rehabil Eng*, 2014.

[68] M. N. Mohsenvand, M. R. Izadi, and P. Maes, “Contrastive representation learning for electroencephalogram classification”, in *Proceedings of the Machine Learning for Health NeurIPS Workshop*, 2020.

[69] H. J. Banville, O. Chehab, A. Hyvärinen, D. A. Engemann, and A. Gramfort, “Uncovering the structure of clinical eeg signals with self-supervised learning”, *Journal of Neural Engineering*, 2020.

[70] C.-J. Chang, W. Guo, J. Zhang, J. Newman, S.-H. Sun, *et al.*, “Behavioral clusters revealed by end-to-end decoding from microendoscopic imaging”, *bioRxiv*, 2021.

[71] Y. Liu, Q. Yan, and A. Alahi, “Social nce: Contrastive learning of socially-aware motion representations”, *arXiv*, 2020.

[72] K. Su, X. Liu, and E. Shlizerman, “Predict & cluster: Unsupervised skeleton based action recognition”, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[73] L. Lin, S. Song, W. Yang, and J. Liu, “MS2L: Multi-task self-supervised learning for skeleton based action recognition”, in *Proceedings of the ACM International Conference on Multimedia*, 2020.

[74] T. Pan, Y. Song, T. Yang, W. Jiang, and W. Liu, “Videomoco: Contrastive video representation learning with temporally adversarial examples”, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[75] I. Dave, R. Gupta, M. N. Rizve, and M. Shah, “TCLR: Temporal contrastive learning for video representation”, *arXiv*, 2021.

[76] D. Kostas, S. Aroca-Ouellette, and F. Rudzicz, “Bendr: Using transformers and a contrastive self-supervised learning task to learn from massive amounts of eeg data”, *Frontiers in Human Neuroscience*, 2021.

[77] S. M. Peterson, R. P. N. Rao, and B. W. Brunton, “Learning neural decoders without labels using multiple data streams”, *bioRxiv*, 2021.

[78] W. Li, S. Ji, X. Chen, B. Kuai, J. He, *et al.*, “Multi-source domain adaptation for decoder calibration of intracortical brain-machine interface”, *Journal of neural engineering*, 2020.

[79] A. Farshchian, J. Gallego, J. Cohen, Y. Bengio, L. Miller, *et al.*, “Adversarial domain adaptation for stable brain-machine interfaces”, *arXiv*, 2018.

[80] G. Kang, L. Jiang, Y. Wei, Y. Yang, and A. G. Hauptmann, “Contrastive adaptation network for single-and multi-source domain adaptation”, *IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI)*, 2020.

[81] R. Wang, Z. Wu, Z. Weng, J. Chen, G.-J. Qi, *et al.*, “Cross-domain contrastive learning for unsupervised domain adaptation”, *arXiv*, 2021.

[82] J. Munro and D. Damen, “Multi-modal Domain Adaptation for Fine-grained Action Recognition”, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[83] M.-H. Chen, Z. Kira, G. AlRegib, J. Yoo, R. Chen, *et al.*, “Temporal attentive alignment for large-scale video domain adaptation”, in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019.

[84] Y. Xu, J. Yang, H. Cao, K. Mao, J. Yin, *et al.*, “Aligning correlation information for domain adaptation in action recognition”, *arXiv*, 2021.

[85] G. Varol, J. Romero, X. Martin, N. Mahmood, M. J. Black, *et al.*, “Learning from synthetic humans”, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[86] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, *et al.*, "Microsoft coco: Common objects in context", in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2014.

[87] L. Sigal, A. Balan, and M. J. Black, "Humaneva: Synchronized video and motion capture dataset and baseline algorithm for evaluation of articulated human motion", *International Journal of Computer Vision (IJCV)*, 2010.

[88] N. Mahmood, N. Ghorbani, N. F. Troje, G. Pons-Moll, and M. J. Black, "Amass: Archive of motion capture as surface shapes", in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[89] M. Andriluka, L. Pishchulin, P. Gehler, and B. Schiele, "2d human pose estimation: New benchmark and state of the art analysis", in *Proceedings of the IEEE Conference on computer Vision and Pattern Recognition (CVPR)*, 2014.

[90] B. Xiao, H. Wu, and Y. Wei, "Integral human pose regression", *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[91] W. Tang and W. Ying, "Deeply learned compositional models for human pose estimation", in *Proceedings of the International Conference on Computer Vision (ECCV)*, 2018.

[92] E. Insafutdinov, L. Pishchulina, B. Andres, M. Andriluka, and B. Schiele, "Deepercut: A deeper, stronger, and faster multiperson pose estimation model", in *Proceedings of the International Conference on Computer Vision (ECCV)*, 2016.

[93] W. Yang, S. Li, W. Ouyang, and X. Wang, "Learning feature pyramids for human pose estimation", in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[94] Y. Xiu, J. Li, H. Wang, Y. Fang, and C. Lu, "Pose Flow: Efficient online pose tracking", in *The British Machine Vision Conference (BMVC)*, 2018.

[95] A.-I. Popa, M. Zanfir, and C. Sminchisescu, "Deep multitask architecture for integrated 2d and 3d human sensing", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[96] D. Mehta, S. Sridhar, O. Sotnychenko, H. Rhodin, M. Shafiei, *et al.*, "Vnect: Real-time 3d human pose estimation with a single rgb camera", in *Siggraph*, 2017.

[97] G. Rogez, P. Weinzaepfel, and C. Schmid, "Lcr-net: Localization-classification-regression for human pose", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[98] G. Pavlakos, X. Zhou, K. G. Derpanis, and K. Daniilidis, "Harvesting multiple views for marker-less 3d human pose annotations", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[99] X. Zhou, Q. Huang, X. Sun, X. Xue, and Y. Wei, "Weakly-supervised transfer for 3d human pose estimation in the wild", in *IEEE International Conference on Computer Vision, ICCV*, 2017.

[100] X. Sun, J. Shang, S. Liang, and Y. Wei, “Compositional human pose regression”, in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[101] D. Pavllo, C. Feichtenhofer, D. Grangier, and M. Auli, “3d human pose estimation in video with temporal convolutions and semi-supervised training”, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[102] X. Zhou, Q. Huang, X. Sun, X. Xue, and Y. Wei, “Towards 3d human pose estimation in the wild: A weakly-supervised approach”, in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[103] B. Wandt and B. Rosenhahn, “Repnet: Weakly supervised training of an adversarial reprojection network for 3d human pose estimation”, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[104] H. Rhodin, V. Constantin, I. Katircioglu, M. Salzmann, and P. Fua, “Neural scene decomposition for human motion capture”, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[105] Y. Yao, Y. Jafarian, and H. S. Park, “Monet: Multiview semi-supervised keypoint detection via epipolar divergence”, in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[106] T. D. Pereira, N. Tabris, J. Li, S. Ravindranath, E. S. Papadoyannis, *et al.*, “Sleap: Multi-animal pose tracking”, *bioRxiv*, 2020.

[107] A. Wu, E. K. Buchanan, M. Whiteway, M. Schartner, G. Meijer, *et al.*, “Deep graph pose: A semi-supervised deep graphical model for improved animal pose tracking”, in *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

[108] J. M. Graving, D. Chae, H. Naik, L. Li, B. Koger, *et al.*, “Deepposekit, a software toolkit for fast and robust animal pose estimation using deep learning”, *eLife*, 2019.

[109] S. Li, S. Günel, M. Ostrek, P. Ramdya, P. Fua, *et al.*, “Deformation-aware unpaired image translation for pose estimation on laboratory animals”, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[110] P. Karashchuk, K. Rupp, E. S. Dickinson, E. Sanders, E. Azim, *et al.*, “Anipose: A toolkit for robust markerless 3D pose estimation”, *bioRxiv*, 2020.

[111] S. Günel, H. Rhodin, D. Morales, J. Campagnolo, P. Ramdya, *et al.*, “DeepFly3D, a deep learning-based approach for 3D limb and appendage tracking in tethered, adult *Drosophila*”, in *eLife*, 2019.

[112] A. Gosztolai, S. Günel, V. Lobato-Ríos, M. Pietro Abrate, D. Morales, *et al.*, “Liftpose3d, a deep learning-based approach for transforming two-dimensional to three-dimensional poses in laboratory animals”, *Nature Methods*, 2021.

[113] M. Pedersen, J. B. Haurum, S. H. Bengtson, and T. B. Moeslund, “3d-zef: A 3d zebrafish tracking benchmark dataset”, in *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[114] S. Zuffi, A. Kanazawa, T. Berger-Wolf, and M. J. Black, “Three-d safari: Learning to estimate zebra pose, shape, and texture from images "in the wild"”, in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2019.

[115] N. Kulkarni, A. Gupta, D. F. Fouhey, and S. Tulsiani, “Articulation-aware canonical surface mapping”, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[116] A. Sanakoyeu, V. Khalidov, M. S. McCarthy, A. Vedaldi, and N. Neverova, “Transferring dense pose to proximal animal classes”, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[117] V. Lobato-Rios, P. G. Özdil, S. T. Ramalingasetty, J. Arreguit, A. J. Ijspeert, *et al.*, “Neuromechfly, a neuromechanical model of adult drosophila melanogaster”, *bioRxiv*, 2021.

[118] L. A. Bolaños, D. Xiao, N. L. Ford, J. M. LeDue, P. K. Gupta, *et al.*, “A three-dimensional virtual mouse generates synthetic training data for behavioral analysis”, *Nature Methods*, 2021.

[119] J. J. Sun, A. Kennedy, E. Zhan, D. J. Anderson, Y. Yue, *et al.*, “Task programming: Learning data efficient behavior representations”, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[120] C. Segalin, J. Williams, T. Karigo, M. Hui, M. Zelikowsky, *et al.*, “The mouse action recognition system (mars): A software pipeline for automated analysis of social behaviors in mice”, *bioRxiv*, 2020.

[121] K. Overman, D. Choi, K. Leung, J. Shaevitz, and G. Berman, “Measuring the repertoire of age-related behavioral changes in drosophila melanogaster”, *bioRxiv*, 2021.

[122] T. D. Pereira, J. W. Shaevitz, and M. Murthy, “Quantifying behavior to understand the brain”, *Nature Neuroscience*, 2020.

[123] R. E. Johnson, S. Linderman, T. Panier, C. L. Wee, E. Song, *et al.*, “Probabilistic models of larval zebrafish behavior reveal structure on many scales”, *Current Biology*, 2020.

[124] T. D. Pereira, D. E. Aldarondo, L. Willmore, M. Kislin, S. S. H. Wang, *et al.*, “Fast animal pose estimation using deep neural networks”, *Nature Methods*, 2019.

[125] S. Isola, J. Zhu, T. Zhou, and A. Efros, “Image-to-image translation with conditional adversarial networks”, 2017.

[126] P. Sangkloy, J. Lu, C. Fang, F. Yu, and J. Hays, “Scribbler: Controlling deep image synthesis with sketch and color”, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[127] Q. Chen and V. Koltun, “Photographic image synthesis with cascaded refinement networks”, in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[128] L. A. Gatys, A. S. Ecker, and M. Bethge, “Image style transfer using convolutional neural networks”, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[129] J. Johnson, A. Alahi, and L. Fei-Fei, “Perceptual losses for real-time style transfer and super-resolution”, in *Proceedings of the International Conference on Computer Vision (ECCV)*, Springer, 2016.

[130] L. A. Gatys, M. Bethge, A. Hertzmann, and E. Shechtman, “Preserving color in neural artistic style transfer”, *arXiv*, 2016.

[131] D. Ulyanov, A. Vedaldi, and V. Lempitsky, “Instance normalization: The missing ingredient for fast stylization”, *arXiv*, 2016.

[132] L. A. Gatys, A. S. Ecker, and M. Bethge, “A neural algorithm of artistic style”, *arXiv*, 2015.

[133] C. Li and M. Wand, “Combining markov random fields and convolutional neural networks for image synthesis”, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[134] R. Mechrez, I. Talmi, and L. Zelnik-Manor, “The contextual loss for image transformation with non-aligned data”, *Proceedings of the International Conference on Computer Vision (ECCV)*, 2018.

[135] Y. Jing, Y. Yang, Z. Feng, J. Ye, Y. Yu, *et al.*, “Neural style transfer: A review”, *arXiv*, 2017.

[136] M.-Y. Liu and O. Tuzel, “Coupled generative adversarial networks”, in *Neural Information Processing Systems (NeurIPS)*, 2016.

[137] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, “Unpaired image-to-image translation using cycle-consistent adversarial networks”, *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[138] K. Bousmalis, N. Silberman, D. Dohan, D. Erhan, and D. Krishnan, “Unsupervised pixel-level domain adaptation with generative adversarial networks”, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[139] A. Shrivastava, T. Pfister, O. Tuzel, J. Susskind, W. Wang, *et al.*, “Learning from simulated and unsupervised images through adversarial training.”, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[140] T. Kim, M. Cha, H. Kim, J. K. Lee, and J. Kim, “Learning to discover cross-domain relations with generative adversarial networks”, *International Conference on Machine Learning (ICML)*, 2017.

[141] D. He, Y. Xia, T. Qin, L. Wang, N. Yu, *et al.*, “Dual learning for machine translation”, in *Conference on Neural Information Processing Systems (NeurIPS)*, 2016.

[142] A. Gokaslan, V. Ramanujan, D. Ritchie, K. In Kim, and J. Tompkin, “Improving shape deformation in unsupervised image-to-image translation”, in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2018.

[143] W. Wu, K. Cao, C. Li, C. Qian, and C. C. Loy, “Transgaga: Geometry-aware unsupervised image-to-image translation”, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[144] M. Jaderberg, K. Simonyan, A. Zisserman, and K. Kavukcuoglu, “Spatial transformer networks”, in *Conference on Neural Information Processing Systems (NeurIPS)*, 2015.

[145] A. Recasens, P. Kellnhofer, S. Stent, W. Matusik, and A. Torralba, “Learning to zoom: A saliency-based sampling layer for neural networks”, in *Proceedings of the International Conference on Computer Vision (ECCV)*, 2018.

[146] R. A.G.D.S.N. P. Zhixin Shu Mihir Sahasrabudhe and I. Kokkinos, “Deforming autoencoders: Unsupervised disentangling of shape and appearance”, in *Proceedings of the International Conference on Computer Vision (ECCV)*, 2018.

[147] A. V. Dalca, M. Rakic, J. Guttag, and M. R. Sabuncu, “Learning conditional deformable templates with convolutional networks”, *Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

[148] G. Balakrishnan, A. Zhao, M. Sabuncu, J. Guttag, and A. V. Dalca, “Voxelmorph: A learning framework for deformable medical image registration”, *IEEE TMI: Transactions on Medical Imaging (T-MI)*, 2019.

[149] B. Kim, J. Kim, J.-G. Lee, D. H. Kim, S. H. Park, *et al.*, “Unsupervised deformable image registration using cycle-consistent cnn”, in *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2019.

[150] D. Rueckert, P. Aljabar, R. A. Heckemann, J. V. Hajnal, and A. Hammers, “Diffeomorphic registration using b-splines”, in *International Conference on Medical Image Computing and Computer Assisted Intervention (MICCAI)*, 2006.

[151] H. Fu, M. Gong, C. Wang, K. Batmanghelich, K. Zhang, *et al.*, “Geometry-consistent generative adversarial networks for one-sided unsupervised domain mapping”, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[152] P. Yin, J. Lyu, S. Zhang, S. J. Osher, Y. Qi, *et al.*, “Understanding straight-through estimator in training activation quantized neural nets”, in *The International Conference on Learning Representations (ICLR)*, 2019.

[153] J. Ashburner, “A fast diffeomorphic image registration algorithm”, *Neuroimage*, 2007.

[154] O. Ronneberger, P. Fischer, and T. Brox, “U-net: Convolutional networks for biomedical image segmentation”, in *International Conference on Medical image computing and computer-assisted intervention*, Springer, 2015.

[155] E. Yemini, T. Jucikas, L. J. Grundy, A. E. X. Brown, and W. R. Schafer, “A database of caenorhabditis elegans behavioral phenotypes”, in *Nature Methods*, 2013.

[156] B. Szigeti, P. Gleeson, M. Vella, S. Khayrulin, A. Palyanov, *et al.*, “Openworm: An open-science approach to modeling caenorhabditis elegans.”, *Frontiers Computational Neuroscience*, 2014.

[157] P. Ramdya, R. Thandiackal, R. Cherney, T. Asselborn, R. Benton, *et al.*, "Climbing favours the tripod gait over alternative faster insect gaits", *Nature communications*, 2017.

[158] Z. Wang, A. C. Bovik, H. R. Sheikh, S. Member, E. P. Simoncelli, *et al.*, "Image quality assessment: From error visibility to structural similarity", *IEEE Transactions on Image Processing*, 2004.

[159] L. Engstrom, *Fast style transfer*, https://github.com/lengstrom/fast-style-transfer/, 2016.

[160] A. Bielski and P. Favaro, "Emergence of object segmentation in perturbed generative models", *arXiv*, 2019.

[161] J. A. Bender, E. M. Simpson, and R. E. Ritzmann, "Computer-assisted 3d kinematic analysis of all leg joints in walking insects", *PloS one*, 2010.

[162] J. Kain, C. Stokes, Q. Gaudry, X. Song, J. Foley, *et al.*, "Leg-tracking and automated behavioural classification in drosophila", *Nature communications*, 2013.

[163] J. G. Todd, J. S. Kain, and B. L. de Bivort, "Systematic exploration of unsupervised methods for mapping behavior", *Physical biology*, 2017.

[164] T. Moeslund and E. Granum, "Multiple cues used in model-based human motion capture", in *Proceedings Fourth IEEE International Conference on Automatic Face and Gesture Recognition*, 2000.

[165] A. Isakov, S. M. Buchanan, B. Sullivan, A. Ramachandran, J. K. Chapman, *et al.*, "Recovery of locomotion after injury in drosophila depends on proprioception", *Journal of Experimental Biology*, 2016.

[166] C. S. Mendes, I. Bartos, T. Akay, S. Márka, and R. S. Mann, "Quantification of gait parameters in freely walking wild type and sensory deprived *Drosophila melanogaster*", *elife*, 2013.

[167] V. Uhlmann, P. Ramdya, R. Delgado-Gonzalo, R. Benton, and M. Unser, "Flylimbtracker: An active contour based approach for leg segment tracking in unmarked, freely behaving *Drosophila*", *PLoS One*, 2017.

[168] G. Pavlakos, X. Zhou, K. Derpanis, G. Konstantinos, and K. Daniilidis, "Coarse-to-fine volumetric prediction for single-image 3d human pose", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[169] F. Moreno-Noguer, "3d human pose estimation from a single image via distance matrix regression", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[170] B. Tekin, P. Marquez-neila, M. Salzmann, and P. Fua, "Learning to fuse 2d and 3d image cues for monocular body pose estimation", in *Proceedings of the International Conference on Computer Vision (ICCV)*, 2017.

[171] A. Elhayek, E. Aguiar, A. Jain, J. Tompson, L. Pishchulin, *et al.*, "Efficient convnet-based marker-less motion capture in general scenes with a low number of cameras", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015.

[172] H. Rhodin, N. Robertini, D. Casas, C. Richardt, H.-P. Seidel, *et al.*, “General automatic human shape and motion capture using volumetric contour cues”, in *Proceedings of the International Conference on Computer Vision (ECCV)*, 2016.

[173] T. Simon, H. Joo, I. Matthews, and Y. Sheikh, “Hand keypoint detection in single images using multiview bootstrapping”, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[174] A. Mathis, P. Mamidanna, K. M. Cury, T. Abe, V. N. Murthy, *et al.*, “Deeplabcut: Markerless pose estimation of user-defined body parts with deep learning”, *Nature Neuroscience*, 2018.

[175] K. Takahashi, D. Mikami, M. Isogawa, and H. Kimata, “Human pose as calibration pattern; 3d human pose estimation with multiple unsynchronized and uncalibrated cameras”, in *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) Workshops*, 2018.

[176] B. Triggs, P. Mclauchlan, R. Hartley, and A. Fitzgibbon, “Bundle adjustment – a modern synthesis”, in *Vision Algorithms: Theory and Practice*, 2000.

[177] J. Puwein, L. Ballan, R. Ziegler, and M. Pollefeys, “Joint camera pose estimation and 3d human pose estimation in a multi-camera setup”, 2014.

[178] P. F. Felzenszwalb and D. P. Huttenlocher, “Pictorial structures for object recognition”, *International Journal of Computer Vision (IJCV)*, 2005.

[179] C. Bishop, *Pattern Recognition and Machine Learning*. 2006.

[180] K. Murphy, Y. Weiss, and M. Jordan, “Loopy belief propagation for approximate inference: An empirical study”, in *Onference on Uncertainty in Artificial Intelligence*, 1999.

[181] C. Ionescu, F. Li, and C. Sminchisescu, “Latent structured models for human pose estimation”, in *2011 International Conference on Computer Vision*, Ieee, 2011.

[182] N. C. Klapoetke, Y. Murata, S. S. Kim, S. R. Pulver, A. Birdsey-Benson, *et al.*, “Independent optical excitation of distinct neural populations”, *Nature methods*, 2014.

[183] S. S. Bidaye, C. Machacek, Y. Wu, and B. J. Dickson, “Neuronal control of drosophila walking direction”, *Science*, 2014.

[184] S. Hampel, R. Franconville, J. H. Simpson, and A. M. Seeds, “A neural command circuit for grooming movement control”, *Elife*, 2015.

[185] G. J. Berman, D. M. Choi, W. Bialek, and J. W. Shaevitz, “Mapping the stereotyped behaviour of freely moving fruit flies”, *Journal of The Royal Society Interface*, 2014.

[186] G. Casiez, N. Roussel, and D. Vogel, “1€ filter: A simple speed-based low-pass filter for noisy input in interactive systems”, in *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, Acm, 2012.

[187] L. Maaten and G. Hinton, “Visualizing high dimensional data using t-sne”, 2008.

[188] T. Chavdarova, P. Baqué, S. Bouquet, A. Maksai, C. Jose, *et al.*, "The wildtrack multi-camera person dataset", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[189] J. Cande, S. Namiki, J. Qiu, W. Korff, G. M. Card, *et al.*, "Optogenetic dissection of descending behavioral control in *Drosophila*", *eLife*, 2018.

[190] T.-W. Chen, T. J. Wardill, Y. Sun, S. R. Pulver, S. L. Renninger, *et al.*, "Ultrasensitive fluorescent proteins for imaging neuronal activity", *Nature*, 2013.

[191] C. E. McKellar, J. L. Lillvis, D. E. Bath, J. E. Fitzgerald, J. G. Cannon, *et al.*, "Threshold-based ordering of sequential actions during drosophila courtship", *Current Biology*, 2019.

[192] A. M. Seeds, P. Ravbar, P. Chung, S. Hampel, F. M. Midgley Jr, *et al.*, "A suppression hierarchy among competing motor programs drives sequential grooming in drosophila", *Elife*, 2014.

[193] M. B. Feany and W. W. Bender, "A drosophila model of parkinson's disease", *Nature*, 2000.

[194] V. Hewitt and A. Whitworth, "Mechanisms of parkinson's disease: Lessons from drosophila", in *Current topics in developmental biology*, 2017.

[195] H.-S. Fang, S. Xie, Y.-W. Tai, and C. Lu, "RMPE: Regional multi-person pose estimation", in *IEEE International Conferene on Computer Vision (ICCV)*, 2017.

[196] S.-E. Wei, V. Ramakrishna, T. Kanade, and Y. Sheikh, "Convolutional pose machines", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[197] Z. Cao, T. Simon, S.-E. Wei, and Y. Sheikh, "Realtime multi-person 2D pose estimation using part affinity fields", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[198] R. Hartley and A. Zisserman, *Multiple View Geometry in Computer Vision*, 2nd ed. 2003.

[199] Q. Gaudry, E. J. Hong, J. Kain, B. L. de Bivort, and R. I. Wilson, "Asymmetric neurotransmitter release enables rapid odour lateralization in *Drosophila*", *Nature*, 2013.

[200] A. S. Machado, D. M. Darmohray, J. Fayad, H. G. Marques, and M. R. Carey, "A quantitative framework for whole-body coordination reveals specific deficits in freely walking ataxic mice", *eLife*, 2015.

[201] B. D. DeAngelis, J. A. Zavatone-Veth, and D. A. Clark, "The manifold structure of limb coordination in walking *Drosophila*", *eLife*, 2019.

[202] H.-J. Lee and Z. Chen, "Determination of 3D human body postures from a single view", *Computer Vision, Graphics, and Image Processing*, 1985.

[203] C. J. Taylor, "Reconstruction of articulated objects from point correspondences in a single uncalibrated image", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2000.

[204] C. Chen and D. Ramanan, "3D human pose estimation = 2D pose estimation + matching", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

[205] A. Gupta, J. Martinez, J. J. Little, and R. J. Woodham, "3D pose from motion for cross-view action recognition via non-linear circulant temporal encoding", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2014.

[206] J. J. Sun, J. Zhao, L.-C. Chen, F. Schroff, H. Adam, *et al.*, "View-invariant probabilistic embedding for human pose", in *Proceedings of the European Conference on Computer Vision (ECCV)*, 2020.

[207] A. Nibali, Z. He, S. Morgan, and L. Prendergast, "3D human pose estimation with 2D marginal heatmaps", in *IEEE Winter Conference on Applications of Computer Vision (WACV)*, 2019.

[208] L. Zhao, X. Peng, Y. Tian, M. Kapadia, and D. N. Metaxas, "Semantic graph convolutional networks for 3D human pose regression", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[209] K. Iskakov, E. Burkov, V. Lempitsky, and Y. Malkov, "Learnable triangulation of human pose", in *International Conference on Computer Vision (ICCV)*, 2019.

[210] A. Kanazawa, J. Y. Zhang, P. Felsen, and J. Malik, "Learning 3D human dynamics from video", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[211] D. Mehta, O. Sotnychenko, F. Mueller, W. Xu, M. Elgharib, *et al.*, "XNect: Real-time multi-person 3D motion capture with a single RGB camera", in *ACM Transactions on Graphics*, 2020.

[212] K. Rematas, C. H. Nguyen, T. Ritschel, M. Fritz, and T. Tuytelaars, "Novel views of objects from a single image", *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2017.

[213] D. Pavllo, C. Feichtenhofer, D. Grangier, and M. Auli, "3D human pose estimation in video with temporal convolutions and semi-supervised training", in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[214] J. Liu, Y. Guang, and J. Rojas, "GAST-Net: Graph attention spatio-temporal convolutional networks for 3D human pose estimation in video", *Preprint at https://arXiv.org/abs/2003.14179*, 2020.

[215] Y. Cai, L. Ge, J. Liu, J. Cai, T.-J. Cham, *et al.*, "Exploiting spatial-temporal relationships for 3D pose estimation via graph convolutional networks", in *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[216] A. Yiannakides, A. Aristidou, and Y. Chrysanthou, "Real-time 3D human pose and motion reconstruction from monocular rgb videos", *Comput. Animat. Virtual Worlds*, 2019.

[217] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting", *The Journal of Machine Learning Research*, 2014.

[218] V. Nair and G. E. Hinton, “Rectified linear units improve restricted boltzmann machines”, in *International Conference on Machine Learning (ICML)*, 2010.

[219] K. He, X. Zhang, S. Ren, and J. Sun, “Deep residual learning for image recognition”, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[220] D. P. Kingma and J. Ba, “Adam: A method for stochastic optimization”, in *International Conference on Learning Representations, (ICLR)*, 2015.

[221] S. Ioffe and C. Szegedy, “Batch normalization: Accelerating deep network training by reducing internal covariate shift”, in *Proceedings of the International Conference on Machine Learning (ICML)*, Pmlr, 2015.

[222] G. Card and M. H. Dickinson, “Visually mediated motor planning in the escape response of *Drosophila*”, *Current Biology*, 2008.

[223] A. Wosnitza, T. Bockemühl, M. Dübbert, H. Scholz, and A. Büschges, “Inter-leg coordination in the control of walking speed in *Drosophila*”, *J. Exp. Biol.*, 2013.

[224] V. H. Sridhar, D. G. Roche, and S. Gingins, “Tracktor: Image-based automated tracking of animal movement and behaviour”, *Methods in Ecology and Evolution*, 2019.

[225] M. De Bono and C. I. Bargmann, “Natural variation in a neuropeptide y receptor homolog modifies social behavior and food response in c. elegans”, *Cell*, 1998.

[226] S. A. Budick and D. M. O’Malley, “Locomotor repertoire of the larval zebrafish: Swimming, turning and prey capture”, *J. Exp. Biol.*, 2000.

[227] M. Louis, T. Huber, R. Benton, T. P. Sakmar, and L. B. Vosshall, “Bilateral olfactory sensory input enhances chemotaxis behavior”, *Nature neuroscience*, 2008.

[228] R Strauss and M Heisenberg, “Coordination of legs during straight walking and turning in *Drosophila* melanogaster”, *Journal of Comparative Physiology A*, 1990.

[229] K. Clarke and J Still, “Gait analysis in the mouse”, *Physiology & behavior*, 1999.

[230] A. B. Wiltschko, M. J. Johnson, G. Iurilli, R. E. Peterson, J. M. Katon, *et al.*, “Mapping sub-second structure in mouse behavior”, *Neuron*, 2015.

[231] W. Hong, A. Kennedy, X. P. Burgos-Artizzu, M. Zelikowsky, S. G. Navonne, *et al.*, “Automated measurement of mouse social behaviors using depth sensing, video tracking, and machine learning”, *Proceedings of the National Academy of Sciences*, 2015.

[232] K. Feng, R. Sen, R. Minegishi, M. Dübbert, T. Bockemühl, *et al.*, “Distributed control of motor circuits for backward walking in drosophila”, *Nature communications*, 2020.

[233] R. Güler, N. Neverova, and I. Kokkinos, “Densepose: Dense human pose estimation in the wild”, *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[234] R. A. Güler and I. Kokkinos, “Holopose: Holistic 3D human reconstruction in-the-wild”, in *IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[235] M. Loper, N. Mahmood, J. Romero, G. Pons-Moll, and M. J. Black, “SMPL: A skinned multi-person linear model”, *ACM Trans. Graphics (Proc. SIGGRAPH Asia)*, 2015.

[236] J. Y. Zhang, P. Felsen, A. Kanazawa, and J. Malik, “Predicting 3D human dynamics from video”, in *IEEE International Conference on Computer Vision (ICCV)*, 2019.

[237] F. Pei, J. Ye, D. Zoltowski, A. Wu, R. H. Chowdhury, *et al.*, *Neural latents benchmark '21: Evaluating latent variable models of neural population activity*, 2021.

[238] S. Nakagome, T. P. Luu, Y. He, A. S. Ravindran, and J. L. Contreras-Vidal, “An empirical comparison of neural networks and machine learning algorithms for eeg gait decoding”, *Nature Scientific Reports*, 2020.

[239] S. Peterson, “Ecog and arm position during movement and rest”, in *bioRxvi*, 2021.

[240] S. H. Singh, S. M. Peterson, R. P. Rao, and B. W. Brunton, “Mining naturalistic human behaviors in long-term video and neural recordings”, *Journal of Neuroscience Methods*, 2021.

[241] Y. Ganin and V. Lempitsky, “Unsupervised domain adaptation by backpropagation”, in *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

[242] Y. Zhang, H. Jiang, Y. Miura, C. D. Manning, and C. P. Langlotz, “Contrastive learning of medical visual representations from paired images and text”, *arXiv*, 2020.

[243] X. Yuan, Z. Lin, J. Kuen, J. Zhang, Y. Wang, *et al.*, “Multimodal contrastive training for visual representation learning”, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2021.

[244] A. van den Oord, Y. Li, and O. Vinyals, “Representation learning with contrastive predictive coding”, *arXiv*, 2019.

[245] J. Wei and K. Zou, “EDA: Easy data augmentation techniques for boosting performance on text classification tasks”, in *Proceedings of the Conference on Empirical Methods in Natural Language Processing and the International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019.

[246] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, *et al.*, “Cutmix: Regularization strategy to train strong classifiers with localizable features”, in *International Conference on Computer Vision (ICCV)*, 2019.

[247] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, “Mixup: Beyond empirical risk minimization”, in *International Conference on Learning Representations (ICLR)*, 2018.

[248] D. Bahdanau, K. Cho, and Y. Bengio, “Neural machine translation by jointly learning to align and translate”, in *Proceedings of the International Conference on Machine Learning (ICML)*, 2015.

[249] J. Chen, F. Zhu, and J. J. Little, “A two-point method for ptz camera calibration in sports”, 2018.

[250] F. Aymanns, “Utils2p”, *https://doi.org/10.5281/zenodo.5501119*, 2021.

[251] ——, “Ofco: Optical flow motion correction”, *https://doi.org/10.5281/zenodo.5518800*, 2021.

[252] J. Lecoq, M. Oliver, J. H. Siegle, N. Orlova, and C. Koch, “Removing independent noise in systems neuroscience data using deepinterpolation”, *bioRxiv*, 2020.

[253] P. Rupprecht, S. Carta, A. Hoffmann, M. Echizen, A. Blot, *et al.*, “A database and deep learning toolbox for noise-optimized, generalized spike inference from calcium imaging”, *Nature Neuroscience*, 2021.

[254] A. Gretton, K. M. Borgwardt, M. J. Rasch, B. Schölkopf, and A. J. Smola, “A kernel method for the two-sample-problem”, in *Proceedings of Advances in Neural Information Processing Systems (NeurIPS)*, 2006.

[255] K. He, H. Fan, Y. Wu, S. Xie, and R. Girshick, “Momentum contrast for unsupervised visual representation learning”, in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2020.

[256] J. Lauer, M. Zhou, S. Ye, W. Menegas, T. Nath, *et al.*, “Multi-animal pose estimation and tracking with deeplabcut”, *bioRxiv*, 2021.

[257] W. Li, Z. Xu, D. Xu, D. Dai, and L. Gool, “Domain generalization and adaptation using low rank exemplar svms”, *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, 2017.

SEMIH GÜNEL

Personal

web: semihgunel.com   Phone:+41787359004   email: gunelsemih@gmail.com   LinkedIn: semih-gunel

I am a fourth-year Ph.D. Candidate at Computer Science Faculty, EPFL, Switzerland. I am advised by Pascal Fua from Computer Vision Lab and Pavan Ramdya from Neuroengineering Lab. I am interested in 3D computer vision, deep learning and biological applications. During my Ph.D., I have worked on 3D pose estimation, neural action decoding, time-series modeling, and self-supervised learning.

Education

2017-2022 Ph.D. Candidate in Computer Science, **École Polytechnique Fédérale de Lausanne**, Switzerland
Computer Vision Lab

2015-2016 Exchange as Master in Computer Science, **KTH Royal Institute of Technology**, Sweden
Major GPA: 4.00/4.00

2012-2017 Bachelor in Computer Science, **İhsan Doğramacı Bilkent University**, Turkey
CGPA: 3.90/4.00 (summa cum laude, 4 out of 150)

Pre-PhD Experience

2016 Jul-Sept Visiting Researcher, **Koc University Intelligent User Interfaces Lab**, Turkey

• I worked on *Sketched symbol recognition with auto-completion* to recognise partial sketches in real-time. • We were able to reduce the training and prediction time of partial sketch recognition by optimizing the underlying algorithm, which uses Constrained K-Means and SVM with majority voting.

2016 Jun-July Software Engineering Intern, **Aselsan**, Turkey

• Aselsan is the largest defense company in the Middle East. I worked with Thermal Camera Group to write testing software to generate and visualise pixel statistics. • Test Software applied non-uniformity correction and bad pixel replacement.

2015 Jun-Sept Research Intern, **Fraunhofer IIS**, Germany

• I worked on Visual Localisation in a Multi-Agent setting. • I designed a system to predict relative camera poses of multiple hand-held cameras with minimum delay. • Location and other sensory information is used for continuously switching between the best camera filming the event.

Scholarship and Honour

2018-2020	École Polytechnique Fédérale de Lausanne iPhD Fellowship
2017-2018	École Polytechnique Fédérale de Lausanne EDIC Fellowship
2012-2017	İhsan Doğramacı Bilkent University Scholarship, including full tuition and stipend
2012-2017	İhsan Doğramacı Bilkent University, Dean's High Honor Student
2012	Performed 901^{th} in Turkish National University Entrance Exam among 2 million participants

Teaching

Teaching Assistant for EPFL courses: Machine Learning (20', 21'), Computer Vision (19', 20', 21'), Convex Optimisation (18'), Foundations of Software (19')

Software

• **DeepFly3D**: (68 stars) Main author for the multi-view markerless motion capture small (3mm) *Drosophila*. Software does 2D pose estimation using deep learning, self-calibration using multi-view pose correspondences without using any calibration pattern and outlier detection using graphical models.

• **LiftPose3D**: (25 stars) Single view 3D motion capture for laboratory animals using deep learning. Easily extensible for new animals for quicker adaptation.

Main Publications

• **S. Günel**, F. Aymanns, S. Honari, P. Ramdya, P. Fua, "Overcoming the Domain Gap in Neural Action Representations" (CVPR 2022 Under Review)

• **S. Günel**, F. Aymanns, S. Honari, P. Ramdya, P. Fua, "Overcoming the Domain Gap in Contrastive Learning of Neural Action Representations" (NeurIPS Workshop 2021)

• A. Gosztolai*, **S. Günel***, H. Rhodin, P. Fua, P. Ramdya, "LiftFly3D, a deep learning-based approach for transforming 2D to 3D pose in adult Drosophila melanogaster" (CVPRW 2021, Nature Methods 2021)

• S. Li, **S. Günel**, M. Ostrek, P. Ramdya, P. Fua, H. Rhodin, "Deformation-aware Unpaired Image Translation for Pose Estimation on Laboratory Animals" (CVPR 2020)

• **S. Günel**, H. Rhodin, D. Morales, P. Ramdya and P. Fua, "DeepFly3D, a deep learning-based approach for 3D limb and appendage tracking in tethered, adult Drosophila" (eLife, 2019)

• D. Bieler, **S. Günel**, P. Fua, H. Rhodin, "Gravity as a Reference for Estimating a Person's Height from Video" (ICCV 2019)

• **S. Günel**, H. Rhodin, and P. Fua, "What Face and Body Shapes Can Tell About Height" (ICCVW, 2019)

• L. Hermans*, M. Kaynak*, J. Braun, V. Ríos, C. Chen, **S. Günel**, F. Aymanns, M. Sakar, P. Ramdya, "Long-term imaging of the ventral nerve cord in behaving adult Drosophila" (bioRxiv, Under Review, 2022)