JOSS
The Journal of Open Source Software

3D reconstruction toolbox for behavior tracked with multiple cameras

Swathi Sheshadri[1, 2], Benjamin Dann[1], Timo Hueser[1], and Hansjoerg Scherberger[1, 2]

1 German Primate Center, Goettingen, Germany **2** Department of Biology and Psychology, University of Goettingen, Germany

Summary

Markerless tracking is a crucial experimental requirement for behavioral studies conducted in many species in different environments. A recently developed toolbox called DeepLabCut (DLC) (Mathis et al. (2018)) leverages Artificial Neural Network (ANN) based computer vision to make precise markerless tracking possible for scientific experiments. DLC uses a deep convolutional neural network, ResNet (He, Zhang, Ren, & Sun (2016)) pre-trained on ImageNet database (Deng et al. (2009)) and adapts it to make it applicable for behavioral tracking tasks. To track complex behaviors such as grasping with object interaction in 3D, experimental setups with multiple cameras are required. Such systems can largely benefit from a robust and easy to use camera calibration and 3D reconstruction toolbox. The current version of DLC allows 3D reconstruction of features tracked in 2D from pairs of cameras only (Nath, Mathis, Chen, Bethge, & Mathis (2019)) and is not sufficient when behavior is tracked with more than 2 cameras. Furthermore, for noisy 2D tracking conditions, such as low light or low resolution, the accuracy of tracked 3D coordinates can be improved by evaluating data from more than two cameras.

To facilitate 3D tracking of complex behaviors requiring multiple cameras (n>=2), we developed pose3d: a Matlab (The MathWorks Inc., Natick, Massachusetts) implementation for 3D reconstruction. pose3d can be divided into two main parts: camera calibration with pairwise estimation of relative camera positions to each other and 3D reconstruction with triangulation performed over 2D feature coordinates tracked from all cameras. Camera calibration refers to the estimation of intrinsic and extrinsic camera parameters including the level of magnification captured by the focal length, the distortion in the camera image caused by the lens and the location of camera in 3D coordinates. In stereo camera calibration, one of the cameras is fixed as the primary camera and the position of the secondary camera with respect to the primary camera is estimated and given as an additional parameter. Stereo camera calibration in pose3d is performed by using checkerboard presented at varied angles to the pair of cameras being calibrated to establish the correspondence between 2D image coordinates and 3D coordinates. Following calibration, the 2D feature tracked across cameras is transformed into 3D coordinates using triangulation. pose3d implements triangulation by minimizing the distance of the estimated 3D feature coordinate from the lines passing through the focal center and the feature tracked in 2D on the image plane of the cameras simultaneously.

pose3d usage and features

To use pose3d for stereo camera calibration and 3D reconstruction the users need only to edit a configuration file to enter their experiment specific details. The configuration file is extensively

commented to help users through the process of editing it. Running the main function of the repository 'main_pose3d.m' with the user's configuration file, automatically creates the folder structure required for stereo camera calibration and 3D reconstruction. Then, pose3d launches Matlab's stereoCameraCalibrator GUI sequentially for every pair of primary and secondary cameras to estimate the parameters of the cameras in the user's experimental setup. The GUI first parses through checkerboard frames recorded simultaneously from the two cameras being calibrated to select frames in which checkerboard can be detected in both cameras. At this point, the user can select the number of distortion coefficients which determines the degree of polynomial used to estimate distortion. If the lens used in the experiment causes higher distortion of images, 3 distortion coefficients can be used instead of the 2 (default) by simply selecting the radio-button on the GUI corresponding to it. After this, clicking the calibrate button on the GUI, estimates camera parameters and relative positioning of secondary camera with respect to the primary. The GUI also displays reprojection errors across calibration frames allowing users to iteratively recalibrate after removing outliers (this can be done by right-clicking on the data browser of the GUI and selecting the option remove and recalibrate). After calibration is complete, the session can be saved (by clicking the save as button). This procedure is then repeated for all camera pairs to be calibrated. pose3d prompts users throughout this procedure, providing messages on the next steps to be carried out.

After calibrating all cameras in the setup with respect to the primary camera, the 2D feature of interest tracked across cameras are corrected for distortion and the 3D coordinates estimated by triangulation. The function for triangulation across 'n' cameras is called 'triangulate_ncams'. This function is available for users in three modes that can be selected based on prior knowledge of the experimental setup. The first mode of triangulation is referred to as 'all' and uses data from all 'n' cameras that are used to track the feature in 2D. The second mode is referred to as 'avg' and performs triangulation over all pairs of cameras in the setup and provides the average over all pairs as the result. The third mode is referred to as 'best-pair' and selects the best camera pair for every time point and feature of interest for triangulation. While the first and second modes can be used for 3D reconstruction of features tracked with any software, the third mode is applicable only when tracked 2D features are associated with likelihood values, e.g., as provided by DLC.

Overall, pose3d offers a semi-automated 3D reconstruction workflow going beyond Matlab's existing functions (details in section titled 'Comparison of pose3d to existing Matlab functions') that takes users through the entire process of camera calibration, undistortion, triangulation, and post processing steps such as filtering to reduce outliers. pose3d also allows users to try different pre- and post-processing parameters and visualizes its effect on the accuracy of 3D reconstruction to help perform manual parameter tuning before saving final results. Other implementations of similar functionality as pose3d exist and use OpenCV (("Open Source Computer Vision Library," 2015)) in python: anipose (Karashchuk (2019)). However, in comparison to OpenCV based implementations, pose3d provides a user-friendly application by integrating with a feature-rich graphical user interface (GUI) for stereo camera calibration in Matlab. A more detailed comparison between pose3d and anipose is included in a subsequent section 'Comparison of pose3d and anipose'. Furthermore, we provide two demo datasets (see next paragraph) illustrating the capabilities of pose3d and a quick-view of the included features. Given the popularity of Matlab in academia and its well documented and easy to use core functions for camera calibration, we believe this toolbox will help make 3D reconstruction of tracked 2D behavior easier to use.

Demo datasets and error measurement in 3D reconstruction

Demo datasets provided in pose3d were acquired by moving a Rubik's cube on a table along different directions and by recording it simultaneously from a 5-camera tracking system. For

the demos, we tracked the corners of the cube using DLC in the first example and manually in the second example. Manual annotations are used to mimic the usage of pose3d for any other 2D tracking software.

The key difference between the two examples is as follows. DLC, in addition to 2D tracking provides users with a likelihood value for every tracked feature that informs the users on how confident the network is about the inferred location of a particular feature of interest at any given time point. pose3d makes use of this information by applying a threshold and automatically selecting the cameras that cross this threshold for 3D reconstruction. From the 2D tracked corners we use pose3d to track corners in 3D for 1000 example frames with DLC and 20 example frames with manual annotations. Following this, we reconstruct the edges of the cube and compare it to the standard edge length of a Rubik's cube (57 mm). In the demo data using DLC based 2D annotations, we obtain on average an error of 1.39 mm in 3D reconstructed edge lengths computed over all 12 edges of the cube across 1000 example frames. For the demo data using manual annotations across 20 frames we obtain an average error of 1.16 mm over all 12 edges computed over 20 manually annotated frames. Furthermore, using 'all' mode of triangulation provided significantly better results in both of our demo datasets than the other two modes of triangulation (comparison tests for the 3 modes of 3D reconstruction included in the demo functions for reference).

Comparison of pose3d and existing Matlab functions

pose3d is a 3D reconstruction workflow integrated closely with functions from Matlab's computer vision toolbox. This workflow fixes one of the cameras in the experimental setup as primary which is crucial to ensure that the data points are within the same coordinate system across camera pairs. The stereo camera calibration part of our implementation can be considered a wrapper around functions from Matlab's computer vision toolbox, however, the triangulation function included with pose3d extends the existing triangulation function in Matlab that operates only for 2 cameras.

Comparison of pose3d and anipose

Functionally, pose3d and anipose provide users with pre- and post-processing tools for 3D reconstruction including the process of camera calibration. The main difference is that pose3d provides a user-friendly GUI utility which is missing in anipose. With pose3d, by entering the experimental details in a configuration file and running a single Matlab function ('main_pose3d.m'), the user can perform the entire procedure of 3D reconstruction. Furthermore, the integration of pose3d with the stereoCameraCalibrator GUI in Matlab makes the task of camera calibration very intuitive. This allows users to estimate camera and lens distortion parameters, select outlier frames from calibration with the click of a button. On the other hand, anipose uses OpenCV functions that can be quite difficult to use for non-experts. For instance, with pose3d, due to its integration with Matlab's stereoCameraCalibrator GUI, all pairs of images used for camera calibration are visualized along with their reprojection errors, which allows easy identification and removal of images with larger error values. With OpenCV functions, users get warning/error messages that are often quite technical in nature and can take longer for non-experts to perform root-cause analysis. In addition, pose3d provides demo datasets including example calibration images along with code to run 3D reconstructions on demo datasets as well as video tutorials to illustrate its operation. Lastly, error calculation routine is included in pose3d to quantify deviation of the reconstructed data from ground truth which is not provided in the current version of anipose. For example, in the demo dataset using Rubik's cube the 12 edges of the cubes were physically measured and compared against the reconstructed edge lengths obtained after running pose3d. Similarly, fixed lengths in the

user's experimental setups that are tracked can be compared against the measured values using the error calculation routine of pose3d.

Some other minor similarities and differences between anipose and pose3d include the following. First, both anipose and pose3d allow users to have the same set of calibration files across sessions or have a separate set per session. This can be decided by the user depending on whether the user wants to have the same calibration files across sessions or not. This is possible in pose3d by having the same experiment path (assigned to variable exp_path) and changing only the experiment name (assigned to variable exp_name) in the configuration file in case the same calibration files are used across sessions. With anipose this is implemented by having a hierarchy for the selection of calibration videos placed in different folders. Second, the triangulation approach used is the same in the two packages. However, with pose3d we have included 3 modes of 3D reconstruction including 'all', 'avg', and 'best-pair'. Of these three modes 'all' corresponds to the one implemented in anipose. With our demo dataset, we have shown that the 'all' mode gives the lowest on average error and is therefore recommended. Third, anipose allows for camera calibration using charuco boards, checker boards or aruco boards of which it recommends the usage of charuco and checker boards. Since pose3d integrates with the camera calibration GUI from MATLAB, only checker board is used for calibration.

For further technical reading on details of triangulation for 3D reconstruction please refer to our supporting document.

References

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., & Fei-Fei, L. (2009). ImageNet: A large-scale hierarchical image database. In *CVPR09*. doi:10.1109/CVPR.2009.5206848

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770–778). doi:10.1109/CVPR.2016.90

Karashchuk, P. (2019). Anipose. *GitHub repository*. https://github.com/lambdaloop/anipose; GitHub. doi:10.5281/zenodo.3364758

Mathis, A., Mamidanna, P., Cury, K. M., Abe, T., Murthy, V. N., Mathis, M. W., & Bethge, M. (2018). DeepLabCut: Markerless pose estimation of user-defined body parts with deep learning. *Nature Neuroscience*, *21*, 1281–1289. doi:10.1038/s41593-018-0209-y

Nath, T., Mathis, A., Chen, A. C., Bethge, M., & Mathis, M. W. (2019). Using DeepLabCut for 3D markerless pose estimation across species and behaviors. *Nature Protocols*. doi:10.1038/s41596-019-0176-0

Open Source Computer Vision Library. (2015). https://opencv.org.