# ANIPOSE:
# A TOOLKIT FOR ROBUST MARKERLESS 3D POSE ESTIMATION

A PREPRINT

**Pierre Karashchuk**
Neuroscience Graduate Program
University of Washington, Seattle, WA

**Katie L. Rupp**
Department of Physiology and Biophysics
University of Washington, Seattle, WA

**Evyn S. Dickinson**
Department of Physiology and Biophysics
University of Washington, Seattle, WA

**Elischa Sanders**
Molecular Neurobiology Laboratory
The Salk Institute for Biological Studies, La Jolla, CA

**Eiman Azim**
Molecular Neurobiology Laboratory
The Salk Institute for Biological Studies, La Jolla, CA

**Bingni W. Brunton***
Department of Biology
University of Washington, Seattle, WA

**John C. Tuthill***
Department of Physiology and Biophysics
University of Washington, Seattle, WA

May 26, 2020

## ABSTRACT

Quantifying movement is critical for understanding animal behavior. Advances in computer vision now enable markerless tracking from 2D video, but most animals live and move in 3D. Here, we introduce Anipose, a Python toolkit for robust markerless 3D pose estimation. Anipose consists of four components: (1) a 3D calibration module, (2) filters to resolve 2D tracking errors, (3) a triangulation module that integrates temporal and spatial constraints, and (4) a pipeline to structure processing of large numbers of videos. We evaluate Anipose on four datasets: a moving calibration board, fruit flies walking on a treadmill, mice reaching for a pellet, and humans performing various actions. Because Anipose is built on popular 2D tracking methods (e.g., DeepLabCut), users can expand their existing experimental setups to incorporate robust 3D tracking. We hope this open-source software and accompanying tutorials (anipose.org) will facilitate the analysis of 3D animal behavior and the biology that underlies it.

***Keywords*** pose estimation · robust tracking · markerless tracking · behavior · 3D · deep learning

*Co-senior authors: bbrunton@uw.edu, tuthill@uw.edu

# 1 Introduction

Tracking body kinematics is a key challenge faced by many scientific disciplines. For example, neuroscientists quantify animal movement to relate it to brain dynamics [1, 2], biomechanists quantify the movement of specific body structures to understand their mechanical properties [3], social scientists quantify the motion of multiple individuals to understand their interactions [4, 5], and rehabilitation scientists quantify body movement to diagnose and treat disorders [6, 7, 8]. In all of these disciplines, achieving rapid and accurate quantification of animal pose is a major bottleneck to scientific progress.

While it is possible for human observers to recognize certain body movements, scoring behaviors by eye is laborious and often fails to detect differences in the rapid, fine-scale movements that characterize many behaviors. Methods for automated tracking of body kinematics from video have existed for many years, but they typically rely on the addition of markers to aid with body part identification. However, the use of markers is often impractical, particularly when studying natural behaviors in complex environments or when tracking multiple small parts on an animal. Thus, there is a great need for methods that enable automated, markerless tracking of body kinematics.

Recent advances in computer vision and machine learning have dramatically improved the speed and accuracy of markerless tracking and body pose estimation. There are now a number of tools that apply these methods to track animal movement from 2D videos [9, 10, 11, 12, 13, 14]. These software packages allow users to label data, train convolutional neural networks, apply them to new data, and filter the output. Their application, particularly within the field of behavioral neuroscience, has already made an important impact [1, 15, 16].

While tracking of animal movement from 2D video is useful for monitoring specific body parts, full body pose estimation and measurement of complex or subtle behaviors require tracking in three dimensions. 3D tracking presents a number of new challenges. First, the addition of multiple cameras necessitates robust calibration and geometric triangulation across different views. Second, not all body parts are equally visible from each camera, so new strategies are required to effectively integrate the 2D detected points in time and space. Third, having more than one camera increases the complexity and scale of video acquisition and processing, motivating the development of a scalable data-processing pipeline.

To address these challenges, we introduce Anipose (a portmanteau of "animal" and "pose"), a new toolkit to quantify body kinematics in three-dimensions accurately and at scale. Anipose consists of a robust calibration module, filters to further refine 2D tracking, and a novel triangulation procedure to obtain high-quality 3D position estimates by leveraging flexible constraints in time and in space. Anipose uses a specific folder structure to facilitate automated processing of large numbers of video files, such as would be collected in a typical behavioral experiment. For users who prefer to maintain their own data processing pipelines, we also provide the functions for calibration and triangulation under aniposelib.

Here, we evaluate the performance of Anipose on ground-truth measurements from four experimental datasets acquired in different labs and with different experimental setups (Figure 1). First, to quantify the calibration and triangulation errors against known physical lengths and angles, we track a precision manufactured ChArUco calibration board filmed from 6 cameras. We then demonstrate that triangulation from multiple camera views enables 3D tracking of the mouse hand during a reaching task filmed with 2 cameras, tethered fruit flies on a spherical treadmill filmed with 6 cameras, and freely moving humans filmed with 4 cameras. Finally, we examine how filtering improves 3D pose estimation for two of these datasets: tethered fruit flies and freely moving humans. We evaluate Anipose using 2D tracking with DeepLabCut [17] and provide guidance on how current DeepLabCut users can adapt their existing experimental setups for 3D pose estimation.

The release of Anipose as free and open-source software facilitates ease of adoption, promotes ongoing contributions by community developers, and supports open science.

# 2 Results

We implement 3D tracking in a series of steps: estimation of calibration parameters from calibration videos, detection and refinement of 2D joint keypoints, triangulation and refinement of keypoints to obtain 3D joint positions, and computation of joint angles (Figure 2). In addition to the processing pipeline, the key innovations of Anipose are a robust 3D calibration module and spatiotemporal filters that refine pose estimation in both 2D and 3D. We evaluated the calibration and triangulation modules without filters by testing their ability to accurately estimate lengths and angles of a calibration board with known dimensions (Figure 1A) and to track the hand of a mouse reaching for a food pellet (Figure 1B) We then evaluate the contributions of 2D and 3D filters to accurate tracking of walking flies (Figure 1C) and humans (Figure 1D). Representative examples of tracking from each dataset are shown in Video 1. Tutorials and further documentation for Anipose are available at `http://anipose.org` .

## 2.1 Robust calibration of multiple camera views

An essential step in accurate 3D pose estimation is camera calibration, which precisely determines the relative location and parameters of each camera (i.e., the focal length and distortions). We implemented an automated procedure that calibrates the cameras from simultaneously acquired videos of a standard calibration board (e.g., checkerboard or ChArUco board) moved by hand through the cameras' field of view. We recommend the ChArUco board because its keypoints may be detected even with partial occlusion and its rotation can be determined uniquely from its pattern from multiple views. The pipeline starts by detecting keypoints on the calibra-

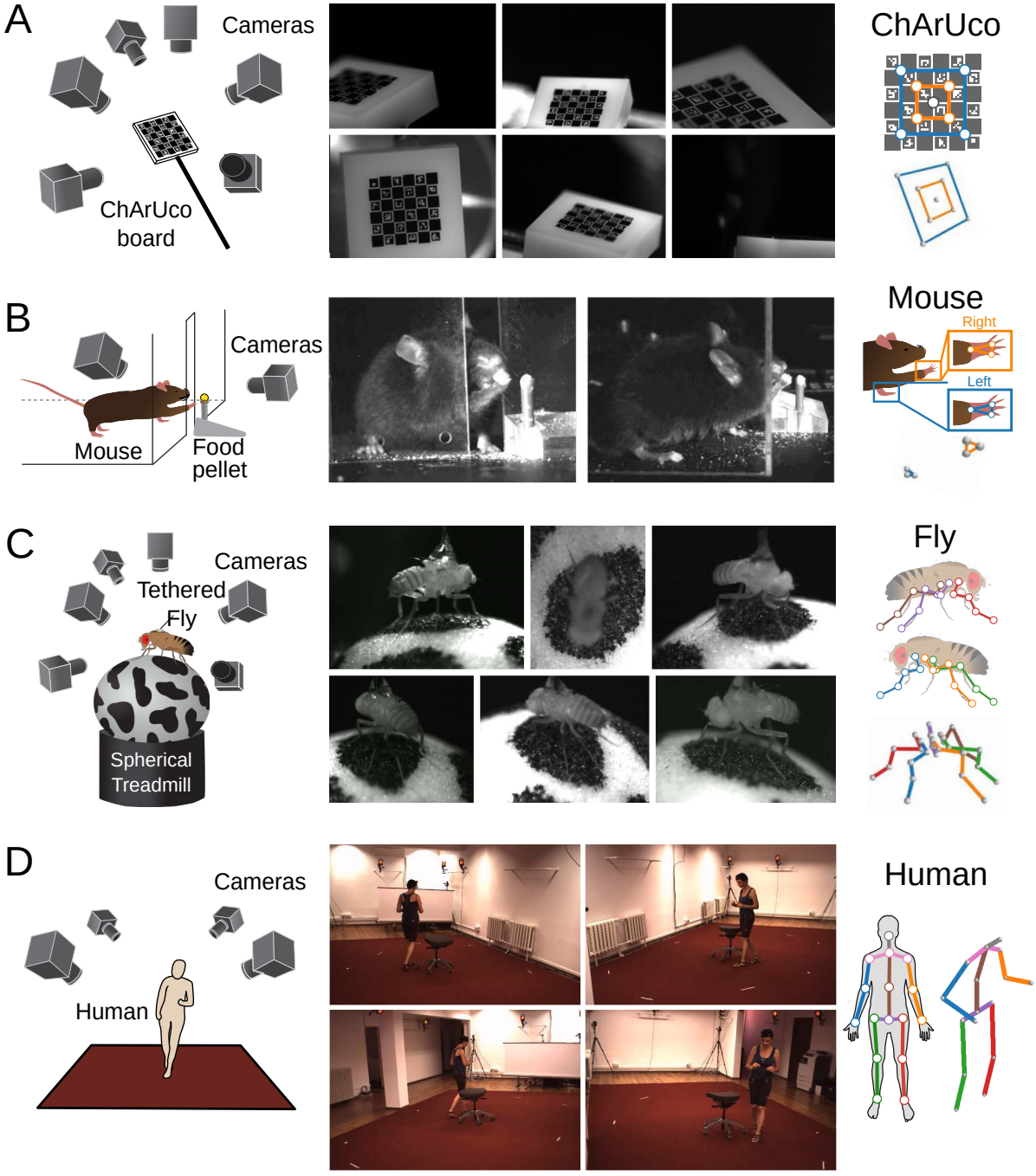


Figure 1: Four experimental datasets were used for evaluating 3D calibration and tracking with Anipose. (A) To evaluate tracking errors, a $2 \times 2$mm precision manufactured ChArUco board was simultaneously filmed from 6 cameras focused on the same point in space. We manually annotated and tracked 9 keypoints on the ChArUco board, a subset of the points that can be detected automatically with OpenCV. (B) Adult mice were trained to reach for food pellets through an opening in a clear acrylic box. After training, reach attempts were captured from 2 cameras. To quantify reach kinematics, we labeled and tracked 3 keypoints on each hand. (C) Fruit flies were tethered and positioned on a spherical treadmill, where they were able to walk, groom, etc. Fly behavior was filmed from 6 cameras evenly distributed around the treadmill. We labeled and tracked 5 keypoints on each of the 6 legs, one keypoint for each of the major leg joints. (D) As part of the Human 3.6M dataset, professional actors performing a range of actions were filmed from 4 cameras. We tracked 17 joints on each human, covering the major joints of the human body.

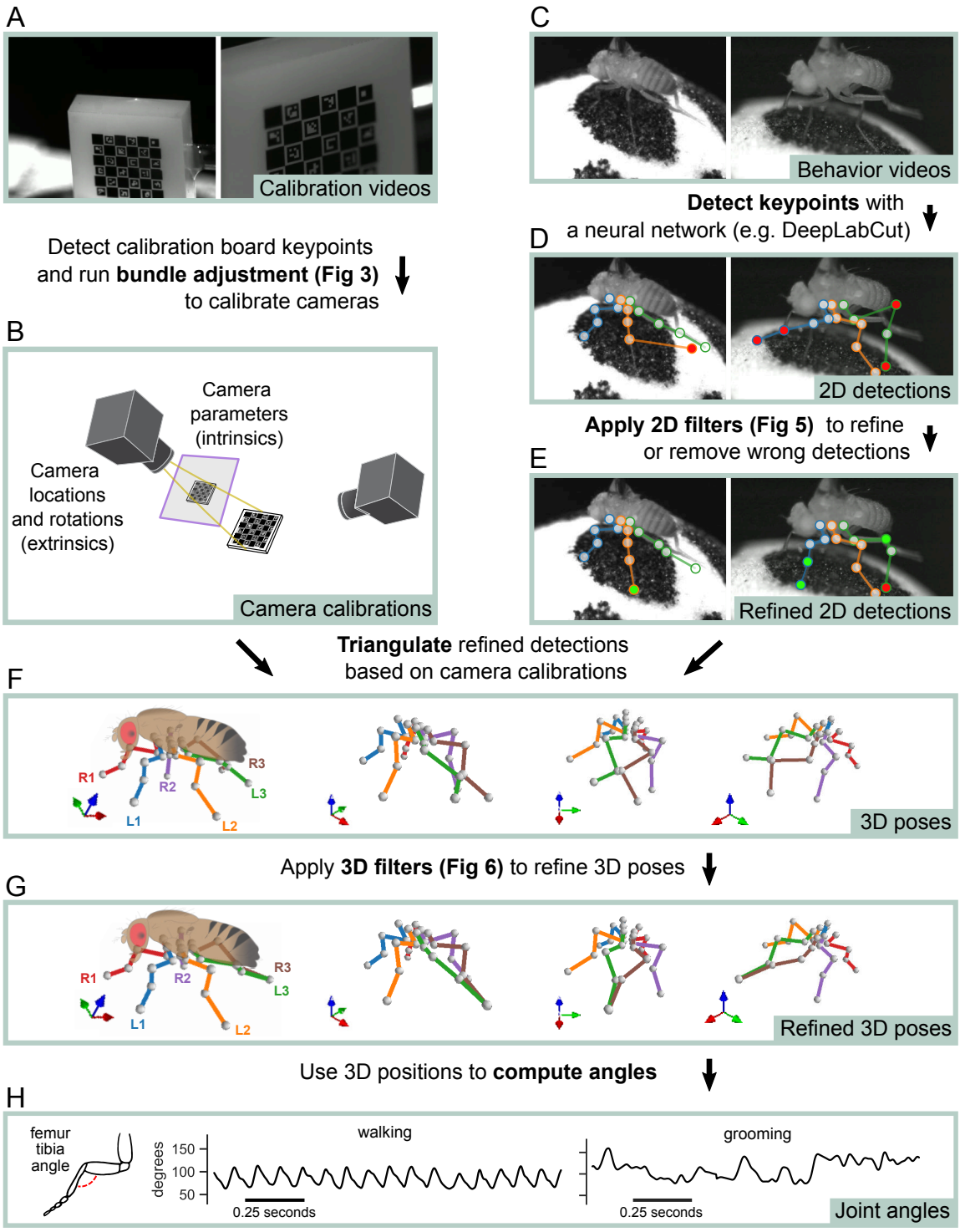


Figure 2: Overview of the Anipose 3D tracking pipeline. (A) The user collects simultaneous video of a calibration board from multiple cameras. (B) Calibration board keypoints are detected from calibration videos and processed to calculate intrinsic and extrinsic parameters for each camera using iterative bundle adjustment (Figure 3). (C) With the same hardware setup as in A, the user collects behavior videos. (D) Behavior videos are processed by a neural network (e.g., DeepLabCut) to detect 2D keypoints. (E) 2D keypoints are refined with 2D filters to obtain refined 2D detections (Figure 6). (F) The filtered 2D keypoints are triangulated to estimate 3D poses. (G) The estimated 3D poses are passed through an additional spatiotemporal filtering step to obtain refined 3D poses (Figure 7). (H) Joint angles are extracted from the refined 3D poses for further analysis.

tion board automatically using OpenCV [18] based on the board's geometric regularities (e.g., checkerboard grid pattern, specific black and white markers).

To obtain accurate camera calibrations, we developed an iterative procedure that performs bundle adjustment [19] in multiple stages, using only a random subsample of detected keypoints points in each stage (see Methods for a full description). This iterative calibration procedure minimizes the impact of erroneous keypoint detections, which would otherwise have to be manually corrected. We compared our method against other common calibration procedures on calibration videos from the fly dataset (Figure 3). We evaluate each calibration procedure by computing the reprojection error, a standard metric to quantify how closely the 2D projections of a triangulated 3D point matches its corresponding 2D keypoints in every camera view [19]. Our method produced significantly more accurate calibration than standard calibration methods, as measured by a substantial reduction in reprojection error (paired t-test relative to bundle adjustment with linear loss: t=-14.9, $p < 0.001$).

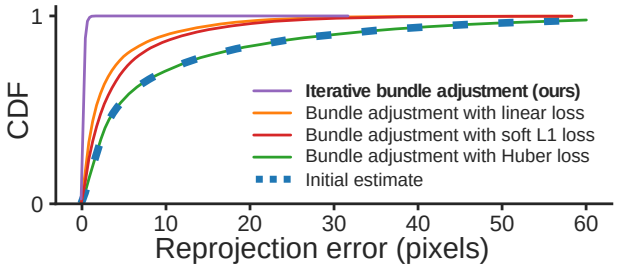


Figure 3: Iterative bundle adjustment calibration on calibration videos from the fly dataset produces more reliable calibration estimates than other calibration procedures, as measured by a substantial reduction in reprojection error.

## 2.2 Accurate reconstruction of physical lengths and angles in 3D

An important test of any calibration method is whether it can accurately reconstruct an object with known dimensions. We evaluated the Anipose calibration and triangulation toolkit by asking whether it could estimate the lengths and angles of a precisely manufactured ChArUco board [20], whose physical dimensions have a tolerance of $< 2\,\mu m$.

As our ground-truth dataset, we chose the known physical lengths and angles between all pairs of 9 corners on the ChArUco board. These 9 corners were used as keypoints (Figure 4A) to compare the accuracy of manual annotation, neural network detections, and OpenCV detections (example detections in Figure 4B). Although manual annotations are typically assumed to be the ground truth in tracking animal kinematics, we started by assessing the reliability of manual annotations relative to high-precision, sub-pixel resolution keypoint detection based on the geometry of the ChArUco board with OpenCV [18, 20]. Relative to the OpenCV points, the manual keypoint annotations had a mean error of (0.52, -0.75) pixels and standard deviation of (2.57, 2.39) pixels, in the (x, y) directions respectively (Figure 4C). These observations provide a useful baseline of manual annotation accuracy.

We evaluated the accuracy of reconstructing lengths and angles as estimated by three methods: manual keypoint annotations, OpenCV keypoint detections, and neural network keypoint detections (see Methods for detailed descriptions). As expected, OpenCV detections had the lowest error in length and angle, as they leveraged prior knowledge of the ChArUco board geometry to make high-precision corner estimates (Figure 4D). Surprisingly, neural network (trained with DeepLabCut) predictions had a lower error than manual annotations, despite the network itself being trained on manual annotations. More than 90% of poses estimated by Anipose had an error of less than 20 $\mu$m in length and 1 degree in angle, relative to the true dimensions of the ChArUco board (Figure 4D). These results demonstrate the efficacy of camera calibration with Anipose and serve as useful guides of expected performance.

## 2.3 Animal tracking in 3D

We proceeded to evaluate triangulation of markerless tracking on three different animal datasets (Figure 5). For each dataset, we computed the error of estimated joint positions and angles on labeled animals withheld from the training data. The error in estimated angle was $<16°$ in over 90% of frames, and $<10°$ in over 75% of frames. Furthermore, the error in the estimated joint position was $<18$ pixels (about 1.6mm, 0.14mm, 86mm for mouse, fly, and human datasets respectively) in over 90% of frames and $<12$ pixels (about 1mm, 0.09mm, 57mm for mouse, fly, and human datasets respectively) in over 75% of frames. Importantly, the position error in units of camera pixels is roughly comparable across these three datasets, spanning over 3 orders-of-magnitude in spatial scale. Therefore, we believe these errors are representative of what can currently be expected for accuracy of 3D markerless tracking.

Although triangulation usually resulted in accurate estimates of joint positions and angles, there were still some frames where it failed due to missing keypoint detections (as in Figure 5A). In other cases, incorrect keypoint detections led to erroneous 3D joint position estimates (as in Figure 5B). Even though these issues occurred in a small minority of frames, tracking errors are especially problematic for analyzing movement trajectories. For instance, missing estimates complicate the estimation of derivatives, whereas erroneous estimates bias the distribution of summary statistics. To prevent these issues, we leveraged complementary temporal and spatial information within each dataset to further refine tracking performance in 3D.

We did not observe the filtering to improve tracking significantly in the mouse dataset. This may be because the dataset has only 2 cameras or too few keypoints labeled in each frame.

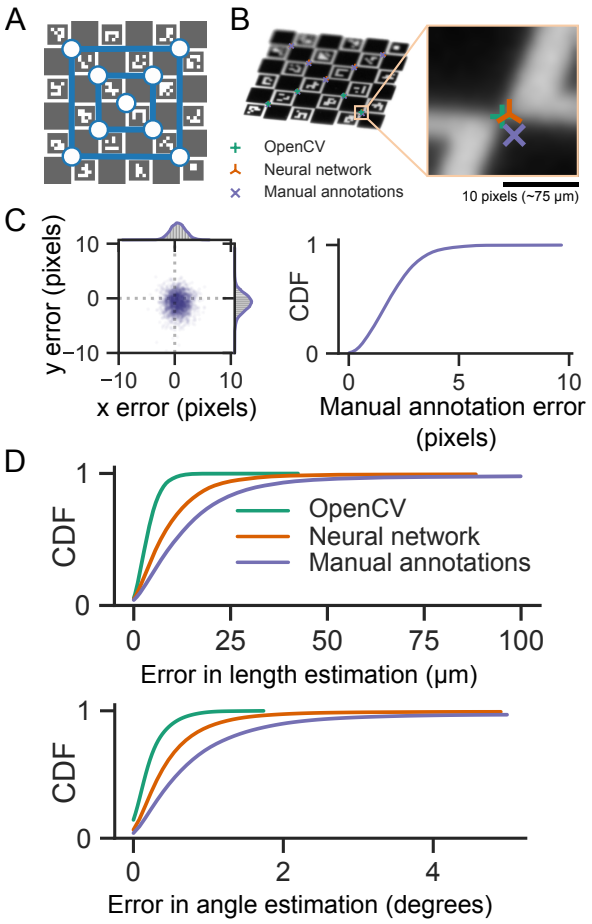


Figure 4: Anipose reliably estimates edge lengths and angles of a precision manufactured ChArUco calibration board. (A) We identified 9 corners as keypoints on the ChArUco board in 200 frames from each of 6 cameras. (B) For comparison, we used manual annotation of the same ChArUco board dataset to train a neural network. We then compared tracking errors of the manual annotations, the neural network, and OpenCV. (C) Error in manually annotated keypoints relative to the sub-pixel precision of OpenCV detections. Manually annotated keypoints had a mean error of (0.52, -0.75) pixels and standard deviation of (2.57, 2.39) pixels. (D) Lengths between all possible pairs of keypoints were computed and compared to the physical lengths. Similarly, all possible angles between triplets of keypoints were computed and compared to known physical angles. OpenCV keypoints provided the most reliable estimates, followed by neural network predictions, then manual annotations. Note that OpenCV generally detected only a small fraction of the keypoints detected by the neural network or through manual annotation (Figure 10).

### 2.4 Anipose filters for robust animal tracking

Naturally behaving animals present unique challenges for 3D pose estimation. Animals can contort their bodies into many different configurations, which means that each behavioral session may include unique poses that have not been previously encountered, even across multiple animals. These issues are exacerbated by manual annotations used for training neural networks, which are already noisy (Figure 4C). Our approach to resolving these limitations is to leverage prior knowledge that animal movements are usually smooth and continuous, and that rigid limbs do not change in length over short timescales. In particular, we developed and implemented a set of 2D and 3D filters that refine keypoints, remove errors in keypoint detections, and constrain the set of reconstructed kinematic trajectories. We demonstrate that both sets of filters work together to significantly improve pose estimation in flies and humans.

#### 2.4.1 Refining keypoints in 2D

We implemented three distinct algorithms to remove or correct errors in keypoint detection: a median filter, a Viterbi filter, and an autoencoder filter. The median and Viterbi filters operate on each tracked joint across frames, and the autoencoder filter then further refines keypoints using learned correlates among all joints. The median filter removes any point that deviates from a median filtered trajectory of user-specified length, then interpolates the missing data. The Viterbi filter finds the most likely path of keypoint detections for each joint across frames from a set of top (e.g., 20) detections per frame, given the expected standard deviation of joint movement in pixels as a prior. Finally, the autoencoder filter corrects the estimated score of each joint based on the scores of the other joints, with no parameters set by the user. Where errors in tracking cannot be corrected by filtering, the keypoint is removed altogether, since the missing joint can be inferred from other camera angles, but an erroneous keypoint can produce large discrepancies in triangulation. We found that the parameters for each filter may be set to reasonable defaults that work across a range of different datasets. Thus, the figures we present are all produced with these default parameters.

The addition of each filtering step noticeably improved the tracking of one example fly leg joint (Figure 6A). The median and Viterbi filters both reduced spurious jumps in keypoint position, which may occur if the neural network detects a similar keypoint on a different limb or at another location in the frame. The Viterbi filter is able to remove small erroneous jumps in detected keypoint trajectories while also preserving high frequency dynamics, whereas the median filter may mistakenly identify fast movements as an error and remove them.

For each of the 2D filters, we quantified the performance improvement in estimating the joint position and angle on manually annotated validation datasets. The 2D median filter significantly reduced error in joint position and angle estimation on the human dataset ($t = -14.8, p < 0.001$ for position , $t = -7.7, p < 0.001$)

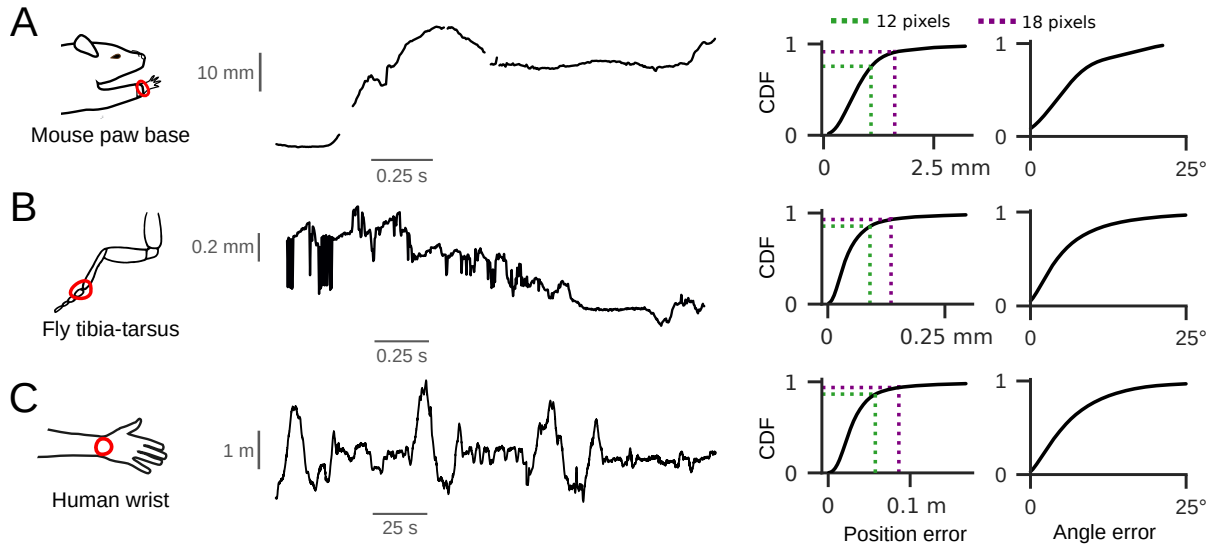


Figure 5: Anipose can consistently estimate positions and angles of joints across three different datasets, although there are outlier or missing keypoint detections prior to filters. In each dataset, the position error is below 12 pixels for 75% of the frames, and below 18 pixels for 90% of the frames. (A) On the left, we plotted an example trace of the tracked 3D position of the base of the mouse hand, projected onto the direction of the reach. On the right, we quantified the distribution of errors when estimating all joint positions and angles, relative to manual annotations. For the mouse dataset, 1 pixel corresponds to approximately 0.09 mm. (B) Same layout as in (A). Here, we plotted an example trace of the tracked 3D position of the fly hind-leg tibia-tarsus joint, projected onto the longitudinal axis of the fruit fly. For the fly dataset, 1 pixel ≈.0075 mm. (C) Same layout as in (A). Here, we plotted an example trace of the tracked 3D position of a human wrist, projected onto an arbitrary axis. Note that the human (and his wrist) is moving throughout the room. For the human dataset, 1 pixel ≈4.8 mm.

but not on the fly dataset ($t = -1.2, p = 0.2$ for position, $t = -0.98, p = 0.3$). The Viterbi filter reduced error on both fly and human datasets ($t = -4.4$ and $t = -4.1$ for fly position and angle, $t = -10.9$ and $t = -8.7$ for human position, with $p < 0.001$ for all). The autoencoder filter also reduced error in joint positions and angles on the fly dataset ($t = -5.4, p < 0.001$ for positions, $t = -2.16, p = 0.03$ for angles). We did not apply the autoencoder filter to human tracking, since all occluded points are annotated in the training dataset. In summary, we found the addition of these three filters improved the ability of Anipose to accurately estimate 2D joint positions and angles.

### 2.4.2 Refining poses and trajectories in 3D

To further refine joint position and angle estimates in 3D, we developed a novel triangulation optimization that takes advantage of the spatiotemporal structure of animal pose and behavior. Specifically, we constrain the estimated pose to be smooth in time (temporal constraints) and for certain limbs to have constant length (spatial constraints). The relative strengths of these constraints may be balanced and tuned independently. As with the 2D filters, we found empirically that we can specify default strengths to work across a variety of datasets. A full description of each filter, along with all the parameters, is detailed in the Methods. For illustration, we compared the performance of these filters (Figure 7A) to other commonly used methods from the literature (Random sample consensus, or RANSAC, triangulation and 3D median filter) on the walking fly dataset. Spatiotemporal constraints substantially improved pose estimation, even after applying 2D filters, and provided a more stable estimate of limb length (Figure 7B).

For each of the 3D filters, we evaluated the improvement in position and angle error relative to tracking with 2D filters alone (Figure 7C and D). We found that RANSAC triangulation did not improve position and angle error. The 3D median filter significantly reduced position and angle errors relative to only 2D filters for the human dataset ($t = -11.8$ for position, $t = -7.3$ for angle, $p < 0.001$ for both), but not for the fly dataset. Spatiotemporal constraints applied together provided the largest reduction in tracking error ($t = -18.7$ and $t = -6.1$ for human positions and angles, $t = -10.8$ and $t = 5.8$ for fly positions and angles, $p < 0.001$ for all). Overall, we find that the 3D filters implemented in Anipose significantly improve pose estimation compared to other methods. These improvements are also obvious in example videos of reconstructed pose before and after filtering (Video 2).

## 2.5 Structured processing of videos

Animal behavior experiments are often high-throughput, meaning that large numbers of videos are recorded over many repeated sessions of different experimental condi-

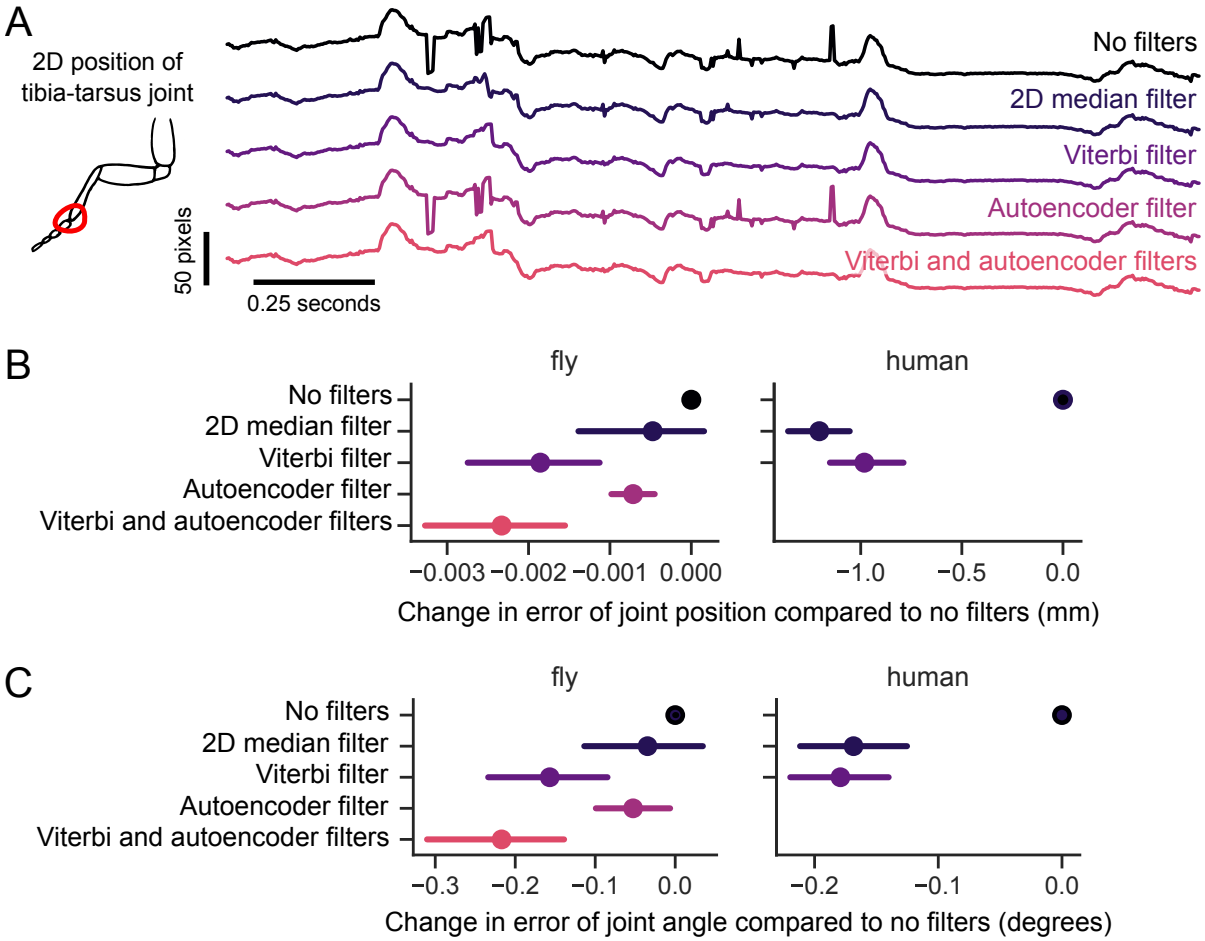


Figure 6: 2D filters improve accuracy of 2D pose estimation by taking advantage of the temporal structure of animal behavior. (A) An example trace of the x-coordinate of the 2D position of a fly's tibia-tarsus joint before and after each step in filtering. Filtering reduces spurious jumps while preserving correct keypoint detections. (B) Comparison of error in joint position before and after filtering. The mean difference in error for the same tracked points is plotted, along with the 95% confidence interval. Viterbi and autoencoder filters significantly improved the estimation of joint position in flies and humans ($p < 0.001$). For the fly dataset, 1 pixel ≈.0075 mm. For the human dataset, 1 pixel ≈4.8 mm. (C) Comparison of angle estimates before and after filtering. The mean difference is plotted as in B. Viterbi and autoencoder filters significantly improved the estimation of angles in flies and humans ($p < 0.001$). The results in (B) and (C) are evaluated on a validation dataset withheld from the training (1200 frames for the fly, 8608 frames for the humans).

tions. To make the process of 3D tracking scalable to large datasets, we designed a specific file structure (Figure 9) to organize and process behavior videos, configuration files, and calibration data. This file structure also facilitates scalable analysis of body kinematics across individual animals and experimental conditions. For example, the command "anipose analyze" detects keypoints for each video in the project folder, and "anipose calibrate" obtains calibration parameters for all the cameras in all calibration folders. Each command operates on all videos in the project, circumventing the need to process each video individually. In addition, this design allows the user to easily re-analyze the same dataset using different filtering parameters or with different 2D tracking libraries. For the users that prefer to set up their own pipelines, we also package the calibration, triangulation, and filtering functions in the aniposelib library.

## 3 Discussion

In this paper, we introduce Anipose, a new open-source Python toolkit to accurately track animal movement in 3D. Anipose is designed to augment existing neural network-based methods for 2D markerless tracking, such as DeepLabCut [9]. Because 3D tracking presents new challenges, the key contributions of Anipose are substantial improvements in optimization for robust calibration, triangulation, and filtering. We validated each new method and the full pipeline against ground truth data from four different experimental datasets and three organ-

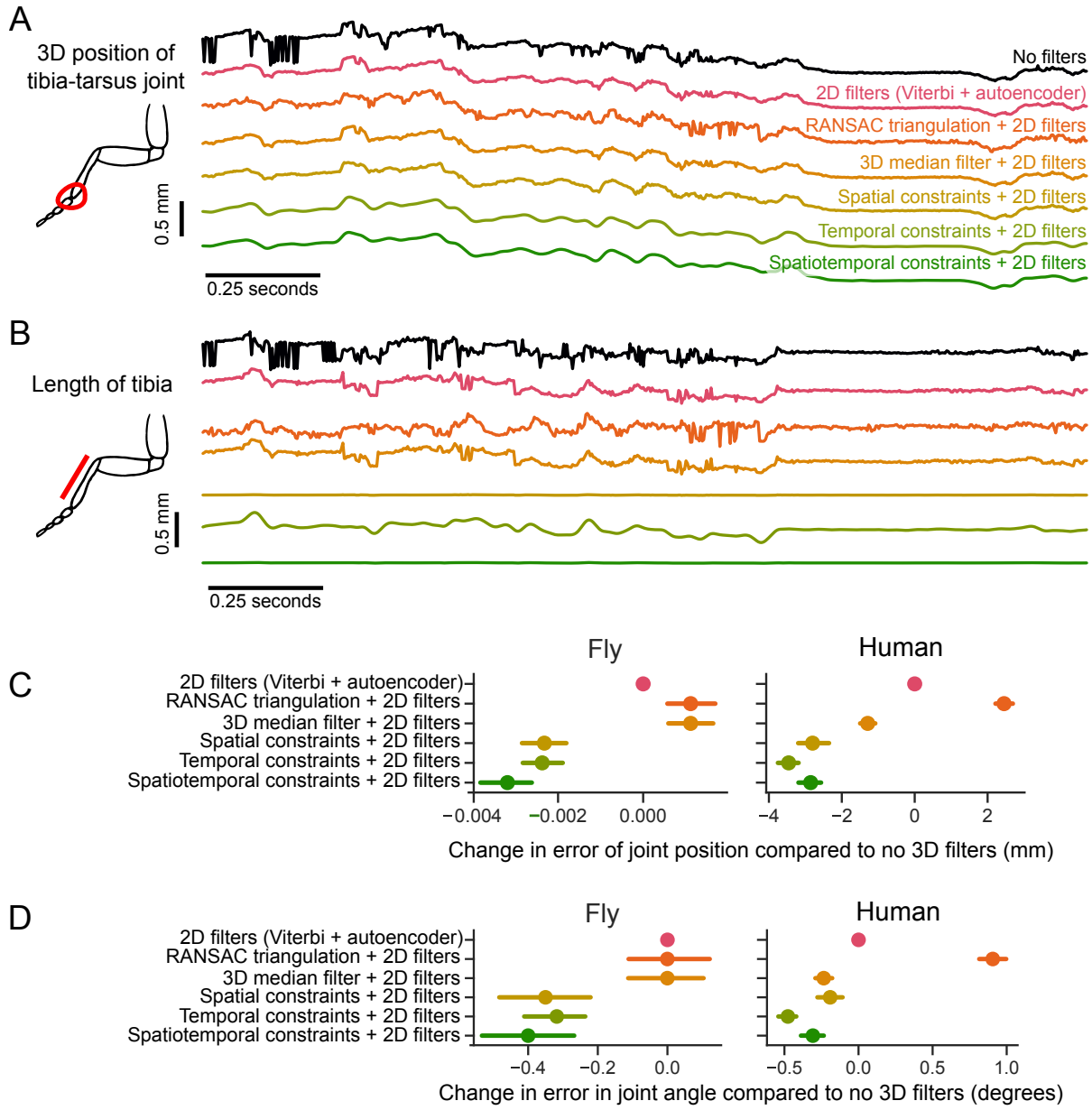


Figure 7: Spatiotemporal filters further improve 3D pose estimation. (A) An example trace of the tracked 3D position of the fly tibia-tarsus joint, before and after filtering. To plot a single illustrative position value, the 3D x-y-z coordinate is projected onto the longitudinal axis of the fly. Also included are comparisons with standard 3D filtering algorithms RANSAC and a 3D median filter. Filtering leads to reduction of sudden jumps and keypoint jitters, even compared to 2D filters alone. (B) Estimation of tibia length over time, before and after filtering. Adding spatial constraints leads to a more stable estimate of the tibia length across frames. (C) Comparison of error in joint position before and after filtering. The mean difference in error for the same tracked points is plotted, along with the 95% confidence interval. Spatiotemporal constraints improve the estimation of joint position significantly above 2D filters in both datasets ($p < 0.001$, paired t-test). The 3D median filter improves pose estimation on the human dataset ($p < 0.001$) but not on the fly dataset. RANSAC triangulation does not improve pose estimation for either dataset. For the fly dataset, 1 pixel corresponds to 0.0075 mm. For the human dataset, 1 pixel corresponds to 4.8 mm. (D) Comparison of angle estimates before and after filtering. The mean difference and confidence intervals are plotted as in C. Spatial and temporal constraints improve angle estimation above 2D filters on both datasets ($p < 0.001$, paired t-test). The 3D median filter improves angle estimation on the human dataset ($p < 0.001$) but not on the fly dataset ($p > 0.8$). RANSAC triangulation does not improve angle estimation for either dataset.

isms, and we achieve accurate reconstruction of 3D joint positions and angles. Individual aniposelib functions can be used for testing on small datasets, and the full Anipose pipeline is designed to streamline structured processing of videos recorded in high-throughput experiments. To help new users get started, we provide tutorials for both the Anipose pipeline and aniposelib at `anipose.org` .

**Relationship to existing markerless tracking tools** Deep learning has led to dramatic improvements in human pose tracking, with toolboxes like OpenPose [12], AlphaPose [21], and DeeperCut [22] allowing researchers to track human pose reliably from 2D video. These methods have also been adapted for markerless tracking in other animals [1]. Current deep-learning based tools for 2D markerless tracking include DeepLabCut [9], LEAP Estimates Animal Pose [11], and DeepPoseKit [10]. DeepFly3D [23] is capable of accurate 3D pose estimation, but it is primarily designed for tracking the limbs of tethered fruit flies and not readily adapted for use in different behavioral setups. Anipose is unique in that it builds upon existing 2D tracking methods to achieve accurate and high-throughput 3D pose estimation. This design provides flexibility for users who want to try out 3D tracking in their existing behavioral setups. Although we designed Anipose to leverage 2D tracking with DeepLabCut [9], it can be compatible with other 2D markerless tracking methods, including SLEAP [11] and DeepPoseKit [10].

### 3.1 Impact of robust markerless 3D tracking

Many key insights in behavioral neuroscience have been gained through carefully controlled behavioral paradigms. In particular, experiments are often designed to accommodate the practical limitations of movement tracking, recording neural activity, and perturbing the animal in real time (e.g., [24, 25, 26, 27, 28]). Recent advances in experimental technologies (e.g., high-density extracellular recording probes [29], optical imaging of fluorescent reporters [30, 31], and optogenetics [32]) have made it feasible to record and perturb neural activity in freely behaving animals in three dimensions. Complementing these technologies, a comprehensive toolbox for high-throughput 3D tracking will not only enable deeper analysis of current experiments, but also make it possible to study more natural behaviors.

A robust 3D markerless tracking solution could also greatly expand the accessibility of quantitative movement analysis in humans. Many neurological disorders, even those commonly thought of as cognitive disorders, affect walking gait [33, 34] and upper limb coordination [35, 36]. Many clinicians and basic researchers currently rely on qualitative evaluations or expensive clinical systems to diagnose motor disorders and assess recovery after treatment. While clinical approaches are commercially available [37], they are costly, require proprietary hardware, rely on the addition of markers to the patient, and cannot assess walking gait in natural contexts such as a patient's home. Anipose could be used as a tool in the diagnosis, assessment, and rehabilitative treatment of movement and neurodegenerative disorders.

### 3.2 Impact of novel filters

When analyzing the results of raw (unfiltered) markerless tracking, we found that outlier keypoint detections had a large impact on overall accuracy. Anipose removes these outliers through filtering and interpolation from other camera views. The resulting improvements in tracking smoothness make it easier to analyze and model kinematic data. Specifically, interpolated data enables the user to obtain better estimates of behavior statistics, such as mean and variance, and to run dimensionality reduction techniques, such as principal components analysis . Additionally, temporal smoothing reduces noise in the first derivative and thus enables the user to obtain more precise estimates of movement speed.

### 3.3 Limitations and practical recommendations

There are several scenarios under which Anipose fails to produce accurate 3D tracking. Below, we enumerate some of the scenarios we have encountered in applying Anipose on different datasets and suggest strategies for troubleshooting.

As is the case for any tracking system, the ability of Anipose to track and estimate body pose is fundamentally limited by the quality of the underlying data. High quality videos are well illuminated, contain minimal motion blur, and provide coverage of each keypoint from different views. A common failure mode we encountered was when the neural network misplaced 2D keypoints in some frames. If the errors are uncorrelated across camera views, then the Anipose filters can compensate and still produce accurate tracking in 3D. But in some cases, multiple views have correlated errors or these errors persist in time. These type of errors most commonly arise when the neural network has not been trained on a subset of rare behaviors, so that the animal adopts poses unseen by the trained network. One solution to reducing the frequency of such errors involves systematically identifying outlier frames, manually relabeling them, then retraining the network. Anipose supports this functionality, as do many other tracking toolboxes [9, 11, 10, 23].

Poor multi-camera calibration also results in tracking errors. A good calibration should have an average reprojection error of the calibration points of less than 3 pixels, and ideally less than 1 pixel. To obtain a quality calibration, the calibration videos should be recorded so that the board is clearly visible from multiple angles and locations on each camera. If it is not possible to achieve this, we suggest exploring a preliminary calibration module in Anipose that operates on detected points on the animal itself. This module was inspired by DeepFly3D [23] but uses our iterative calibration procedure.

We recommend users start with no filters to first evaluate the quality of the tracking. If outliers or missing data impede data analysis, then we recommend enabling the default filter parameters in Anipose, which we have found to produce good results for multiple datasets. In some

cases, some additional tuning of parameters may be required, especially on datasets with unique constraints or when studying behaviors with unusual dynamics.

### 3.4 Outlook

3D markerless tracking and pose estimation are rapidly emerging technologies. We expect this task is well poised to leverage future innovations in machine learning and computer vision. Current machine-learning based tools for animal pose estimation operate on single frames of video and ignore temporal and spatial information. These tools could be improved by learning temporal priors, or taking advantage of end-to-end neural networks that fit models in 3D directly from multi-view images using spatial priors [38, 39]. Beyond tracking single animals, toolboxes like SLEAP [11] and OpenPose [12] have some support for multi-animal pose estimation in 2D. To disambiguate animals in 3D, it may be possible to extend associative embeddings [40] to multiple views, though an entirely new approach may be needed. Further, the availability of large kinematics datasets presents new challenges for user interfaces, data visualization, and analysis. Innovations may involve building on VIA [41, 42] or Napari [43] to enable visualization of animal behavior in concert with synchronized kinematics. In addition to our own ongoing efforts to improve its functionality, we welcome community contributions to improving and extending the Anipose toolkit.

## 4 Methods

### 4.1 Video collection and annotation

**ChArUco dataset.** To evaluate the performance of Anipose compared to physical ground truth, we collected videos of a precision-manufactured ChArUco board [20]. The ChArUco board was manufactured by Applied Image Inc (Rochester, NY) with a tolerance of 2 $\mu$m in length and $2^\circ$ in angle. It is a 2 mm $\times$ 2 mm etching of opal and blue chrome, on a 5 mm $\times$ 5 mm board. The ChArUco pattern itself has 6 $\times$ 6 squares, with 4 bit markers and a dictionary size of 50 markers. With these parameters, the size of each marker is 0.375 mm and the size of each square is 0.5 mm. We filmed the ChArUco board from 6 cameras (Basler acA800-510$\mu$m) evenly distributed around the board (Figure 1A), at 30Hz and with a resolution of 832 x 632 pixels, for 2-3 minutes each day over 2 separate days. While filming, we manually rotated the ChArUco board within the field of view of the cameras. These videos were used as calibration videos for both the ChArUco dataset and the fly dataset detailed below.

We chose 9 of the corners as keypoints for manual annotation and detection (Figures 1A and 4A). We extracted and manually annotated 200 frames from each camera from day 1, and an additional 200 cameras per camera from day 2 (1200 frames per day, 2400 frames total). We used the frames for day 1 for training the neural network and the frames from day 2 for evaluation of all methods.

**Mouse dataset.** Reaching data were obtained from four adult C57BL/6 mice ($\sim$8-12 weeks old, two male and two female) trained on a goal-directed reaching task. Procedures performed in this study were conducted according to US National Institutes of Health guidelines for animal research and were approved by the Institutional Animal Care and Use Committee of The Salk Institute for Biological Studies. The reaching task is as previously described [44]. Briefly, the training protocol consisted of placing the mouse in a 20 cm tall $\times$ 8.5 cm wide $\times$ 19.5 cm long clear acrylic box with an opening in the front of the box measuring 0.9 cm wide and 9 cm tall. A 3D-printed, 1.8 cm tall pedestal designed to hold a food pellet (20 mg, 3 mm diameter; Bio-Serv) was placed 1 cm away from the front of the box opening and displaced to the right by 0.5 cm (to encourage mice to use their left forelimbs), and food pellets were placed on top as the reaching target (Fig. 1B). Mice were food deprived to $\sim$85% of their original body weight and trained to reach for food pellets for either 20 minutes or until 20 successful reaches (defined as pellet retrieval) were accomplished. Mice were trained in this setup for 14 consecutive days before reaches were captured with 2 cameras (Sentech STC-MBS241U3V with Tamron M112FM16 16mm lens) placed in front and to the side of the mouse ($\sim 85^\circ$ apart). Videos were acquired at a frame rate of 200 Hz at a resolution of 1024 $\times$ 768 pixels.

We chose 6 points on the mouse hands as keypoints (Figure 1B). On each mouse hand, we labeled 3 points: the dorsal wrist, the base of digit 5, and the proximal end of digit 3. In total, we manually labeled 2200 frames (1100 frames per camera) for training the neural network from 2 mice. For test data to evaluate the post estimation performance, we labeled an additional 400 frames (200 frames per camera) taken from videos of 2 mice that were not in the training set.

**Fly dataset.** Male and female Berlin wild type *Drosophila melanogaster* 4 days post-eclosion were used for all experiments. Flies were reared on standard cornmeal agar food on a 14 hr/10 hr light-dark cycle at 25 °C in 70% relative humidity. The flies' wings were clipped 24–48 hours prior to the experiment in order to increase walking and prevent visual obstruction of the legs and thorax. For all experiments, a tungsten wire was tethered to the dorsal thorax of a cold-anesthetized fly with UV cured glue. Flies were starved with access to water for 2-5 hours before they were tethered. After 20 minutes of recovery, tethered flies were positioned on a frictionless spherical treadmill [45, 46] (hand-milled foam ball, density: 7.3 mg/mm$^3$, diameter: 9.46 mm) suspended on a stream of compressed air (5 L/min). Six cameras (imaging at 300 Hz, Basler acA800-510$\mu$m with Computar zoom lens MLM3X-MP) were evenly distributed around the fly, providing full video coverage of all six legs (Figure 1C). Fly behavior was recorded in 2 second trials every 25 seconds for 30 minutes, capturing a range of behaviors such as walking, turning, grooming, and pushing against the ball. The recording region of each video was cropped slightly so that the fly filled the frame and the camera was able to acquire at 300Hz.

We chose 30 points on the fly body as keypoints (Figure 1C). On each leg, we labeled 5 points: the body-coxa, coxa-femur, femur-tibia, and tibia-tarsus joints, as well as the tip of the tarsus. In total, we manually labeled 6632 frames (about 1105 frames per camera) for training the neural network. For test data to evaluate the post estimation performance, we labeled an additional 1200 frames (200 frames per camera) taken from videos of 5 flies that were not in the training set.

**Human dataset.** To compare our tracking method to existing computer vision methods, we evaluated the accuracy of Anipose on the Human 3.6M dataset [47, 48]. In this dataset, Ionescu et al. [48] filmed 11 professional actors performing a range of actions, including greeting, posing, sitting, and smoking. The actors were filmed in a 4m $\times$ 3m space with 4 video cameras (Basler piA1000) at a resolution of 1000 $\times$ 1000 pixels at 50Hz (Figure 1D). To gather ground-truth pose data, the actors were also outfitted with reflective body markers and tracked with a separate motion capture system, using 10 Vicon cameras at 200 Hz. Leveraging these recordings, the authors derived the precise 3D positions of 32 body joints and their 2D projections onto the videos. For camera calibration, we used the camera parameters from the Human 3.6M dataset, converted by Martinez et al. [49].

To compare the performance of Anipose against previous methods, we used a protocol from the literature [38]. Specifically, we used 17 of the 32 provided joints as keypoints (Figure 1D). The Human 3.6M dataset contains data from 5 subjects as a training dataset (2 female and 3 male), 2 subjects as a validation dataset, and 4 subjects as a testing dataset (2 female and 2 male). We used frames from the training dataset to train the network and evaluated the predictions on the validation dataset. We also removed frames from the training dataset in which the subject did not move relative to the previous frame ($< 40$mm movement of all joints from the previous frame). We evaluated the tracked human dataset on every 64th frame. Iskakov et al. [38] showed that some scenes from the S9 validation actor (parts of the Greeting, SittingDown, and Waiting actions) have ground-truth shifted in global coordinates compared to the actual position [38], so we exclude these scenes from the evaluation set. Furthermore, for subject S11, one of the videos is corrupted (part of the "Directions" action), so we exclude this from the dataset as well. In total, we obtained 636,724 frames (159,181 per camera) for training the neural network, and 8608 frames (2152 per camera) frames for evaluation.

**Manual annotation of datasets.** To produce neural network training data, we annotated the fly dataset using a mixture of Fiji [50] and the VGG Image Annotator (VIA) [41, 42]. All the images in the fly test set were annotated with VIA. We annotated all the images in the ChArUco dataset and mouse dataset with VIA.

### 4.2 Neural network keypoint detections

Detection of keypoints in each of the datasets was performed with DeepLabCut 2.1.4 [17]. Briefly, to produce training data, we used k-means clustering to pick out unique frames from each of the views, then manually annotated the keypoints in each frame. We trained a single Resnet-50 [51] network for all camera views for the fly, mouse, and ChArUco datasets, starting from a network pretrained on Imagenet. For the human dataset, we started with a Resnet-101 network pretrained on the MPII human pose dataset [22]. During training, we augmented the training dataset with cropping, rotation, brightness, blur, and scaling augmentations using Tensorpack [52]. We then used the Anipose pipeline to run the network on each video. For each keypoint, the network produced a list of predicted positions, each associated with a confidence score (between 0 and 1). We saved the top-n most likely predictions of each joint location for each frame for use in Viterbi filtering of likely keypoints in 2D, as described below.

### 4.3 Filtering of 2D keypoint detections

The raw keypoint detections obtained with DeepLabCut were often noisy or erroneous (Figure 6). Thus, filtering the detections from each camera was necessary before triangulating the points. Anipose contains 3 main algorithms to filter keypoint detections; we elaborate on each algorithm below. Example applications of these filters and results are compared in Figure 6.

**Median filter.** The first algorithm identifies outlier keypoint detections by comparing the raw detected trajectories to median filtered trajectories for each joint. We started by computing a median filter on the detected trajectory for each joint's x and y positions, which smooths the trajectory estimate. We then compared the offset of each point in the raw trajectory to the median filtered trajectory. If a point deviated by some threshold number of pixels, then we denoted this point as an outlier and remove it from the data. The missing points are then interpolated by fitting a cubic spline to the neighboring points. The median filter is simple and intuitive, but it cannot correct errors spanning multiple frames.

**Viterbi filter.** To correct for errors that persist over multiple frames, we implemented the Viterbi algorithm to obtain a single most consistent path in time from the top-n predicted keypoints in each frame for each joint. To be specific, we expressed this problem as a hidden Markov model for each joint, wherein the possible values at each frame are the multiple possible detections of this keypoint. To obtain a cleaner model, we removed duplicate detections (within 7 pixels of each other) within each frame. To compensate for missed detected keypoints over many frames, we augmented the possible values at each frame with all detections up to $F$ previous frames, weighted in time elapsed by multiplying their probability $2^{-F}$. We then identified the best path through the hidden Markov model using the Viterbi algorithm [53]. This procedure estimates a consistent path, even with missed detections of up to $F$ frames.

**Autoencoder filter.** We found that the network would often try to predict a joint location even when the joint was occluded in some view. This type of error is particularly problematic when used in subsequent 3D triangulation. Frustratingly, the confidence scores associated with these predictions can be very high, making them difficult to distinguish from correct, high-confidence predictions. To remove these errors, inspired by [54], we implemented a neural network that takes in a set of confidence scores from all keypoints in one frame, and outputs a corrected set of confidence scores. To generate a training set, we made use of the fact that human annotators do not label occluded joints but label all of the visible joints in each frame. Thus, we generated artificial scores from biased distributions to mimic what the neural network might predict for each frame, with visible joints given a higher probability on average. The task of the network is to predict a high score for each joint that is truly visible in that frame and a low score for any occluded joint. We train a multilayer perceptron network with a single hidden layer to perform this task, using the scikit-learn library [55].

### 4.4 Calibration of multiple cameras.

**Camera model.** A camera captures 2D images of light reflecting from 3D objects; thus, we can think of each camera as a projection, transforming 3D vectors to 2D vectors. To establish our notation, for a point $\boldsymbol{p} = (x, y, z)^T$ or $\boldsymbol{u} = (x, y)^T$, we use a tilde to denote that point in homogeneous coordinates (with a 1 at the end), so that $\tilde{\boldsymbol{p}} = (x, y, z, 1)^T$ or $\tilde{\boldsymbol{u}} = (x, y, 1)^T$.

A camera model specifies a transformation from a 3D point $\tilde{\boldsymbol{p}}$ to a 2D point $\tilde{\boldsymbol{u}}$. We use the camera model described by Zhang [56], which consists of a product of an intrinsics matrix $\mathbf{A}$, an extrinsics matrix $\mathbf{P}$, and a distortion function $\mathcal{D}$.

The extrinsics matrix $\mathbf{P} \in \mathbb{R}^{4\times3}$ describes how the camera is positioned relative to the world. We represent $\mathbf{P}$ as the product of a rotation matrix and a translation matrix. Both rotations and translations may be fully specified with 3 parameters each, for 6 parameters total in $\mathbf{P}$.

The intrinsics matrix $\mathbf{A} \in \mathbb{R}^{3\times3}$ describes the internal coordinate system of the camera. It is often modeled using 5 parameters: focal length terms $f_x$ and $f_y$, offset terms $c_x$ and $c_y$, and a skew parameter $s$:

$$A = \begin{bmatrix} f_x & s & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 0 \end{bmatrix}.$$

In practice, we found that we obtain a more robust calibration by reducing the number of parameters, setting $f = f_x = f_y$, $s = 0$, and $(c_x, c_y)$ to be at the center of the image, so that we need to estimate only the focal length parameter $f$ for the intrinsics matrix.

The distortion function models nonlinear distortions in the camera pixel grid. This distortion is typically modeled with 3 parameters as

$$\mathcal{D}([x, y]) = \begin{bmatrix} x + x\left(k_1(x^2+y^2) + k_2(x^2+y^2)^2 + k_3(x^2+y^2)^4\right) \\ y + y\left(k_1(x^2+y^2) + k_2(x^2+y^2)^2 + k_3(x^2+y^2)^4\right) \end{bmatrix}.$$

In practice, we found that the higher-order distortion terms $k_2$ and $k_3$ are often small for modern cameras, so we assume $k_2 = k_3 = 0$ and only estimate a single parameter $k_1$.

Thus, the full mapping may be written as

$$\tilde{\boldsymbol{u}} = \mathcal{D}(\mathbf{A}\mathbf{P}\tilde{\boldsymbol{p}}).$$

In total, the camera model involves estimating 8 parameters per camera: 6 for extrinsics, 1 for intrinsics, and 1 for distortion.

For the camera calibration and triangulation methods described below, we define the projection $\mathcal{T}$ from $\tilde{\boldsymbol{p}}$ to $\tilde{\boldsymbol{u}}$ as

$$\mathcal{T}(\tilde{\boldsymbol{p}}, \boldsymbol{\theta}_c) = \tilde{\boldsymbol{u}} = \mathcal{D}(\mathbf{A}\mathbf{P}\tilde{\boldsymbol{p}}),$$

where $\boldsymbol{\theta}_c$ are the 8 parameters for the camera model of camera $c$.

**Initial estimate of camera parameters.** In order to calibrate the cameras and estimate parameters of the camera models, we start by obtaining an initial estimate of the camera parameters. We detected calibration board keypoints in videos simultaneously captured from all cameras. We then initialized intrinsics based on these detections following the algorithm from Zhang [56]. We initialized the distortion coefficients to zero.

We developed the following method to initialize camera extrinsics from arbitrary locations. For each pair of cameras, the number of frames in which the board is seen simultaneously is counted and used to build a graph of cameras. To be specific, each node is a camera, and edges represent pairs of cameras whose relation we will use to seed the initialization.

The greedy graph construction algorithm is as follows. Start with the pair of cameras for which the number of frames the board is simultaneously detected is the largest, connect the two camera nodes with an edge. Next, proceed with iterations in decreasing order of the number of boards simultaneously detected. At each iteration, if the two nodes (cameras) are not already connected through some path, connect them with an edge. Processing iteratively through all pairs of cameras in this manner, a graph of camera connectivity is produced. Full 3D calibration is possible if and only if the graph is fully connected.

To initialize the extrinsics using this graph, we start with any camera and set its rotation and translation to zero. Then, we initialize its neighbors from the estimated relative pose of the calibration board between them using the initial intrinsics. This procedure is continued recursively until all cameras are initialized. A diagram of the camera initialization for an example dataset is provided in Figure 8.

**Bundle adjustment.** To refine the camera parameters from initial estimates, we performed a bundle adjustment by implementing a nonlinear least-squares optimization to minimize the reprojection error [19]. Given all $\tilde{\boldsymbol{u}}_{c,j,t}$, the detected $j^{\text{th}}$ keypoints from the calibration board at cameras $c$ in frames $t$, we solve for the best camera parameters

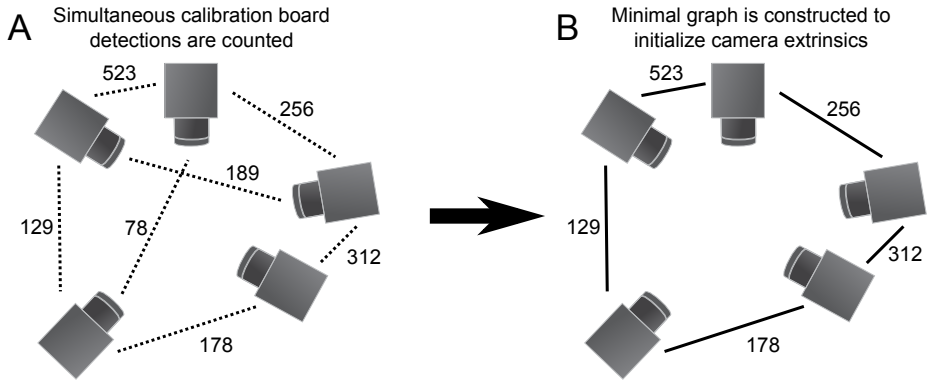


Figure 8: Illustration of the camera parameter initialization procedure. (A) The calibration board is detected simultaneously in some number of frames for each pair of cameras. Based on these simultaneous detections, we build a graph with edge weights being the number of frames. (B) We build a new fully connected but minimal graph (a tree) using a greedy approach.

$\boldsymbol{\theta}_c$ and 3D points $\tilde{\boldsymbol{p}}_{j,t}$ such that the reprojection loss $\mathcal{L}$ is minimized:

$$\mathcal{L} = \sum_c \sum_j \sum_t E\left(\tilde{\boldsymbol{u}}_{c,j,t} - \mathcal{T}(\tilde{\boldsymbol{p}}_{j,t}, \boldsymbol{\theta}_c)\right).$$

Here, $E(\cdot)$ denotes the norm using which the error is computed. This norm may be the least squares norm, but in practice, we used a robust norm, such as the Huber or soft $\ell_1$ norm, to minimize the influence of outliers.

This optimization is nonlinear because of the camera projection function $\mathcal{T}$. We recognized that it is a nonlinear least-squares problem with a sparse Jacobian and thus solved it efficiently using the Trust Region Reflective algorithm [57, 58], as implemented in SciPy [59].

**Iterative bundle adjustment.** When calibrating cameras, we found that outliers have an outsized impact on calibration results, even when using robust losses such as the Huber loss or soft $\ell_1$ loss. Thus, we designed an iterative calibration algorithm, inspired by the fast global registration algorithm from Zhou et al. [60], which solves a minimization with a robust loss efficiently through an alternating optimization scheme.

We approximate this alternating optimization in the camera calibration setting through an iterative threshold scheme. In our algorithm, at each iteration, a reprojection error threshold is defined and the subset of points $\boldsymbol{u}_{c,i}$ with reprojection error below this threshold is chosen. Bundle adjustment is then performed on these points alone. The threshold decreases exponentially with each iteration, to refine the points to be calibrated. The pseudocode for the algorithm is listed in Algorithm 1.

### 4.5 Triangulation and 3D filtering

The 3D triangulation task seeks 3D points $\boldsymbol{p}_{j,t}$ for joint $j$ at frame $t$, given a set of detected 2D points $\boldsymbol{u}_{c,j,t}$ from cameras $c$ with camera parameters $\boldsymbol{\theta}_c$. There are several common methods for solving this triangulation task. Below, we describe 3 of these methods, then describe our method for spatiotemporally constrained triangulation.

**Algorithm 1** Iterative bundle adjustment

**Input:**
Initial camera parameters $\boldsymbol{\theta}$
Keypoint detections $\boldsymbol{u}$ from multiple cameras
Starting and ending thresholds $\mu_{\text{start}}$ and $\mu_{\text{end}}$

**for** $i \leftarrow 1$ to $N_{\text{iter}}$ **do**
  $\boldsymbol{u}_{\text{eval}} \leftarrow \text{sample}(\boldsymbol{u})$
  $\textbf{errors}_{\text{eval}} \leftarrow \text{reprojection\_errors}(\boldsymbol{u}^{\text{eval}}, \boldsymbol{\theta})$
  $\textbf{low} \leftarrow \text{percentile}(\textbf{errors}_{\text{eval}}, 15\%)$
  $\textbf{high} \leftarrow \text{percentile}(\textbf{errors}_{\text{eval}}, 75\%)$
  $\mu_i \leftarrow \left(\frac{\mu_{\text{end}}}{\mu_{\text{start}}}\right)^{i/N_{\text{iter}}}$
  $\mu_i \leftarrow \max(\textbf{low}, \min(\mu_i, \textbf{high}))$
  $\mu_{\text{picked}} \leftarrow$ points from $\boldsymbol{u}_{\text{eval}}$ for which reprojection error is below $\mu_i$
  $\boldsymbol{\theta} \leftarrow \text{bundle\_adjust}(\boldsymbol{\theta}, \boldsymbol{u}_{\text{picked}})$
**end for**
**return** $\boldsymbol{\theta}$

For illustration, a comparison of the performance of these methods is shown on an example dataset in Figure 7.

**Linear least-squares triangulation.** The first method triangulates 3D points by using linear least-squares [61]. Linear least-squares is the fastest method for multi-camera triangulation, but it may lead to poor results when the 2D inputs contain noisy or inaccurate keypoint detections. To be specific, we start with a camera model with parameters estimated from the calibration procedure described above, so that the extrinsics matrix $\mathbf{P}_c$, intrinsics matrix $\mathbf{A}_c$, and distortion function $\mathcal{D}_c$ are known for each camera $c$. By rearranging the camera model, we may write the following relationship:

$$\mathcal{D}_c^{-1}(\tilde{\boldsymbol{u}}_{c,j,t}) = \mathbf{A}_c \mathbf{P}_c \tilde{\boldsymbol{p}}_{j,t}.$$

We solved this linear system of equations using the singular value decomposition (SVD) of the product $\mathbf{A}_c\mathbf{P}_c$ to approximate the solutions for the unknown $\tilde{\boldsymbol{p}}_{j,t}$ [61].

**Median-filtered least-squares triangulation.** As a simple extension to least-square triangulation to correct some of the noisy detections, we applied a median filter to the resulting 3D points tracked across frames. This filtering improves the tracking, but at the cost of losing high frequency dynamics. Furthermore, a median filter does not improve triangulation if the original tracking is consistently poor.

**RANSAC triangulation.** Random sample consensus (RANSAC) triangulation aims to reduce the influence of outlier 2D keypoint detections on the triangulated 3D point, by finding the subset of keypoint detections that minimizes the reprojection error. We implemented RANSAC triangulation by triangulating all possible pairs of keypoints detected from multiple views and picking the resulting 3D point with the smallest reprojection error.

Formally, let $\tilde{\boldsymbol{p}}_{j,t}^{a,b}$ be the triangulated 3D point for keypoint $j$ at frame $t$ computed using the 2D keypoint detections from cameras $a$ and $b$, then our algorithm finds $\tilde{\boldsymbol{p}}_{jt}$ using the following relation:

$$\tilde{\boldsymbol{p}}_{j,t} = \underset{\tilde{\boldsymbol{p}}_{j,t}^{a,b}}{\arg\min} \left\| \mathcal{T}\left(\tilde{\boldsymbol{p}}_{j,t}^{a,b}, \boldsymbol{\theta}_a\right) - \tilde{\boldsymbol{u}}_{a,j,t} \right\|_2 + \left\| \mathcal{T}\left(\tilde{\boldsymbol{p}}_{j,t}^{a,b}, \boldsymbol{\theta}_b\right) - \tilde{\boldsymbol{u}}_{b,j,t} \right\|_2 .$$

**Spatiotemporally constrained triangulation.** We developed spatiotemporally constrained triangulation and formulated triangulation as an optimization problem, which allows us to specify soft constraints on the triangulated points. We propose that the points must satisfy three major constraints: (1) the projection of the 3D points onto each camera should be close to the tracked 2D points, (2) the 3D points should be smooth in time, and (3) the lengths of specified limbs in 3D should not vary too much. Each of these constraints may be formulated as a partial loss in the full objective function.

First, the **reprojection loss** is written as

$$L_{\text{proj}} = \sum_c \sum_j \sum_t E\left(\mathcal{T}\left(\tilde{\boldsymbol{p}}_{j,t}, \boldsymbol{\theta}_c\right) - \tilde{\boldsymbol{u}}_{c,j,t}\right).$$

Here, $E(\cdot)$ is a robust norm function such as the Huber or soft-$\ell_1$ norm, to minimize the influence of outlier detections.

Second, the **temporal loss** may be formulated as a total variation norm to minimize the first finite-difference derivative of the 3D trajectory:

$$L_{\text{time}} = \sum_j \sum_t \left\| \tilde{\boldsymbol{p}}_{j,t} - \tilde{\boldsymbol{p}}_{j,(t-1)} \right\|_2 .$$

This penalty may be extended to minimize higher-order (e.g. 2nd or 3rd) finite-difference derivatives, which produces smoother trajectories but has less impact on important high frequency dynamics.

Third, the **limb loss** may be formulated by adding an additional parameter $d_l$ for each limb $l$, defined to consist of joints $j_1$ and $j_2$:

$$L_{\text{limb}} = \sum_{l,j_1,j_2 \in \text{limbs}} \sum_t \left( \frac{\left\| \tilde{\boldsymbol{p}}_{j_1,t} - \tilde{\boldsymbol{p}}_{j_2,t} \right\|_2 - d_l}{d_l} \right)^2 .$$

The limb error is normalized relative to the limb length so that each limb contributes equally to the error.

Given each of the losses above, the overall objective function to minimize may be written as:

$$\mathcal{L} = L_{\text{proj}} + \alpha_{\text{time}} L_{\text{time}} + \alpha_{\text{limb}} L_{\text{limb}}.$$

We solve this sparse nonlinear least-squares problem efficiently using the Trust Region Reflective algorithm [57, 58], as implemented in SciPy [59], similarly to the bundle adjustment optimization. To initialize the optimization, we use linear least-squares triangulation.

The parameters $\alpha_{\text{time}}$ and $\alpha_{\text{limb}}$ may be tuned to adjust the strength of the temporal or limb loss, respectively. Note, however, that the temporal loss is in units of distance, which may be in an arbitrary scale. Thus, to standardize these parameters across datasets, we break down the parameter $\alpha_{\text{time}}$ in terms of a user-tunable parameter $\beta_{\text{time}}$ and an automatically computed scale $\gamma$ such that

$$\alpha_{\text{time}} = \beta_{\text{time}} \gamma.$$

We compute the scale $\gamma$ as

$$\gamma = \frac{N}{\sum_j \sum_t \left\| \tilde{\boldsymbol{p}}_{j,t} - \tilde{\boldsymbol{p}}_{j,(t-1)} \right\|_2},$$

where $\tilde{\boldsymbol{p}}_{j,t}$ is an initial estimate obtained from linear least-squares triangulation. We found that the parameters $\beta_{\text{time}} = 2$ and $\alpha_{\text{limb}} = 2$ work well across a variety of datasets and produced all the results across all four datasets with these parameters.

**Estimating joint angles.** We estimated joint angles from the tracked 3D positions. To compute the joint angle defined by the three 3D points surrounding the joint $(\boldsymbol{p}_i, \boldsymbol{p}_j, \boldsymbol{p}_k)$, where point $\boldsymbol{p}_j$ lies at the joint, the angle $\phi_j$ is

$$\phi_j = \arccos\left( (\boldsymbol{p}_i - \boldsymbol{p}_j) \cdot (\boldsymbol{p}_k - \boldsymbol{p}_j) \right).$$

### 4.6 Evaluation

**Evaluation against physical ground truth.** To evaluate the calibration and triangulation, we compared the accuracy of manual keypoint annotations, neural network keypoint detections, and OpenCV keypoint detections (Figure 4). The ground truth was considered to be known physical length and angles of the ChArUco board. The physical lengths were calculated between all pairs of keypoints by taking the length between the known position of corners. Similarly, the physical angles were estimated between all triplets of non-collinear keypoints. The subpixel OpenCV detections were done using the Aruco module [20]. The manual annotation and neural network methods are detailed above. Given the keypoint detections

from each method, we used linear least-squares triangulation to obtain 3D points and computed angles using the dot product method detailed above. If a keypoint was detected in fewer than 2 cameras at any time, we could not triangulate it and therefore did not estimate the error at that frame.

**Evaluation of 3D tracking error for different filters.** To evaluate the contribution of different 2D and 3D filters, we applied each filter and measured the reduction in error. For the 2D filters, we applied each of the filters (2D median filter, Viterbi filter, and autoencoder filter) and computed the 3D position using linear least-squares triangulation. We could not train the autoencoder filter on the human dataset, as the filter relies on occluded keypoints not being present in the annotated dataset, but due to the nature of the dataset all keypoints are annotated from every view at every frame. We measured the error in joint positions and angles relative to those computed from manual annotations, using the $\ell_2$ norm. To evaluate the effect of the filter addition, as there was a lot of variance in error across points, we computed the difference in error for each point tracked. We treated points with reprojection error above 20 pixels as missing. The procedure for evaluating the 3D filters was similar, except that we compared the error in joint position and angle relative to the error from 3D points obtained with a Viterbi filter and autoencoder filter with linear least-squares triangulation.

## Contributions

PK, BWB, and JCT conceived the project. PK designed, implemented, and evaluated the Anipose toolkit. KR wrote the Anipose documentation and contributed Tensorpack data augmentation to DeepLabCut. ESD collected the ChArUco and fly datasets. ES and EA collected the mouse dataset. PK, BWB, and JCT wrote the paper, with input from ESD, KR, ES, and EA.

## Acknowledgments

We thank Su-Yee Lee for the design of Anipose's logo. We also thank Su-Yee Lee and Chris Dallmann for help in annotating keypoints on flies, John So for help with annotating keypoints on the ChArUco board, and Sam Mak for help with annotating keypoints on mice. We thank Stephen Huston for loan of his calibration board and Julian Pitney for contributing code to check calibration board detections to Anipose. Finally, we thank the Tuthill and Brunton labs and Mackenzie and Alexander Mathis for support, suggestions, and feedback on the manuscript.

PK was supported by a National Science Foundation Graduate Research Fellowship. KLR was supported by fellowships from the University of Washington's Institute for Neuroengineering (UWIN) and Center for Neurotechnology (CNT). ESD was supported by a fellowship from University of Washington's Institute for Neuroengineering. ES was supported by the National Institutes of Health (F31NS115477). EA was supported by the National Institutes of Health (R00 NS088193, DP2NS105555, R01NS111479, and U19NS112959), the Searle Scholars Program, The Pew Charitable Trusts, and the McKnight Foundation. JT was supported by the Searle Scholars Program, The Pew Charitable Trusts, the McKnight Foundation, and National Institutes of Health grants R01NS102333 and U19NS104655. BWB was supported by a Sloan Research Fellowship and the Washington Research Foundation.

## References

[1] Mackenzie Weygandt Mathis and Alexander Mathis. Deep learning tools for the measurement of animal behavior in neuroscience. *Current Opinion in Neurobiology*, 60:1–11, February 2020.

[2] Nidhi Seethapathi, Shaofei Wang, Rachit Saluja, Gunnar Blohm, and Konrad P. Kording. Movement science needs different pose tracking algorithms. *arXiv:1907.10226 [cs, q-bio]*, 2019. arXiv: 1907.10226.

[3] David E. Alexander. *Nature's Machines: An Introduction to Organismal Biomechanics*. Academic Press, 1 edition, 2017.

[4] Mac Schwager, Carrick Detweiler, Iuliu Vasilescu, Dean M. Anderson, and Daniela Rus. Data-driven identification of group dynamics for motion prediction and control. *Journal of Field Robotics*, 25(6-7):305–324, 2008.

[5] Jamin Halberstadt, Joshua Conrad Jackson, David Bilkey, Jonathan Jong, Harvey Whitehouse, Craig McNaughton, and Stefanie Zollmann. Incipient Social Groups: An Analysis via In-Vivo Behavioral Tracking. *PLOS ONE*, 11(3):e0149880, 2016.

[6] Richard B. Souza. An Evidence-Based Videotaped Running Biomechanics Analysis. *Physical Medicine and Rehabilitation Clinics of North America*, 27(1):217–236, February 2016.

[7] Hiroshi Chiba, Satoru Ebihara, Naoki Tomita, Hidetada Sasaki, and James P Butler. Differential gait kinematics between fallers and non-fallers in community-dwelling elderly people. *Geriatrics and Gerontology International*, 5(2):127–134, June 2005.

[8] Nicole J. Rinehart, Mark A. Bellgrove, Bruce J. Tonge, Avril V. Brereton, Debra Howells-Rankin, and John L. Bradshaw. An Examination of Movement Kinematics in Young People with High-functioning Autism and Aspergers Disorder: Further Evidence for a Motor Planning Deficit. *Journal of Autism and Developmental Disorders*, 36(6):757–767, August 2006.

[9] Alexander Mathis, Pranav Mamidanna, Kevin M. Cury, Taiga Abe, Venkatesh N. Murthy, Mackenzie Weygandt Mathis, and Matthias Bethge. DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience*, 21(9):1281, September 2018.

[10] Jacob M Graving, Daniel Chae, Hemal Naik, Liang Li, Benjamin Koger, Blair R Costelloe, and Iain D Couzin. Deepposekit, a software toolkit for fast and robust animal pose estimation using deep learning. *eLife*, 8:e47994, oct 2019.

[11] Talmo D. Pereira, Diego E. Aldarondo, Lindsay Willmore, Mikhail Kislin, Samuel S.-H. Wang, Mala Murthy, and Joshua W. Shaevitz. Fast animal pose estimation using deep neural networks. *Nature Methods*, 16(1):117, 2019.

[12] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. OpenPose: realtime multi-person 2D pose estimation using Part Affinity Fields. In *arXiv preprint arXiv:1812.08008*, 2018.

[13] Ana S Machado, Dana M Darmohray, João Fayad, Hugo G Marques, and Megan R Carey. A quantitative framework for whole-body coordination reveals specific deficits in freely walking ataxic mice. *eLife*, 4:e07892, October 2015.

[14] Kristin Branson. Animal part tracker. `https://github.com/kristinbranson/APT`.

[15] Anthony W. Azevedo, Evyn S. Dickinson, Pralaksha Gurung, Lalanti Venkatasubramanian, Richard Mann, and John C. Tuthill. A size principle for leg motor control in Drosophila. *bioRxiv*, page 730218, 2019.

[16] Alice A. Robie, Jonathan Hirokawa, Austin W. Edwards, Lowell A. Umayam, Allen Lee, Mary L. Phillips, Gwyneth M. Card, Wyatt Korff, Gerald M. Rubin, Julie H. Simpson, Michael B. Reiser, and Kristin Branson. Mapping the Neural Substrates of Behavior. *Cell*, 170(2):393–406.e28, 2017.

[17] Tanmay Nath, Alexander Mathis, An Chi Chen, Amir Patel, Matthias Bethge, and Mackenzie Weygandt Mathis. Using DeepLabCut for 3d markerless pose estimation across species and behaviors. *Nature Protocols*, 14(7):2152–2176, 2019.

[18] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[19] Bill Triggs, Philip F. McLauchlan, Richard I. Hartley, and Andrew W. Fitzgibbon. Bundle adjustment a modern synthesis. In Bill Triggs, Andrew Zisserman, and Richard Szeliski, editors, *Vision Algorithms: Theory and Practice*, volume 1883, pages 298–372. Springer Berlin Heidelberg, 2000.

[20] S. Garrido-Jurado, R. Muñoz-Salinas, F. J. Madrid-Cuevas, and M. J. Marín-Jiménez. Automatic generation and detection of highly reliable fiducial markers under occlusion. *Pattern Recognition*, 47(6):2280–2292, June 2014.

[21] Hao-Shu Fang, Shuqin Xie, Yu-Wing Tai, and Cewu Lu. RMPE: Regional multi-person pose estimation. In *International Conference on Computer Vision (ICCV)*, 2017.

[22] Eldar Insafutdinov, Leonid Pishchulin, Bjoern Andres, Mykhaylo Andriluka, and Bernt Schiele. Deepercut: A deeper, stronger, and faster multi-person pose estimation model. *ECCV*, 2016.

[23] Semih Günel, Helge Rhodin, Daniel Morales, João Campagnolo, Pavan Ramdya, and Pascal Fua. Deepfly3d, a deep learning-based approach for 3d limb and appendage tracking in tethered, adult *Drosophila*. *eLife*, 8:e48571, oct 2019.

[24] Thomas M. Tzschentke. Review on CPP: Measuring reward with the conditioned place preference (CPP) paradigm: update of the last decade. *Addiction biology*, 12(3):227–462, 2007.

[25] Rudi DHooge and Peter P. De Deyn. Applications of the morris water maze in the study of learning and memory. *Brain research reviews*, 36(1):60–90, 2001.

[26] David S. Olton. Mazes, maps, and memory. *American psychologist*, 34(7):583, 1979.

[27] Kristin Branson, Alice A. Robie, John Bender, Pietro Perona, and Michael H. Dickinson. High-throughput ethomics in large groups of drosophila. *Nature Methods*, 6(6):451–457, 2009.

[28] Gordon J. Berman, Daniel M. Choi, William Bialek, and Joshua W. Shaevitz. Mapping the stereotyped behaviour of freely moving fruit flies. *Journal of The Royal Society Interface*, 11(99):20140672, 2014.

[29] James J. Jun, Nicholas A. Steinmetz, Joshua H. Siegle, Daniel J. Denman, Marius Bauza, Brian Barbarits, Albert K. Lee, Costas A. Anastassiou, Alexandru Andrei, Çaatay Aydn, Mladen Barbic, Timothy J. Blanche, Vincent Bonin, João Couto, Barundeb Dutta, Sergey L. Gratiy, Diego A. Gutnisky, Michael Häusser, Bill Karsh, Peter Ledochowitsch, Carolina Mora Lopez, Catalin Mitelut, Silke Musa, Michael Okun, Marius Pachitariu, Jan Putzeys, P. Dylan Rich, Cyrille Rossant, Wei-lung Sun, Karel Svoboda, Matteo Carandini, Kenneth D. Harris, Christof Koch, John OKeefe, and Timothy D. Harris. Fully integrated silicon probes for high-density recording of neural activity. *Nature*, 551(7679):232–236, November 2017.

[30] Hod Dana, Yi Sun, Boaz Mohar, Brad K. Hulse, Aaron M. Kerlin, Jeremy P. Hasseman, Getahun Tsegaye, Arthur Tsang, Allan Wong, Ronak Patel, John J. Macklin, Yang Chen, Arthur Konnerth, Vivek Jayaraman, Loren L. Looger, Eric R. Schreiter, Karel Svoboda, and Douglas S. Kim. High-performance calcium sensors for imaging activity in neuronal populations and microcompartments. *Nature Methods*, 16(7):649–657, July 2019.

[31] Ahmed S. Abdelfattah, Takashi Kawashima, Amrita Singh, Ondrej Novak, Hui Liu, Yichun Shuai, Yi-Chieh Huang, Luke Campagnola, Stephanie C. Seeman, Jianing Yu, Jihong Zheng, Jonathan B. Grimm, Ronak Patel, Johannes Friedrich, Brett D. Mensh,

Liam Paninski, John J. Macklin, Gabe J. Murphy, Kaspar Podgorski, Bei-Jung Lin, Tsai-Wen Chen, Glenn C. Turner, Zhe Liu, Minoru Koyama, Karel Svoboda, Misha B. Ahrens, Luke D. Lavis, and Eric R. Schreiter. Bright and photostable chemigenetic indicators for extended in vivo voltage imaging. *Science*, 365(6454):699–704, 2019.

[32] Jacob G. Bernstein, Paul A. Garrity, and Edward S. Boyden. Optogenetics and thermogenetics: technologies for controlling the activity of targeted cells within intact neural circuits. *Current Opinion in Neurobiology*, 22(1):61–71, February 2012.

[33] Henning Stolze, Stephan Klebe, Christoph Baecker, Christiane Zechlin, Lars Friege, Sabine Pohle, and Günther Deuschl. Prevalence of gait disorders in hospitalized neurological patients. *Movement Disorders*, 20(1):89–94, 2005.

[34] Joanne E. Wittwer, Kate E. Webster, and Hylton B. Menz. A longitudinal study of measures of walking in people with Alzheimer's Disease. *Gait & Posture*, 32(1):113–117, May 2010.

[35] C. Solaro, G. Brichetto, M. Casadio, L. Roccatagliata, P. Ruggiu, G.L. Mancardi, P.G. Morasso, P. Tanganelli, and V. Sanguineti. Subtle upper limb impairment in asymptomatic multiple sclerosis subjects. *Multiple Sclerosis Journal*, 13(3):428–432, April 2007.

[36] William J. Tippett, Adam Krajewski, and Lauren E. Sergio. Visuomotor Integration Is Compromised in Alzheimers Disease Patients Reaching for Remembered Targets. *European Neurology*, 58(1):1–11, 2007.

[37] Markus Windolf, Nils Götzen, and Michael Morlock. Systematic accuracy and precision analysis of video motion capturing systems exemplified on the Vicon-460 system. *Journal of Biomechanics*, 41(12):2776–2780, August 2008.

[38] Karim Iskakov, Egor Burkov, Victor Lempitsky, and Yury Malkov. Learnable triangulation of human pose. In *International Conference on Computer Vision (ICCV)*, 2019.

[39] Yuan Yao, Yasamin Jafarian, and Hyun Soo Park. MONET: Multiview semi-supervised keypoint detection via epipolar divergence. In *International Conference on Computer Vision (ICCV)*, 2019.

[40] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative Embedding: End-to-End Learning for Joint Detection and Grouping. *NIPS*, 2017.

[41] A. Dutta, A. Gupta, and A. Zissermann. VGG image annotator (VIA). `http://www.robots.ox.ac.uk/~vgg/software/via/`, 2016.

[42] Abhishek Dutta and Andrew Zisserman. The VIA annotation software for images, audio and video. In *Proceedings of the 27th ACM International Conference on Multimedia*, MM '19, New York, NY, USA, 2019. ACM.

[43] Nicholas Sofroniew, Kira Evans, Talley Lambert, Juan Nunez-Iglesias, Ahmet Can Solak, kevinyamauchi, Genevieve Buckley, Tony Tung, Jeremy Freeman, Hagai Har-Gil, Peter Boone, Loic Royer, Shannon Axelrod, jakirkham, Reece Dunham, Pranathi Vemuri, Mars Huang, Bryant, Ariel Rokem, Simon Li, Russell Anderson, Matthias Bussonnier, Justin Kiggins, Hector, Heath Patterson, Guillaume Gay, Eric Perlman, Davis Bennett, Christoph Gohlke, and Alexandre de Siqueira. napari/napari: 0.2.12. `https://github.com/napari/napari`, February 2020.

[44] Eiman Azim, Juan Jiang, Bror Alstermark, and Thomas M Jessell. Skilled reaching relies on a v2a propriospinal internal copy circuit. *Nature*, 508(7496):357–363, 2014.

[45] E. Buchner. Elementary movement detectors in an insect visual system. *Biological Cybernetics*, 24(2):85–101, 1976.

[46] K. G. Götz. Visual control of locomotion in the walking fruitfly Drosophila. *Journal of Comparative Physiology*, 85(3):235–266, 1973.

[47] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2014.

[48] Cristian Sminchisescu Catalin Ionescu, Fuxin Li. Latent structured models for human pose estimation. In *International Conference on Computer Vision*, 2011.

[49] Julieta Martinez, Rayat Hossain, Javier Romero, and James J. Little. A simple yet effective baseline for 3d human pose estimation. In *ICCV*, 2017.

[50] Johannes Schindelin, Ignacio Arganda-Carreras, Erwin Frise, Verena Kaynig, Mark Longair, Tobias Pietzsch, Stephan Preibisch, Curtis Rueden, Stephan Saalfeld, Benjamin Schmid, Jean-Yves Tinevez, Daniel James White, Volker Hartenstein, Kevin Eliceiri, Pavel Tomancak, and Albert Cardona. Fiji: an open-source platform for biological-image analysis. *Nature Methods*, 9(7):676–682, July 2012.

[51] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep Residual Learning for Image Recognition. *CVPR*, 2016. arXiv: 1512.03385.

[52] Yuxin Wu et al. Tensorpack. `https://github.com/tensorpack/`, 2016.

[53] David G. Forney. The viterbi algorithm. *Proceedings of the IEEE*, 61(3):268–278, 1973.

[54] Daniel Murphy. *Markerless 3D Pose Estimation from RGB Data*. Bachelor's thesis, Brown University, 2019.

[55] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos,

D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[56] Z. Zhang. A flexible new technique for camera calibration. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1330–1334, 2000.

[57] Richard H. Byrd, Robert B. Schnabel, and Gerald A. Shultz. Approximate solution of the trust region problem by minimization over two-dimensional subspaces. *Mathematical programming*, 40(1):247–263, 1988.

[58] Mary Ann Branch, Thomas F. Coleman, and Yuying Li. A subspace, interior, and conjugate gradient method for large-scale bound-constrained minimization problems. *SIAM Journal on Scientific Computing*, 21(1):1–23, 1999.

[59] Eric Jones, Travis Oliphant, Pearu Peterson, et al. SciPy: Open source scientific tools for Python, 2001–.

[60] Qian-Yi Zhou, Jaesik Park, and Vladlen Koltun. Fast global registration. In *ECCV*. 2016.

[61] Richard I. Hartley and Peter Sturm. Triangulation. *Computer vision and image understanding*, 68(2):146–157, 1997.

# Extra figures

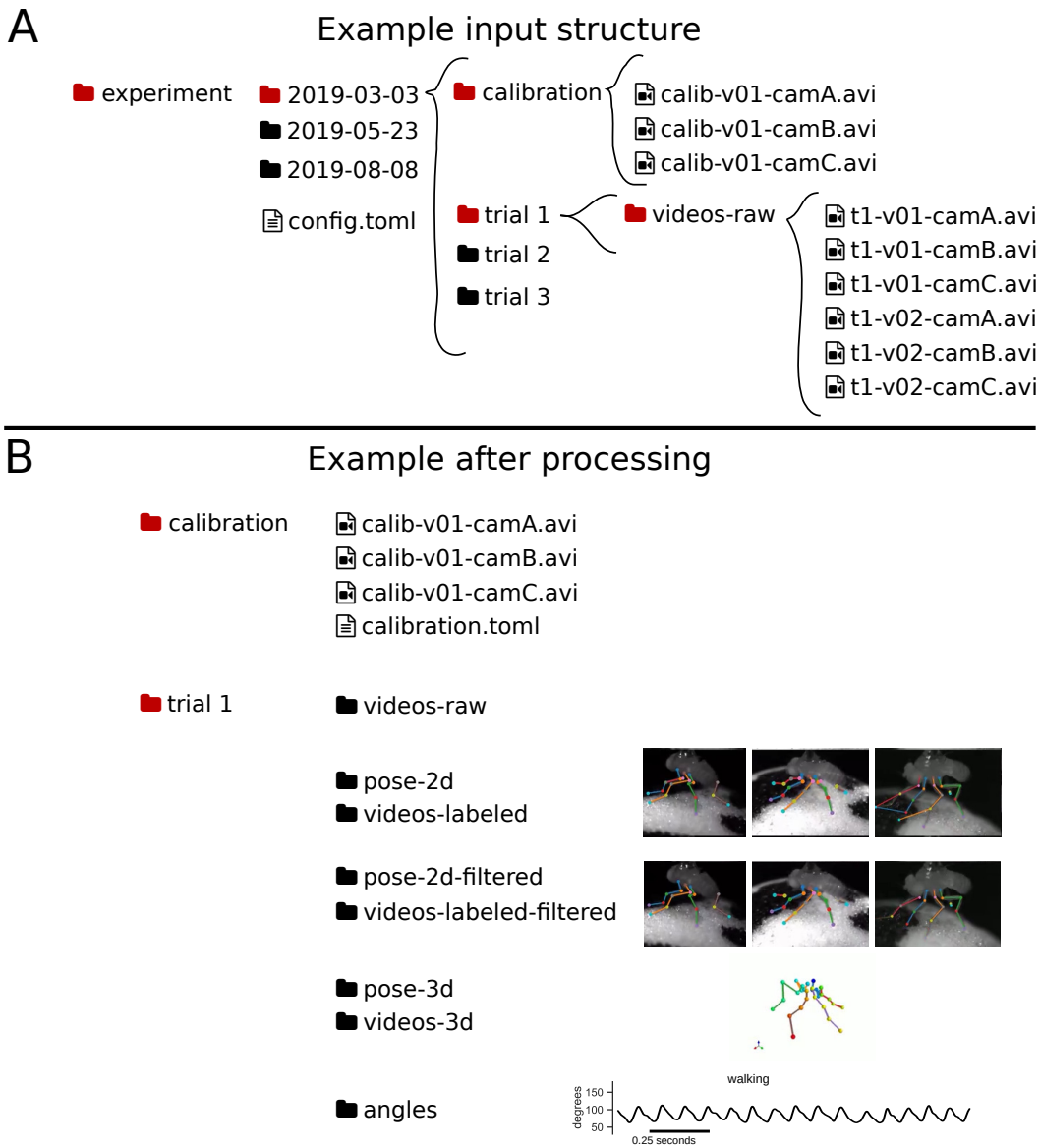


Figure 9: An example of the Anipose file structure. (A) The input file structure consists of folders nested to arbitrary depths (e.g. "experiment/2019-03-03/trial 1") with a folder for raw videos at each leaf of the directory tree. The calibration folder may be placed anywhere and will apply recursively to all folders adjacent to it. (B) When the user runs Anipose, it will create a folder for each step of processing. New folders created include "pose-2d" and "videos-labeled" which contain the unfiltered keypoint detections and visualizations of those, "pose-2d-filtered" and "videos-labeled-filtered" which contain the filtered keypoint detections and visualizations, "pose-3d" and "videos-3d" which contain the triangulated 3D keypoint detections and visualizations of these, and finally "angles" which contains angles computed based on the 3D keypoint detections.

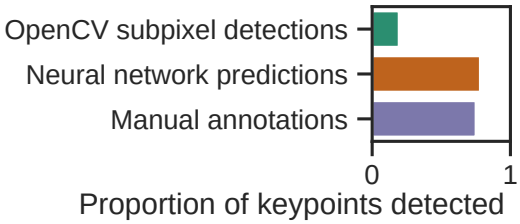


Figure 10: For the ChArUco evaluation dataset, OpenCV detected only a small fraction of the keypoints detected by the neural network or through manual annotation.

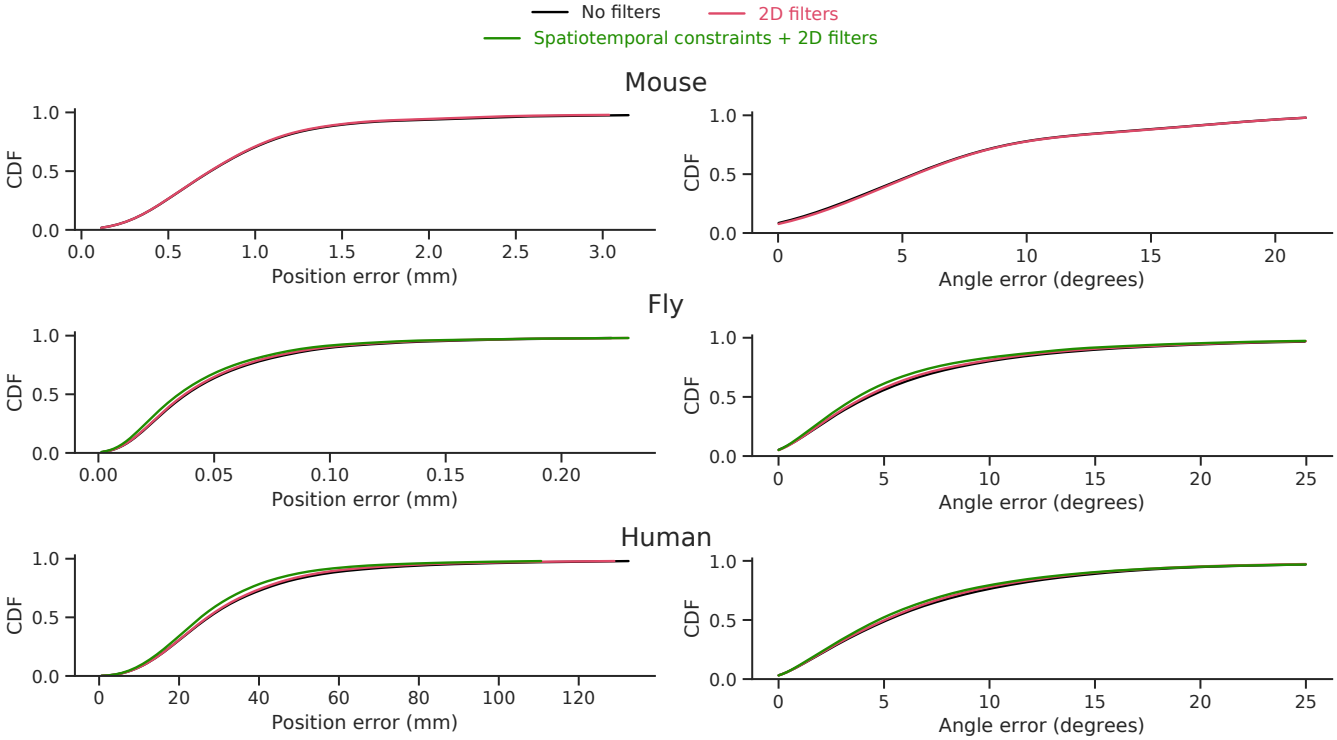


Figure 11: Full cumulative distribution functions of the position and angle error with and without filters for each of the datasets.

| | ChArUco | Mouse | Fly | Human |
|---|---|---|---|---|
| Training frames | 1200 | 2200 | 6632 | 636724 |
| Test frames | 1200 | 400 | 1200 | 159181 |
| Num cameras | 6 | 2 | 6 | 4 |
| Pixel scale (mm) | 0.0075 | 0.0897 | 0.0075 | 4.79 |
| **2D filter** | | | | |
| score threshold | | | 0.05 | 0.05 |
| n_back | | | 3 | 3 |
| medfilt | | | 13 | 13 |
| offset_threshold | | | 15 | 30 |
| spline | | | true | true |
| **3D filter** | | | | |
| score_threshold | 0.3 | 0.3 | 0.3 | 0.3 |
| reproj_error_threshold | | | 5 | 5 |
| scale_length | | | 3 | 1.5 |
| scale_length_weak | | | 0.5 | 0.5 |
| scale_smooth | | | 2 | 4 |
| n_deriv_smooth | | | 3 | 2 |

Table 1: Anipose configuration parameters used in this paper