EURASIP Journal on Image and Video Processing

RESEARCH Open Access

A novel method for robust markerless tracking of rodent paws in 3D

Omid Haji Maghsoudi[1]*, Annie Vahedipour[2] and Andrew Spence[3]

Abstract

Studying animal locomotion improves our understanding of motor control and aids in the treatment of motor impairment. Mice are a premier model of human disease and are the model system of choice for much of basic neuroscience. Placement of the tips of appendages, here paws, is typically critical for locomotion. Tracking paws from a video is difficult, however, due to frequent occlusions and collisions. We propose a method and provide software to track the paws of rodents. We use a superpixel-based method to segment the paws, direct linear transform to perform 3D reconstruction, a 3D Kalman filter (KF) to solve the matching problem and label paws across frames, and spline fits through time to resolve common collisions. The automated method was compared to manual tracking. The method had an average of 2.54 mistakes requiring manual correction per 1000 frames with a maximum of 5.29 possible errors while these values were estimates of the expected errors. We present an algorithm and its implementation to track the paws of running rodents. This algorithm can be applied to different animals as long as the tips of the legs can be differentiated from the background and other parts of the body using color features. The presented algorithm provides a robust tool for future studies in multiple fields, where precise quantification of locomotor behavior from a high-speed video is required. We further present a graphical user interface (GUI) to track, visualize, and edit the tracking data.

Keywords: Biomechanics, Animal tracking, Optimization, Spline matching, 3D reconstruction, Superpixels, Kalman filter python software

1 Introduction

Understanding biological movement is an important challenge for modern science. This understanding has a direct impact on human health and wellbeing. It promises to give new treatments for musculoskeletal injuries and neurological disorders, improve prosthetic limb design, and aid in the construction of more capable legged robots, all in addition to providing a wealth of necessary scientific information about biological systems [35].

One of the main features of locomotion is the gait (relative timing of leg recirculation, e.g., walk, run, trot, or gallop). How gait is chosen and the regulation of gait can provide detailed information about the condition of a subject [8]. Although significant insight into the neuromechanical basis of movement has been gained [43], there are many important open questions in this area; such as how does gait control reflect the morphology and dynamics of the fast moving body and how is sensory feedback used during rapid legged locomotion?

The intentional changes in an animal gait, the timing of paw motion relative to each other using the animal [11], can be seen during movement. The animal movement can be perturbed using an internal or external perturbation. A mechanical perturbation (e.g., earthquake) while the animal is running, for example, deflecting the surface during running, an electrical stimulation applied to the nervous system using nerve cuffs [33], or even the application of new genetically targeted techniques, like optogenetics [10] or designer receptors exclusively activated by designer drugs [50], are several of the increasingly sophisticated methods applying perturbations that dissect the movement control.

Mice are a premier model of human disease and increasingly the model system of choice for basic neuroscience. High frame rates (higher than 150 Hz) are needed to quantify the kinematics of running mice, due to their high stride frequency (up to 10 Hz). Achieving an adequate

*Correspondence: o.maghsoudi@temple.edu
[1]Omid Haji Maghsoudi Department of Radiology, University of Pennsylvania, Philadelphia PA, 19104, USA
Full list of author information is available at the end of the article

number of strides to capture inter-stride variability may require 3 s or more of a video; at least 450 frames need to be captured. This number increases rapidly with frame rate, which may be increased to capture sudden movements or reaction to impulsive perturbations, or with duration. This frame rate increase may yield large data sets for a more sophisticated analysis of locomotor dynamics [48]. More extensive data sets give researchers a better insight into their studies [61], but larger data sets cause difficulties in requiring space to store data and algorithms to track the desired animal body regions automatically.

Commercially available systems (Digigait [12, 15, 40], Motorater [46], Noldus Catwalk [11, 20, 23, 44]) are prohibitively expensive and may only provide information about paws during the stance phase which makes them limited for some studies. In both research and commercial systems, tracking of mice has frequently relied on shaving fur and then drawing markers on the skin for subsequent tracking from raw video [14], or on the attachment of retroreflective markers, and the use of optical motion capture systems [27].

Also, some computerized methods (simple thresholding, circular correlation, or template matching) have been proposed to answer this need [62]. However, manual [3] or semi-manual [22] clicking can be considered the usual method to track some markers or features. In addition, it is often important to track the tips of appendages, e.g., paws, feet, and hands, because they often carry extensive information relevant to a task [37]. However, they are often prone to difficult and periodic occlusions. Therefore, the need for a robust method to help neuroscientists and biologists have been felt.

To track animal kinematics, studies are limited to a few trials, because of low yield and a requirement for attaching or drawing markers. For example, Karakostas et al. [27] provide validation with two mice, one trial per mouse. While the scale-invariant feature transform (SIFT) has been successfully used for markerless tracking of the rat head [29], in comparison with a paw, the range of motion was limited because the animal was restrained during imaging. As an example of animal tracking, methods for fly tracking has been developed which they required a constant background and a limited range of motion [39]. In our setup, which is very common in neuroscientific applications, the use of a treadmill causes difficulties for the background subtraction process. For these reasons, we present a method for automated, markerless tracking of rodent paws.

Two-dimensional tracking from a video can provide the required information to examine gait. However, access to 3D information can improve our understanding of locomotion including roll, pitch, and yaw [38]. Indeed, a foot position relative to the body has been found to explain a large amount of the variation in kinematics from stride to stride [37]. In addition, using 3D can resolve occlusions and collisions of paws with other paws or the animal's body.

In general, tracking has been a recent favorite topic in image processing [63]. Many methods have been developed for different applications: cell migration tracking [45], human tracking [30], and tracking diseased tissue across frames [16], [36]. Tracking methods typically need to be developed based on the specific problem at hand. Kalman filters (KF) [54], global nearest neighbor standard filters [58], joint probabilistic data association methods [49], multiple hypotheses tracking [28], Markov chain Monte Carlo data association [41], and probabilistic multiple-hypothesis tracking [57] techniques are some of the most common methods presently used for multiple object tracking.

We take advantage of superpixels for segmentation of paws [35]. We have shown the possibility to segment different parts of the body using superpixel methods in [35]. Here, we extend these methods to make a more robust, automated tracker for paws (required one-time clicking of user), by utilizing 3D information across views, and temporal prediction based on generalized paw kinematics. The main contribution of this study is thus a markerless paw tracker that is robust to occlusions and collisions over multiple frames, resolving many of the limitations of our previous work [17], [19]. We achieved this using 3D information, integrating information from four cameras at the same time. A 3D KF and the direct linear transform were used to achieve this goal. We employed an optimization method to fit a spline reference of paw kinematics to the observed data during tracking, to predict the position of paws and resolve occlusions and collisions. Finally, we provide an accessible software to use the tracker, to speed up kinematics analyses by researchers in, e.g., neuroscience and biomechanics.

This manuscript is organized into five sections: the methods, software, experimental conditions, results, and conclusion. Section 2 presents our proposed tracking algorithm in detail. Section 3 provides the required information about the Python packages and the main parameters being involved for implementing the algorithm as a code. Section 4.1 gives more details about our experimental values for designing the software. Section 4 presents information about our methods performance, and Section 6 summarizes our findings and possible future directions.

2 Methods

We first describe the imaging setup and techniques to control animal speed, then detail the segmentation process, before finally discussing the 3D tracker.

2.1 Setup

2.1.1 Camera system

Ximea USB3 (camera model: MQ022CG-CM) cameras are used to capture frames using a 250 Hz external synchronization signal. The trigger signal is generated and synchronized with a host PC using the *triggerbox* tools published by the Straw Laboratory [56]. Briefly, the trigger pulses are generated by an Arduino Uno, running the *triggerbox* firmware (the code can be found on Triggerbox (https://github.com/strawlab/triggerbox). The Arduino is controlled via serial over USB by a standard desktop PC. The camera resolution is set to 2048 × 700 pixels at 8-bit depth, using a Bayer filter pattern to recover color. The capture time is 4 s, gathering 1000 frames for each trial. The frames are Bayer encoded, and we use a debayering function to convert them to RGB color space frames [19]. Last but not least, the treadmill arena lightening condition was remained constant using LED panels.

2.1.2 Treadmill and tracking system

We use a closed-loop treadmill system described in [55] to control and adjust the speed of a treadmill while a mouse is running. The feedback loop helps us to keep the animal in a specific place on the treadmill by varying the belt speed (for example in the middle) or to automatically capture data when the animal movement meets specified criteria (e.g., constant speed for *n* strides). Four cameras (four side views; Fig. 1) are used to capture the locomotion of the animal on the belt. An additional camera located at the top of the belt (shown in red) tracks the animal to provide a real-time feed of position and speed for the visual servoing of the treadmill belt [55]. Here, we analyze the frames captured by the cameras on side views (Fig. 1).

The mouse paw color is pink, which is a relatively unique color in the video sequences, restricted to tail, ear, nose, and any shaved parts of the body (if a study needs shaving). We further enhance segmentation by painting the treadmill belt with Chroma-key Green paint (Sherwin Williams, interior acrylic latex, green-yellow custom blended to match ROSCO GaffTac fluorescent green keying tape). This helps to simplify the segmentation of paws as described in [35].

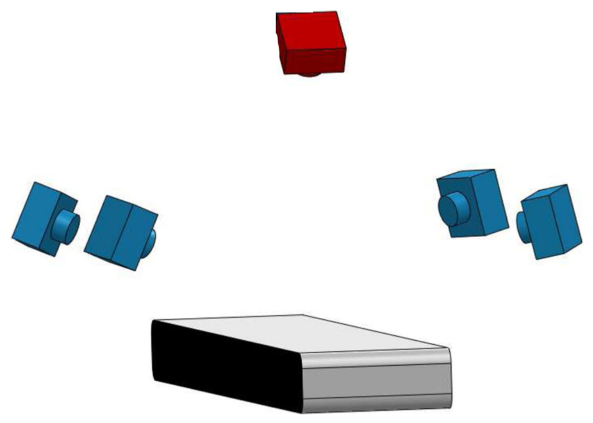


Fig. 1 Camera and treadmill arena. Four side view cameras are used for capturing high-speed kinematics, and one camera above (red) provided tracking of the animal and adjusted the speed of the belt using the closed-loop feedback method of Spence et al. [55]

2.2 Segmentation

Normalized cuts [51], the mean shift algorithm [9], graph-based methods [13], simple linear iterative clustering (SLIC) superpixels [1], and optimization-based superpixels [59] are all modern methods with which to segment regions of an image. Superpixels contract and group uniform pixels which make a more natural and perceptually meaningful representation of the input image, as compared to single pixels. We choose to use superpixels because their over- and under-segmentation criteria and performance are well suited to the task of segmenting the paws. We use simple linear iterative clustering [1] because it generates superpixels faster than other methods. As the size of our images is constant, the number of superpixels is the critical parameter. The speed of the superpixels algorithm depends largely on this number of superpixels and the size of the image as it has been discussed in [1]. Figure 2 shows how the number of SLIC segments can affect the segmentation process around the body and paws.

2.3 3D tracking system

For clarity, we describe the tracking system in four parts: first, manual initialization of paws positions for the first frames of each camera (and if needed, for the second frames); second, the 3D reconstruction and how it is employed to aid in tracking; third, a typical paw locomotion pattern; and finally, the automatic tracking system, referred as the "general tracker."

The following terms should be clarified to simplify the descriptions: "hidden paw," "collision type I," "collision type II," and "collision type III." The "hidden paw" is a paw becoming obscured by the body. The collisions of a paw with paws on the other side of the body is referred to as a "collision type I." The collision of the front and hind limbs on the same side is referred to as a "collision type II." We detect the "collision type II" when the paws on a same side are closer than "collision type II threshold," as defined in Section 3.2. The "hidden paw" typically happens for the front paws at the early stage of the swing phase of a stride. This is similar for the "collision type II," with a small difference: the "collision type II" occurs before having a "hidden paw." It should be noted that "collision type II" might happen more frequently as compared with the "hidden paw." On the other hand, "collision type I" happened while one paw is in the stance phase and the other one, on the other

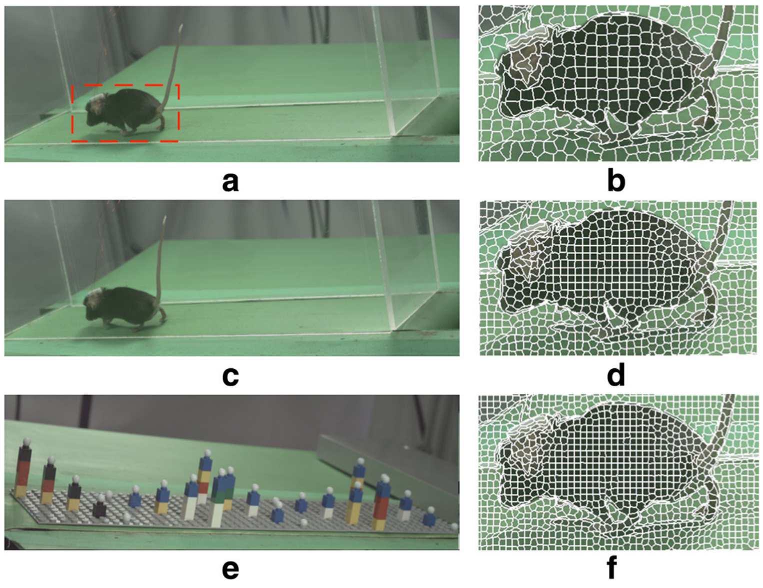


Fig. 2 a A frame captured from a C57BL/6 mouse where the paws have a "collision type II." The SLIC results with 5000, 10,000, and 15,000 superpixels for the marked red region in **a** are illustrated in **b**, **d**, and **f**, respectively. The marked red region is selected to demonstrate a better resolution of SLIC segmentation around the mouse body; this box does not show the "sub-image" to reduce the time required for generating superpixels. Panel **c** shows a "collision type I" for the front paw. Note that a similar collision can occur for the hind paw. The calibration object with 25 markers is seen in **e**

side, is in the swing phase. Finally, the "collision type III" is like the "collision type I," but it happens because of resolving the "collision type II." The detected paw, while resolving the "collision type II," might jump to the other side paws which can cause a similar condition as "collision type I." This can happen because the other side paw has the same color information and being relatively close to the desired paw. "Collision type I" and "type II" are shown in Fig. 2.

2.3.1 Initialization

The major role of this simple step is to find the paws coordinates in the four 2D camera frames and extract some features to aid in finding the best-matched segment from among the paw segments for the subsequent frames. Therefore, the superpixels are generated using the initial value for the number of superpixels, and the user is asked to zoom in, using a rectangle zoom tool in the software, for a better resolution and click on the paws (front and hind paws, respectively) for each camera. This means that the initialization consists of one round of clicking on the paws by the user and further processing as described in Algorithm 1 and Algorithm 2. From this step, we extract the following features: initial red, hue, and green values ($R_{D_{[m,0,c]}}$, $H_{D_{[m,0,c]}}$, and $G_{D_{[m,0,c]}}$ in Eq. 3). The red and green values are extracted from RGB color space while hue is extracted from HSV color space. These three values would remain constant unless the user requests to modify the tracking parameters. In the case that a user asked for changes, these values could be updated by new values for

Algorithm 1 The presented algorithm includes an initialization step for the first frame which needs manual clicking by user. The variables m, n, c, and i are paw index, frame number, camera number, and superpixel number, respectively. The parameter j keeps two same side cameras (cameras 1 and 2 or cameras 3 and 4) index to use 3D reconstruction information. H, G, and R are the related hue, green, and red channels, respectively. D, U, and V are the detected paw, coordinate in the horizontal direction of an image plane, and coordinate in the vertical direction of the image plane, respectively. The parameters X, Y, and Z are coordinates in 3D. SP shows superpixels. DLT and DLTI are direct linear transform and direct linear transform inverse which can be calculated based in equation 1. The initial tracker generates superpixels for the entire image and the user select the paws in each camera image plane. It should be noted that we have four KFs for the paws.

```
1: procedure INITIAL TRACKING(First Frame (n=1))
2:     for each camera (c) do Generate $SPs$
3:         for each paw (m) do
4:             Manual Clicking
5:             Extract Features
6:             Set $D_{[m,n,c]}, H_{D_{[m,n,c]}}, G_{D_{[m,n,c]}}, R_{D_{[m,n,c]}}$
7:     for j=[1,2] or [3,4] do
8:         for each paw (m) do
9:             $(X,Y,Z) \leftarrow \mathrm{DLT}(D_{[m,n,j(1)]}, D_{[m,n,j(2)]})$
10:            Initialize KF$(X,Y,Z,m,j)$
```

that frame. These values are used to extract features for the next frame, see Eq. 3.

Then, the paw coordinates in each direction (U and V) are summed with a number ("PawXWindowSize" in the U direction and "PawYWindowSize" in the V direction, defined in Section 3.2) to make sure that the paw is not missed in the next frame. "PawXWindowSize" and "PawYWindowSize" are typically 70 and 40 pixels, respectively. The values of 70 pixels in the U direction and 40 pixels in the V direction ensure that, at the chosen frame rate of 250 Hz, the paws do not move beyond the window of detection in one frame. We analyze a range of animal speeds from 10 cm/s to 70 cm/s, and the maximum displacement of the paw for two consecutive frames is 30 pixels in the U direction and 15 pixels in the V direction. This value can be adjusted by the user, based on the setup; however, having higher values would reduce the speed.

The initialization process can be repeated for the second frame of each camera or be bypassed. The second frame initialization is recommended if "collision type I," "collision type II," or "hidden paw" are occurring at the start. However, the tracker does not require the second frame initialization, and frequently tracks the paw successfully without this. Any incorrect tracking can be corrected using the software described in Section 3.

2.3.2 3D reconstruction and 3D Kalman filter

We previously developed 2D-based tracking systems using a support vector machine [19]. The main limitations for those systems (relying on machine learning) are a loss of tracking due to "hidden paw," "collision type I," and "collision type II" issues, that are difficult to overcome using only 2D information. Also, the 2D nature of those tracking systems means that they are sensitive to the animal moving diagonally across the belt. This diagonal movement results in large numbers of hidden paws.

We can take advantage of 3D reconstruction to solve the "hidden paw" and "collision type I" issues, as well as the change of direction, if we track the paws in at least two cameras. We calibrate the treadmill volume with a custom Lego calibration object, with markers located at known coordinates. This calibration object is shown in Fig. 2.

Then, direct linear transform (DLT) is used to map the 2D coordinates to 3D in the calibrated volume [21], [47]. DLT has been a popular method for 3D modeling of objects, in both biology and biomechanics fields [6, 7, 22, 25, 26]. The projection from 3D domain to the 2D camera image planes (focal planes) is shown in Fig. 3. DLT method formalizes the following relation between an object located in 3D with a corresponded object on an image plane of a camera, e.g., camera 1:

Algorithm 2 The second frame tracking algorithm. This includes a possible initial tracking for the second frame; however, this can be done by the general tracker as explained in Algorithm 3. In other words, this is an optional step which can be skipped; however, it can help for more accurate results. The variables are defined in Algorithm 1.

```
1: procedure INITIAL TRACKING(Second Frame (n=2))
2:     for each camera (c) do Generate $SPs$
3:         for each paw (m) do
4:             Manual Clicking
5:             Extract Features
6:             Set $D_{[m,n,c]}, H_{D_{[m,n,c]}}, G_{D_{[m,n,c]}}, R_{D_{[m,n,c]}}$
7:     for j=[1,2] or [3,4] do
8:         for each paw (m) do
9:             $(X, Y, Z) \leftarrow \mathrm{DLT}(D_{[m,n,j(1)]}, D_{[m,n,j(2)]})$
10:            Update KF$(X, Y, Z, m, j)$
```

$$U1 = \frac{L_1X + L_2Y + L_3Z + L_4}{L_9X + L_{10}Y + L_{11}Z + 1},$$
$$V1 = \frac{L_5X + L_6Y + L_7Z + L_8}{L_9X + L_{10}Y + L_{11}Z + 1}. \quad (1)$$

Where $U1$ and $V1$ are the coordinates in the focal image plane of camera 1. $U2$ and $V2$ for other cameras can be calculated based on calculated coefficients corresponding to that camera. X, Y, and Z are the coordinates in the 3D domain, and L_1 to L_{11} are eleven DLT coefficients being calculated by the calibration object [22]. The camera calibration is done using DLTdv5 Matlab package and manually clicking on the 25 balls from each camera, shown for a camera in Fig. 3, to calculate the DLT coefficients. Using Eq. 1, we can map the 2D coordinates from two cameras to 3D or come back from the 3D coordinates to the 2D camera image planes.

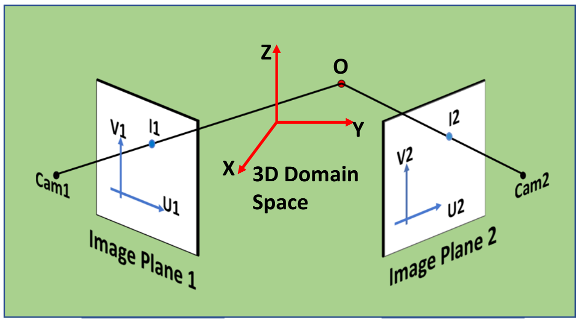


Fig. 3 This figure shows the concept for 3D reconstruction using two cameras image planes (focal planes of cameras). *O* is a point located in a 3D object space having the *X*, *Y*, and *Z* coordinate system. *I1* and *I2* are the projected points in the image plane 1 of camera 1 and the image plane 2 of camera 2, respectively

For the reasons explained, we use the 3D reconstructed coordinates to predict the position of a paw for the next frame based on the tracked markers for the current frame. We employ a 3D KF that can take observed measurements over time and estimate variables related to the motion [24] to achieve this goal. The KF model considers a state for a frame n evolving from the prior state at frame $n-1$ [5]. We had previously used a 2D KF, while here, we apply the KF to X, Y, and Z directions (in 3D; the coordinate system illustrated in Fig. 3). The 3D KF includes the following parameters: (1) the dynamic parameters or dimensionality of the states are three states because we need three dimensions (X, Y, and Z), (2) the measure parameters or dimensionality of the measurements are six measurements because we assume the paw motion has a constant speed (no acceleration) pattern and to avoid complexity, and (3) default covariance matrices by OpenCV package. Therefore, the measurement matrix is defined as follows:

$$\begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}$$

and this is the transition matrix:

$$\begin{bmatrix} 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}.$$

The KF implementation details as a code can be found in Section 3.1. Therefore, we can use the estimation the 3D location of paws for the next frame using the coordinates from the initialization step or the previously tracked paws.

2.3.3 Computation of a spline template of paw kinematics for collision resolution

By looking at tracked paw coordinates in a 2D camera image for just one stride, it can be similar to donuts shape. Animals have to take the paw off from the ground, called swing, and then step down, called stance. This pattern might have a longer or shorter stance time, referred to as "frequency of paw locomotion;" the paw goes higher in the air, referred to as "amplitude of paw locomotion;" and time difference that a paw does a specific part of the motion pattern, referred to as time shift. The swing time depends on the morphology and mechanical properties of paws. A typical walk and trot have been shown in Fig. 4. The "amplitude of paw locomotion" is more related to the size of the animal and it remains roughly constant with the same animal size. On the other hand, the "frequency of paw locomotion" can vary from one stride to another stride. This pattern is shown in Fig. 5. However, the paw locomotion is not necessarily periodic; we use the term "frequency of paw locomotion" to show the changes in the required time for one full circulation. It should be mentioned that the front and hind paws have different splines, but the other side limbs (either the front or hind paws) have the same spline functions.

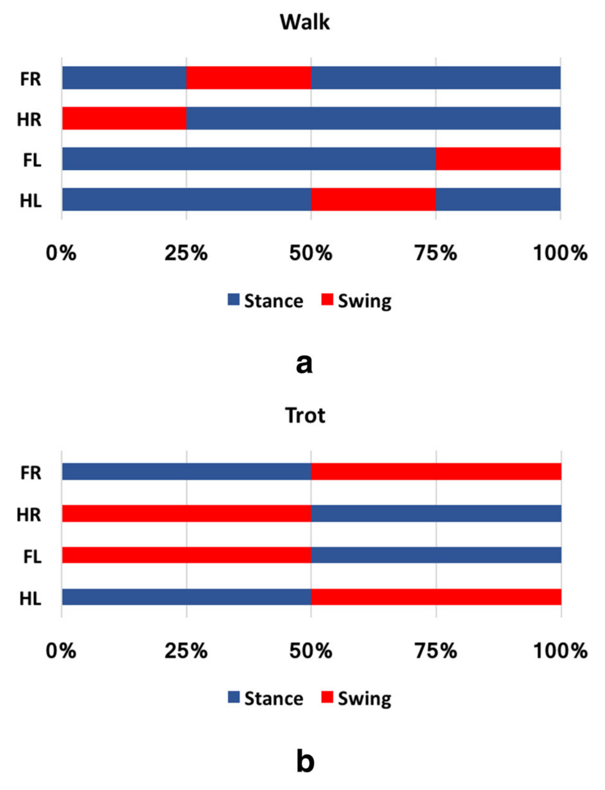


Fig. 4 Gait Diagram. Panel **a** shows the typical walk gait diagram for quadrupeds and panel **b** shows the ideal trot gait for quadrupeds, adopted from [53]. In each of the graphs, blue and red bars are the stride cycle part that leg is in the stance phase (foot on the ground) and the part that leg is in the swing phase, respectively. FR, HR, FL, and HL are front right, hind right, front left, and hind left, respectively

We tracked ten strides from an animal with constant a speed, 30 cm/s, and calculated the average. Cubic splines were fit using SciPy to the average for each paw.

$$G(x) = d_q x^3 + e_q x^2 + g_q x + h_q \quad 0 < x < K \tag{2}$$

where K is the number of points for the locomotion pattern (K was 50 for rats in our case). This number is selected based on the average frames numbers for one stride among those ten strides constant the speed of 30 cm/s. d_q, e_q, g_q, and h_q are coefficients for each knots (q) while the term x denotes the continuous numbers in the range covering the spline function. This spline is used in the "general tracker" step to help to resolve the "collision type II." It should be mentioned that splines are calculated before running the software.

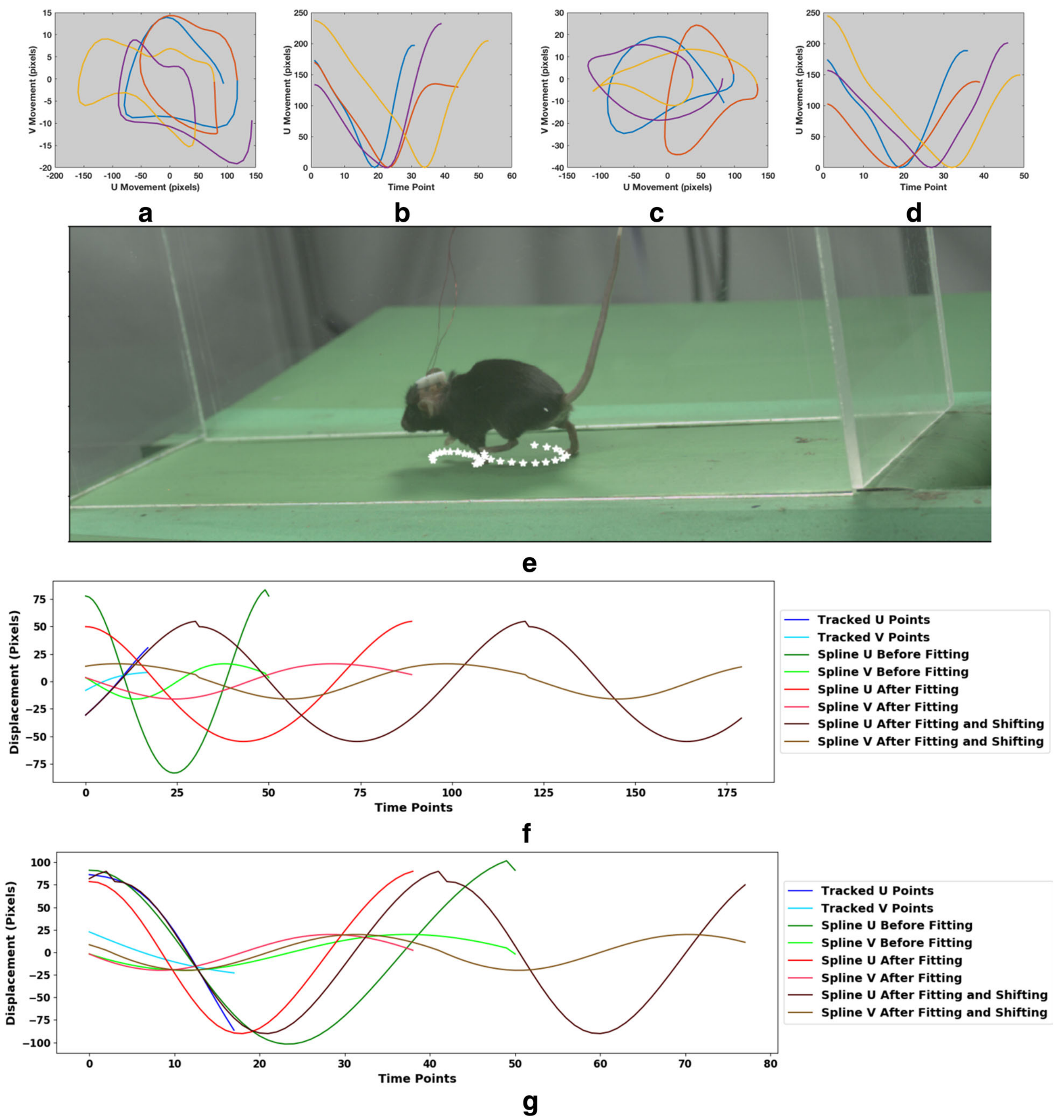


Fig. 5 Utilization of a spline function for tracking through collisions. The front paw motion pattern in the $U - V$ plane, the front paw motion pattern in the U direction, the hind paw motion pattern in the $U - V$ plane, and the hind paw motion pattern in the U direction for four consecutive strides are illustrated in **a**, **b**, **c**, and **d**, respectively. In panels a to d, the colors show different strides. Panels b and d show the coordinates shifted up to have the same minimum number (zero). The plots in the $U - V$ plane are shifted to have approximately the same start point. The predicted points for future frames when "collision type II" occurs are shown in **e**. Front and hind paw predicted locations based on previously tracked points are illustrated in **f** and **g**, respectively. Each of these two plots contains the spline function (which is generated from a previously studied mouse with a constant speed of 30 cm/s and averaged from ten strides) before fitting, after fitting of the spline templates to the tracked paw coordinates by applying "frequency of paw locomotion" and not time shift, and final spline function helping the prediction of paws location when "collision type II" happens. It should be noted that fitting is applied for the U coordinates, and the V coordinates are moved accordingly. Finally, two cycles of splines are considered to make sure that the trajectory of coordinates will be continuous

Algorithm 3 The presented algorithm includes general tracker. The variables are defined in Algorithm 1. In addition, L is a symbol for the spline, defined in equation 7. Flag_Collision is a flag to keep the collision event condition (if it happens or not). The variables, $\hat{\tau}, \hat{f}$, and $\hat{a}$ are the time shift, "frequency of paw locomotion," and "amplitude of paw locomotion," respectively. Last but not least, O shows that we have a pair of image planes, but we process the cameras in 2D separately using DLTI function. DLT and DLTI use the coefficients calculated in equation 1. In other words, O has the value which is the j vector but not assigned as the value of c. The general tracker uses the pair of cameras located on a same side (j) and DLTI to map the predicted 3D coordinates of KF to provide the window in 2D image planes. However, if the collision type II happened, it uses the Spline_Predictor vector for this purpose.

```
1: procedure GENERAL TRACKING(Frame number n)
2:   for j= same side camera number ([1,2] or [3,4]) do
3:     Flag_Collision(j(1)) = 0
4:     Flag_Collision(j(2)) = 0
5:     for each paw (m) do
6:       for each camera (c) in j do
7:         if Flag_Collision(c) is equal by 0 then
8:           $(X, Y, Z) \leftarrow$ Predict KF$(X, Y, Z, m, j)$
9:           $[(U_{m,n,c}, V_{m,n,c}), (U_{m,n,O}, V_{m,n,O})] \leftarrow$ DLTI$(X, Y, Z, j(1), j(2))$
10:        else
11:          $[(U_{m,n,c}, V_{m,n,c}), (U_{m,n,O}, V_{m,n,O})] \leftarrow$ Spline_Predictor$(c)$
12:      for each camera (c) in j do
13:        Create sub-images based on $(U_{m,n,c}, V_{m,n,c})$
14:        Generate SPs
15:        for each SP (i) do
16:          Extract Features (described in equation 3)
17:          Apply Fusion Function (equation 6)
18:        Find SP with minimum Score
19:        Keep Best Three SPs
20:        Set $D_{[m,n,c]}, H_{D_{[m,n,c]}}, G_{D_{[m,n,c]}}, R_{D_{[m,n,c]}}$
21:        if Collision Type II happens and Flag_Collision(c) is equal by 1 then
22:          Fit Spline to Detected Paw (m)
23:          Flag_Collision(c) = 1
24:          Estimate ( $\hat{\tau}, \hat{f}, \hat{a}$) parameters using the fitting process defined in equation 8
25:          Set Spline_Predictor(c) = L(t-$\tau$)
26:          Set Weights = Adjust_Weight_Factors(Weights) as described in Algorithm 4
27:        else
28:          Flag_Collision(c) = 0
29:          Set the weight factors to the original values
30:        if Collision Type III happens then
31:          if Collision Type III happens for the first time in this stride then
32:            Based on the expected paw motion direction (front paw should go forward and hind paw should go backward), Find the first SP among Best Three SPs that follows the motion direction of paw
33:          else
34:            Remove the SP with minimum Score from the Best Three SPs (Two SPs remaining)
35:            Find the first SP (priority goes with the better SP in the remaining two) that has an U coordinate (horizontal direction) less than previous frame ($D_{[m,n-1,c]}$) and higher than the SP with minimum Score ($D_{[m,n,c]}$)
36:          Set $D_{[m,n,c]}, H_{D_{[m,n,c]}}, G_{D_{[m,n,c]}}, R_{D_{[m,n,c]}}$ with the results
37:    $(X, Y, Z) \leftarrow$ DLT$(D_{[m,n,j(1)]}, D_{[m,n,j(2)]})$
38:    Update_KF$(X, Y, Z, m, j)$
```

2.3.4 General tracker

In a first step, we use a median filter to reduce noise in the image (low-pass filter works the same). We subsequently focus on a 140 × 80 pixel region of interest, given by the 2D projection of the 3D coordinate predicted by a 3D KF filter (see Section 2.3.2 for more details). This point has 2D coordinates of $[U_{P_{[m,n,c]}}, V_{P_{[m,n,c]}}]$ in image plane of camera number c for the paw number m and frame number n. It should be noted that paw number is one (front paw) and two (hind paw). We refer to this zoomed region as the "sub-image." The "sub-image" center is determined by the 3D KF filter prediction. Then, we generate the superpixels for the "sub-images" and extract eight color and location features as follows:

$$\begin{aligned}
F_{1[i,m,n,c]} &= |\text{Mean}\left(G_{SP_{[i,n,c]}}\right) - \text{Mean}\left(G_{D_{[m,0,c]}}\right)| \\
F_{2[i,m,n,c]} &= |\text{Mean}\left(G_{SP_{[i,n,c]}}\right) - \text{Mean}\left(G_{D_{[m,n-1,c]}}\right)| \\
F_{3[i,m,n,c]} &= |\text{Mean}\left(H_{SP_{[i,n,c]}}\right) - \text{Mean}\left(H_{D_{[m,0,c]}}\right)| \\
F_{4[i,m,n,c]} &= |\text{Mean}\left(H_{SP_{[i,n,c]}}\right) - \text{Mean}\left(H_{D_{[m,n-1,c]}}\right)| \\
F_{5[i,m,n,c]} &= |\text{Mean}\left(R_{SP_{[i,n,c]}}\right) - \text{Mean}\left(R_{D_{[m,0,c]}}\right)| \\
F_{6[i,m,n,c]} &= |\text{Mean}\left(R_{SP_{[i,n,c]}}\right) - \text{Mean}\left(R_{D_{[m,n-1,c]}}\right)| \\
F_{7[i,m,n,c]} &= \text{Sqrt}\left(\left[\text{Mean}(U_{SP_{[i,n,c]}}) - U_{B_{[m,n,c]}}\right]^2 + \left[\text{Mean}(V_{SP_{[i,n,c]}}) - V_{B_{[m,n,c]}}\right]^2\right) \\
F_{8[i,m,n,c]} &= \text{Sqrt}\left(\left[\text{Mean}(U_{SP_{[i,n,c]}}) - U_{P_{[m,n,c]}}\right]^2 + \left[\text{Mean}(V_{SP_{[i,n,c]}}) - V_{P_{[m,n,c]}}\right]^2\right)
\end{aligned} \tag{3}$$

where i, m, c, and n are the superpixel number (for all superpixels in a "sub-image"), the paw number (four cameras and two paws for each camera), the camera number, and the frame number. $SP_{[i,n,c]}$, $D_{[m,n,c]}$, and $P_{[m,n,c]}$ are the superpixel number i for the frame n in camera c, the detected paw number m for frame number n in camera c, and the predicted position of paw number m for frame number n in camera c. $F_{1[i,m,n]}$ to $F_{8[i,m,n,c]}$ are the eight features corresponding to $SP_{[i,n,c]}$. In addition, R, H, and G are intensity values in the red channel from the RGB color space [18], values in the hue channel from the HSV color space [31], intensity values in the green channel from the RGB color space, and the center of a "sub-image," respectively. B and P are the coordinates of the bottom left and the center of the "sub-image" in the camera image planes. Therefore, the term Mean$(G_{SP_{[i,n,c]}})$ means the average of intensity values of pixels in i-th superpixel of n-th frame from the c-th camera.

It should be noted that the SP denotes superpixels for the current frame (in other words, superpixels candidates for being a paw in frame number n), and the term i is an index to test all the superpixels in the window. On the other hand, D is the detected superpixel from the previous frame (frame number $n-1$). After the segmentation, tracking, and the resolution of the collisions (if needed), the best superpixel is assigned to $D_{[m,n,c]}$ for frame number n, as illustrated in Algorithm 3. In addition, since a superpixel has many pixel members and each pixel has coordinates in the image in the U and V directions, U_{SP}, V_{SP}, U_P and V_P are arrays of scalars. Therefore, Mean(U_{SP}) means that we calculate the average of coordinates in a direction for all for the pixels associated with that superpixel.

Eight features are normalized ($N_{k[i,m,n,c]}$ where k shows the feature number between one to eight) as follows:

$$N_{k[i,m,n,c]} = 1 - \frac{F_{k[i,m,n,c]} - \min(F_{k[\forall i,m,n,c]})}{\max(F_{k[\forall i,m,n,c]}) - \min(F_{k[\forall i,m,n,c]})} \quad (4)$$

The normalized features are weighted based on the importance of features using the following arrays:

$$\begin{aligned} &\text{Weights\_F} = [2, 0, 4, 2, 2, 0, 1, 4] \\ &\text{Weights\_H} = [2, 0, 4, 1, 2, 0, 2, 4] \end{aligned} \quad (5)$$

where Weights_F and Weights_H are the weights to calculate a score for the front and hind paw segments, respectively. The reason for the selection of these weights is discussed in Section 4.1.2. By having the weights and normalized features, we can have a fusion function as follows:

$$\text{PF}(i, m, n, c) = \frac{\sum\limits_{k=1:8} N_{k[i,m,n,c]} W_k}{\sum W} \quad (6)$$

where W is the one of the weight vectors (Weights_F or Weights_H). PF is calculated for each superpixel in a "sub-image", and a superpixel with minimum value is selected as the paw for that image plane, referred to as the "best superpixel." However, three superpixels with minimum values, referred to as the "three best superpixels," are stored for further possible required processes (resolving the collisions). Therefore, we have the most likely superpixel showing the paw and storing the other two possible candidates for the paw. The number for other candidates can be any number; however, we suggest keeping the number close to two or three because the higher number can increase the chance of mislabeling of paw.

Through experimentation, we found that the "best superpixel" is accurate enough to track the paw if a "collision type II" does not happen. But, this issue happens in almost every stride, and an error in the tracking of the paws for a single frame could lead to missed tracking for all subsequent frames. To solve this issue, we develop a method previously calibrated spline from a paw motion template pattern to the tracked paw coordinates, as described in section donuts.

We fit the spline function to the previously tracked paw coordinates to calculate the "frequency of paw locomotion" and the time shift. The same coefficients calculated in Eq. 2 (d, e, g, and h) is used to find the proper knots for integer points based on the time shift and "frequency of paw locomotion" as follows:

$$\mathrm{L}_f(x) = d_q(xf)^3 + e_q(xf)^2 + g_q(xf) + h_q \; (\forall xf \text{ in } [0, K]) \quad (7)$$

where L is the spline points of a paw for fitting process. The term f shows the "frequency of paw locomotion." The term x is all the possible integer numbers in $[0, K]$.

A cost function is defined to minimize by finding the maximum value of circular correlation between these two signals by varying the "frequency of paw locomotion" and the time shift as shown in Fig. 5. It should be noted that a low-pass filter was applied to smooth the tracked coordinates. The frequency of cutoff was set in a way to avoid the frequency of movement. It should be noted that the cut-off frequency of filter is selected based on the maximum frequency of paw movement (10 Hz) versus the capturing rate (250 Hz). The fitting for amplitude and frequency is performed as follows:

$$(\hat{a}, \hat{f}) = \underset{a,f}{\arg\min}((T_{[t]} - \overline{T_{[t]}}) - a(L_f(t-\hat{\tau}) - \overline{L_f(t-\hat{\tau})})) \quad (8)$$

where $\hat{\tau}$ is being calculated based on the maximum value of circular correlation formalized by the product of the tracked points and spline knots as follow:

$$\hat{\tau} = \underset{\tau,f}{\arg\max}((T_{[t]} - \overline{T_{[t]}}) * a(L_f(t-\tau) - \overline{L_f(t-\tau)})) \quad (9)$$

where T are the previously tracked U coordinates of a paw. T is an array which does circular rotation on array elements based on the value of t as time shift. The term $\hat{\tau}$ is the time shift to maximize the the circular correlation product. The terms $\hat{f}$ and $\hat{a}$ are the frequency and amplitude of paw locomotion being minimized by the defined cost function. The term $\overline{\psi(t)} = \frac{1}{2}(\max(\psi(t)) + \min(\psi(t)))$ denotes the midpoint between the upper and lower evaluated points of the sequence. To make sure that T and L have the same number of elements, we added zeros if T has a fewer number of elements. Also, we make sure that T does not have more members than L. It should be noted that we fitted the spline function for the U coordinates, not V. The reason is that the V coordinates are noisier, causing poor fits. Therefore, we use the frequency and time shift from the U movement to adjust the related spline function in the V direction for the prediction, as shown in Fig. 5.

Algorithm 4 This is adjustment function. This function can affect up to %10 in the major revision needed. Therefore, it can be an optional step.

```
1: function ADJUST_WEIGHT_FACTORS(Weights)
2:     if Weights is equal by Weights_F then
3:         Weights[1] = Weights[1] + 1
4:         Weights[4] = Weights[4] - 1
5:         Weights[7] = 0
6:     elseWeights is equal by Weights
7:         Weights[3] = Weights[3] + 2
8:     Return(Weights)
```

The calculated "frequency of paw locomotion," "amplitude of paw locomotion," and the time shift for achieving a maximum value for matching of the two signals are used to make a new spline function, referred as the "predicted spline function." This "predicted spline function" helps to predict the location of paws on a same side when "collision type II" happened. It means that the paws are closer than a threshold to each other, called "collision type II threshold." This function is illustrated in Algorithm 3.

However, while the "predicted spline function" fixed the issue of "collision type II," it could cause a new problem: a possible jump from the same side front paw (happening for the front paw, not the hind paw) to the other side front paw. As described at the beginning of Section 2.3, this makes "collision type III." This issue is solved using the fact that the front paw after having the "collision type II" should go forward while the front paw on the other side should move toward the back. By tracking the paws for all four cameras, we can use the information from the other side to find the direction of paw movement for both sides based on the movement direction from the previous frames. In addition, 3D reconstruction aided in finding "collision type III," because the jump from the correct front paw to other side paw causes a significant 3D reconstruction error. This error is used to make sure the front paw is tracked correctly. The "collision type III" could be detected after having the paws tracked based on the "best superpixel." To correct this mistake in tracking, we find the best superpixel having the same motion direction and being closest in 3D among the "three best superpixels." The algorithm is shown in Algorithm 3.

Last but not least, the algorithm illustrated in Algorithm 4 can be considered as an option step. We realized that this step can reduce the number of mistakes in detection as explained in Section 4.5. The reasons for selecting the changes have been discussed in that section.

3 Software

The advantages of using Python, which is not limited to clear syntax, useful built-in objects, ease of extension, and so many packages developed for scientific applications, compared to the other programming languages are discussed in [42]. In addition, Python for image processing has a very rich set of image analysis tools including OpenCV [4] and the scikit-image libraries [60]. Therefore, we use it for developing the software.

In this section, the required packages for using the software are discussed while the software can be downloaded on GitHub Repository (https://github.com/omaghsoudi/3D-Paw-Tracking-Edition-Python.git). The graphical user interface of software has been shown in Fig. 6.

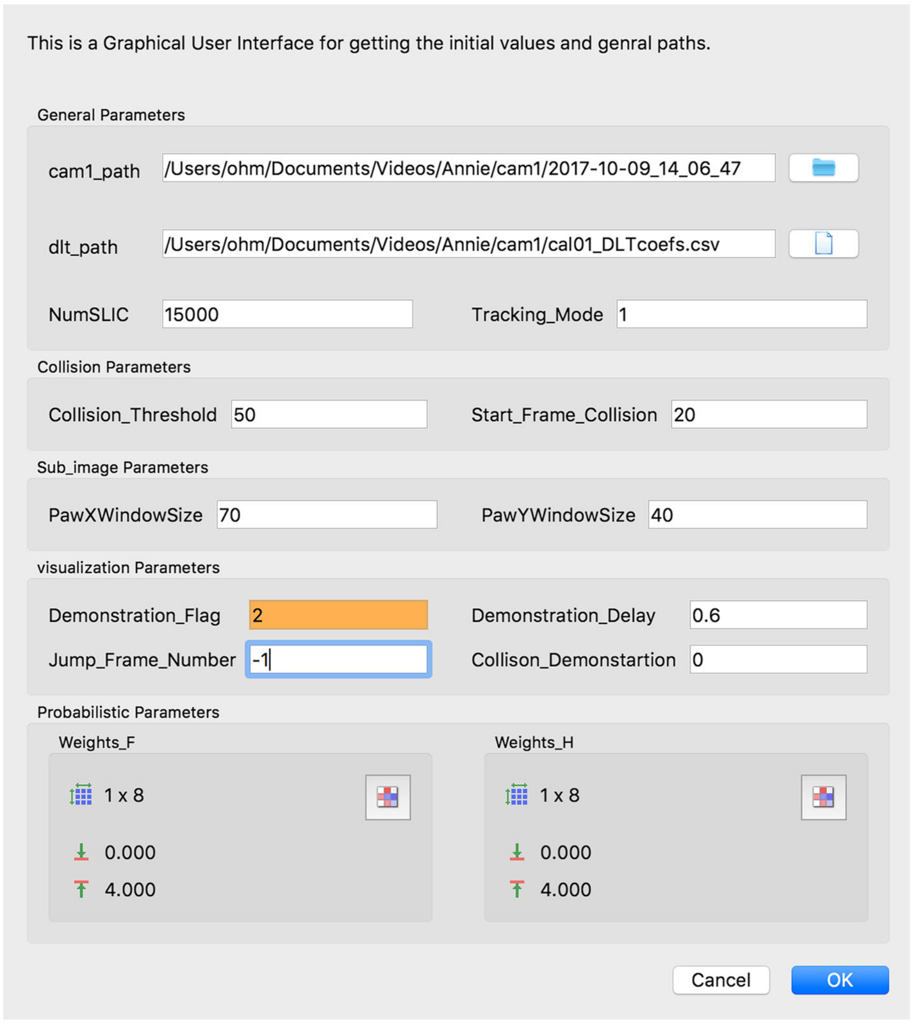
3.1 Required packages and software engineering

Python 3.6.5 is used to develop the open-source software. The following packages are needed to be installed for using the software: OpenCV 3.4.1, numpy 1.14.5, matplotlib 2.2.2, pandas 0.23.1, scipy 1.1.0, guidata 1.6.1, termcolor 1.1.0, and scikit-image 0.14.0.

From SciPy package, "minimize" function is used for the fitting process. In addition, a low-pass filter to smooth the previously tracked coordinates (before doing the fitting process) is performed using "signal" toolbox in SciPy. Many functions from OpenCV are used, including, color transform, KF, and drawing circle on an image to demonstrate the tracked coordinates. SLIC method is performed using the scikit-image package, and pandas is used to save the coordinates in a "csv" format file or to load the saved file.

The code is developed in an object-oriented programming method [52] to easily track the functions and their outputs. This can help programmers to develop and extend our work. In addition, we consider the main code, "GUI_Tracker.py," calling four modules (codes): (1) "CSV_RW.py" to read and write the results; (2) "Global_Var.py" to pass all required global variables between the modules and the main code; (3) "Keyboard_Fun.py" to interact with a user through keyboard, in addition, the SLIC is performed using this module; and 4) "Kalman_DLT.py" to apply 3D reconstruction functions or 3D KF.

In addition, we have considered a system to edit the tracking mistakes. If any mistake would be seen in the tracked coordinates, the user can request to edit them using some keyboard shortcuts. However, it should be noted that the system would not stop tracking unless the user asks for a demonstration or edition. The proper implementation and integration of resolving the mistakes with the tracker is difficult due to a need for resetting many parameters (e.g., features, coordinates, Kalman predictor system, 3D coordinates) based on a request for visualizing (with ability to go back and fore in the frames), editing the tracked coordinates (with ability to go back and fore in the frames), tracking of the paws, or saving



This is a Graphical User Interface for getting the initial values and genral paths.

General Parameters

cam1_path /Users/ohm/Documents/Videos/Annie/cam1/2017-10-09_14_06_47

dlt_path /Users/ohm/Documents/Videos/Annie/cam1/cal01_DLTcoefs.csv

NumSLIC 15000 Tracking_Mode 1

Collision Parameters

Collision_Threshold 50 Start_Frame_Collision 20

Sub_image Parameters

PawXWindowSize 70 PawYWindowSize 40

visualization Parameters

Demonstration_Flag 2 Demonstration_Delay 0.6

Jump_Frame_Number -1 Collison_Demonstartion 0

Probabilistic Parameters

Weights_F: 1 x 8, 0.000, 4.000

Weights_H: 1 x 8, 0.000, 4.000

Fig. 6 The Python GUI. Five major parameters classes are illustrated in the GUI: general, collision, "sub-image", visualization, and fusion parameters. The general parameters contain the camera one path which the other paths would be found based on this by just replacing the camera number. The collision parameters show the two major parameters to resolve the collision issue. The "sub-image" parameters easily help the user to set the paw region that segmentation would be done. The visualization parameters can be adjusted based on the user need. The fusion parameters are the two weight factors being used in Eq. 5. Therefore, the user can adjust the parameters based on the setup and animal. Last but not least, if a parameter would be set out of the acceptable boundary, like Demonstrating_Flag, the system will highlight it

the results by a user. We employ some keyboard shortcuts to get commands from a user in which the details can be found on GitHub Repository.

3.2 Software key parameters

The important parameters needed to be set for the software are listed as follow:

- NumSLIC. The number of segments for the SLIC method. In this paper, we refer to it as the number of superpixels. Higher number generates more superpixels, but it increases the required processing time. Default = 15,000.
- Collision_Threshold. If the paws on the same side have less than this number, "collision type II" might happen, it is referred to as "collision type II threshold." Default = 60.
- Start_Frame_Collision. This number shows the minimum frame number that "collision type II" is checked. The reason is the signal shape with just a few points can be noisy and causing mistracking. Default = 20.
- PawXWindowSize. The number of pixels in the U direction that "sub-image" is extended from its center to both sides for faster tracking. Default = 70.
- PawYWindowSize. The number of pixels in the V direction that "sub-image" is extended from its center to both sides for a faster tracking. Default = 40.
- Weights_F. This is the weight vector for the front paw defined in Section 2.3.4. Default = [2, 0, 4, 2, 2, 0, 1, 4].

- Weights_H. This is the weight vector for the hind paw defined in Section 2.3.4. Default = [2, 0, 4, 1, 2, 0, 2, 4].

All these parameters might need to be varied based on the size of the animal and the resolution of the image, expect the "weights vector." "Weights Vector" might need to be varying based on setup if the background color is not green or the animal has different color features.

4 Results

4.1 Experimental conditions

4.1.1 Animal housing conditions

We analyze data from five C57BL/6 mice, a common strain of laboratory mice, because it is the most widely used strain in basic research and biomedicine.

The animals were housed under a 12-12 h light-dark cycle in a temperature-controlled environment with food and water available ad libitum. Animal procedures were approved by the Temple University Institutional Animal Care and Use Committee, in ACUP #4675 to Andrew Spence.

4.1.2 Fusion function weights selection

This section will help researchers to adjust the weights, referred to as the fusion function, based on the setup and the animal's paw features.

The weight vectors for the fusion function could play a significant role in determining the results. However, this vector is mainly defined based on the general features of the setup and animal. The eight features have two components from each of the followings terms: green channel, red channel, hue channel, and location of the tracked paw in the previous frame. The green background preserves information in the green channel. It means that the green channel could be helpful to remove all the superpixels related to the treadmill and background. We realized that one of the channels could differentiate mouse paw from the rest of body is the red channel; however, it could get noisy, and the difference gets negligible. On the other hand, the best channel is the hue which has been studied in [35]. Finally, the location of the tracked paw in the previous frame is a critical parameter for tracking.

The weight is set based on the importance of the information that they could have. We considered the weights to be a power of 2 and used 0 for showing the feature should be excluded. In other words, 4, 2, 1, and 0 show the feature having supreme importance for tracking, are needed, show the minimum importance for tracking, and are not required, respectively. Therefore, the features based on the average of the hue channel for the paw in the initialization step, F_3, and the tracked paw coordinates from the previous frame, F_8, get the weight of 4. The features based on the average of the red (F_5) and green (F_1) channels for the paw in the initialization step can differentiate paws from the background; thus, they get a weight of 2. The average values for the red and green channels from a previous frame should not affect the tracking for the current frame because one mistake in tracking could lead to further errors. Therefore, the weight of the related features, F_2 and F_6, are considered 0.

The front paw compared to the hind paw has a smaller size, more variation in the shape and color information, the possibility to get hidden by the hind paw, and has more potential to be mislabeled with the other parts. These differences make us consider two different weights vectors for the front and hind paws, Eq. 5. The weight vectors, Weights_F and Weights_H, are similar except the related weights for F_4 and F_7.

F_4 involves the hue color information from the previous tracked frame (not the initial tracked one). It helps us to track the front paw as it had more changes. Especially in the early swing phase, from the back view, the fury part of the paw would be seen which it has a little different hue value compared with the foot part. This could not happen for the hind paw as it is much larger than front paw and always being visible. Therefore, we assigned a factor of 2 for the front and a factor of 1 for the hind paw.

F_7 shows the difference between superpixels with the bottom left corner of "sub-image". This helped to track a relatively constant point of the paw, the tip of the paw. The front paw was small, and it generated one to three superpixels which the centers of them were close to each other. This is why the weight is 1 for the front paw. While the hind paw can have more superpixels. A weight of 2 was considered for F_7 to avoid the movement of tracked paw because of this issue.

The last weight, F_8, has a factor of four as the comparison of coordinates for superpixels with the predicted paw either using the 3D KF followed by inverse DTL or using the spline fitting protocol for paws collision resolution plays an important role in labeling of paws.

It also should be mentioned that a paw (the tip of paw which is being tracked) might consist of a few superpixels close to each other. The fusion function helps us to find the best superpixel based on the color similarity (F_{1-6}) and distance from the previously tracked coordinate (F_8). However, we considered F_7 with the lowest weight of one for front paw (as it is smaller in size) and a weight of two for hind paw (as it has a larger size) to find the superpixel which is closest to the bottom left corner of "sub-image."

4.2 Mistakes in tracking

We used five mice running freely on the treadmill to analyze the performance of our algorithm. Two trials from each mouse were randomly selected. Each trial contained data from four cameras and having 1000 frames. However, the frames should meet one condition that the mouse should move forward on the belt (any direction toward front but not backward) or stay stationary. In other words, an animal should not turn back and move backward or try to climb the plexiglass walls. Therefore, 3300 frames were eliminated from the study for this reason, and a total of 36,700 frames were selected to evaluate the performance.

For all these frames, the animal was freely moving, and the speed of animal relative to the belt was varying between 0 and 64 cm/s. The average speed was 35 cm/s with a standard deviation of 15 cm/s. The treadmill was adjusting the speed using closed-loop feedback described in [55] to keep the animal in the middle of the belt for a better resolution. The frames were captured if the animal was running faster than 20 cm/s for at least 2 s (500 frames) to make sure that the animal would move during the capturing time.

We considered two parameters to evaluate the number of mistakes for our method: major and minor errors. The major error was the number of times that a user needed to intervene to avoid a "critical mistake." While a "critical mistake" was defined as a mistake for which the system could not track the paws correctly after it occurred without user intervention. The minor error was the number of times that mistakes happened which the software could still manage to track the paws after a few times wrong labeling. In other words, the second type of mistakes happened, but the system was recovered a few mistakes.

In addition, we compared our algorithm with three methods. The first method used the SLIC method for segmentation followed by a 2D KF as a tracking system; it is referred to as "SLIC+2D." The second method was the SLIC method followed by a 3D KF as tracking system; it is referred to as "SLIC+3D." In addition, the proposed algorithm results were compared to manual tracking (MT). This comparison is shown in Fig. 7 and Table 2. The 3D KF method uses the same algorithm described in Algorithm 3 while it does not apply any collision resolution steps (any steps related to splines). Also, the 2D KF uses the KF to predict the position

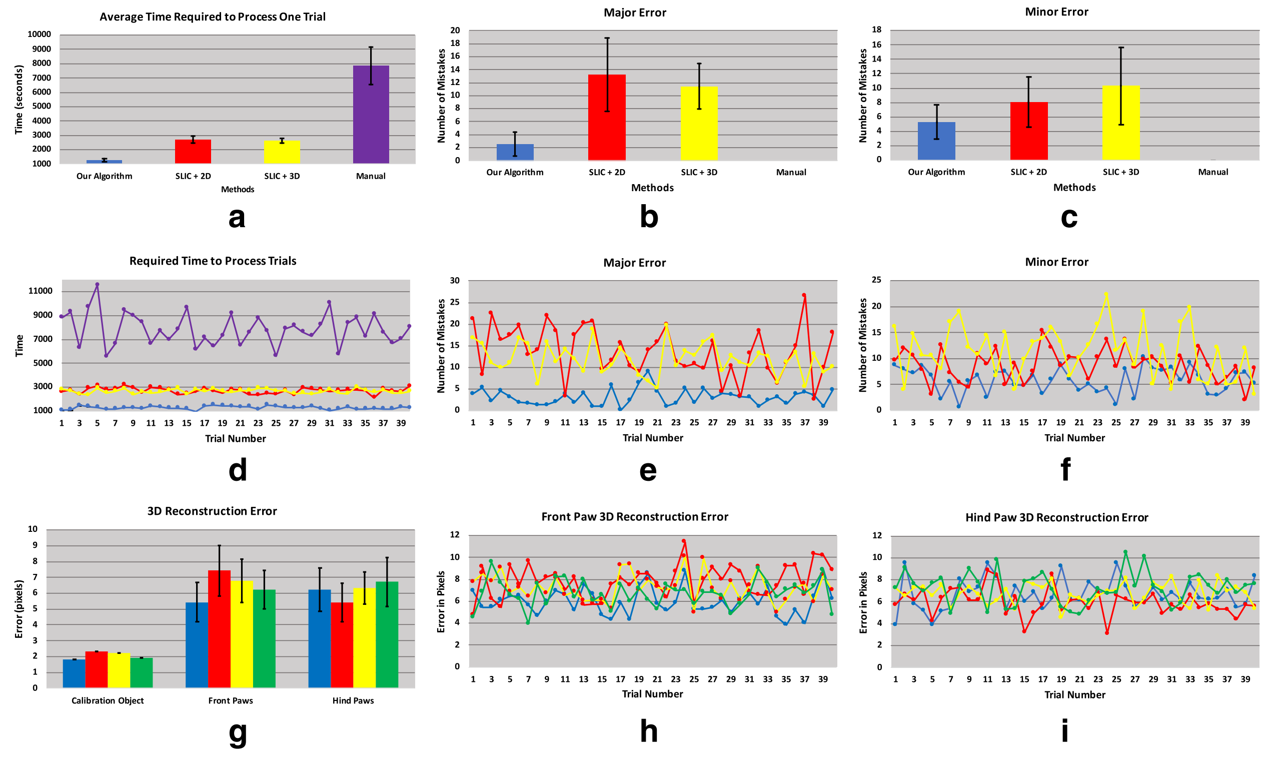


Fig. 7 The error plots. Panels **a**, **b**, and **c** show a comparison between our proposed algorithm with four other methods for the dataset (4 cameras × 2 trials each animal × 5 mice = 40 trials). These three methods are SLIC with 2D KF, "SLIC+2D"; SLIC with 3D KF, "SLIC+3D;" and MT. Panel **a** shows the required time by different methods to process one trial, 1000 frames from each of the four cameras. Panels **b** and **c** show the average number of "critical mistakes" per trials that software needed user intervention to label paws correctly after that mistake (major error) and the average number of mistakes per trials that the software did not need any user intervention to recover from the mistake (minor error), respectively. While panels **d**, **e**, and **f** show the results of each trial for the measures presented in panels **a**, **b**, and **c**, respectively. The colors show the results for the methods presented in **a**, **b**, and **c**. 3D reconstruction error, Section 4.4, is shown in **g**. It shows the error for the calibration object, front paw, and hind paw in the four cameras. The blue, red, yellow, and green colors in **g** show the results of camera 1, 2, 3, and 4. The plots **h** and **i** show the 3D reconstruction error of different trials for front and hind paws, respectively

of paws followed by the fusion function, please see Algorithm 3.

It should be noted that majority of the failings in tracking happened during the collision of paws (although we resolved most of them) or because of the fast or sudden movement of paws.

Last but not least, one of the advantages of 3D reconstruction has been shown in Fig. 8. This figure illustrates how we can find the location of paws on a camera image, which the paws were not tracked for that camera. The paws can be tracked on other side cameras. Then, based on the 3D reconstruction of tracked paws (filled circles) for those cameras using DLT, we can find the location of paw (unfilled circles) in different cameras using the DLT inverse. Also, Table 1 shows how the cameras were significantly different compared to each other for the 3D reconstruction error.

4.3 Time performance of algorithm

The performance was examined on a MacBook Pro 2.7 GHz Intel Core i5 with 8 GB 1867 MHz DDR3. The average and standard deviation of the required time to process one frame from four cameras were 0.94±0.12 seconds. The slowest steps in our algorithm were generating superpixels (0.15 ± 0.03 seconds), demonstrating the tracked coordinates on image planes if requested (0.33 ± 0.05 seconds), and loading the frames from the hard drive (0.38 ± 0.02 seconds). These numbers were calculated without considering the time needed for the initialization processing and any possible required editions. However, we investigated the average required time to process one trial, as shown in Fig. 7. The results showed an average of 1267 seconds with a standard deviation of 129 seconds for processing of 1000 frames for the four cameras.

The demonstration is an optional parameter; however, it might be necessary to visualize the tracking results and to edit the possible mistakes. We considered some keyboard shortkeys to let the user close the tracking results for a frame, ”current frame,” and see the results after a specific frame number (based on the pressed shortkey). Therefore, the user can save time for visualization based on the need for the edition.

On the other hand, the required time to perform the SLIC on a frame could be reduced if the images size was lower or the number of superpixels would be smaller. We reduced the frame size from 2048×700 to the “sub-image” size (the default is 140×80). The “sub-image” size can be adjusted by the user if needed as explained in Section 3.2.

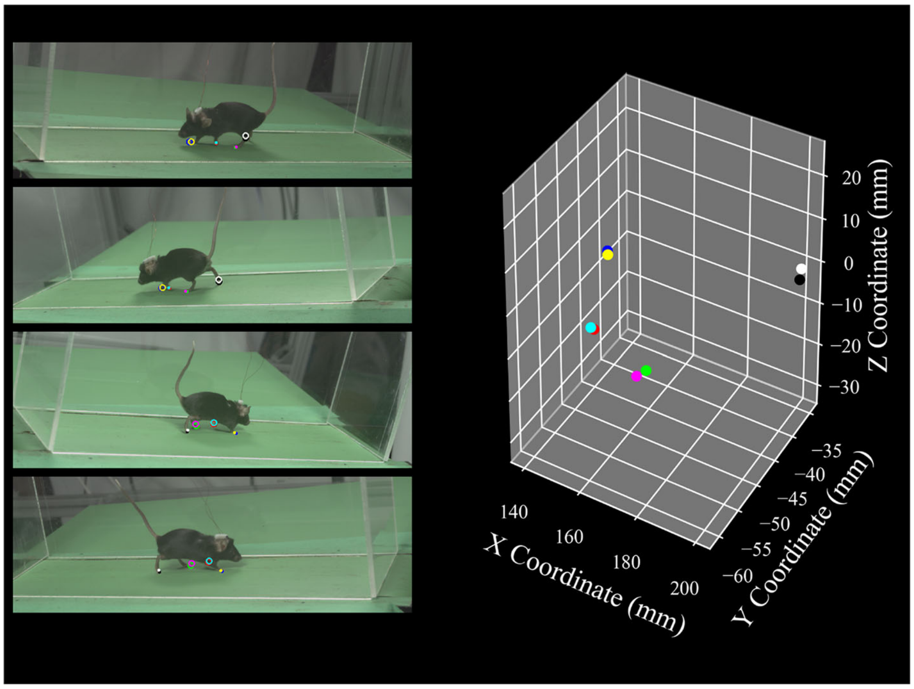


Fig. 8 A video (MP4 82.3 MB) of tracked paws for 100 consecutive frames which compare the results between MT and our proposed method (OPM). The filled and unfilled circles show the tracked paws using the cameras located on the same side of paws and the remapped of paws locations to the other side cameras relative to the paw side, respectively. The unfilled circles show the concept of transferring 2D coordinates to 3D and remapping the 3D coordinates to 2D images (in this figure to other cameras while it can come back to the same cameras and finding the 3D reconstruction error). The red, blue, green, black, cyan, yellow, magenta, and white show the tracked coordinates of paws for front right by MT, hind right by MT, front left by MT, hind left by MT, front right by OPM, hind right by OPM, front left by OPM, and hind left by OPM

Table 1 *P*-value calculated by t-test for 3D reconstruction error between cameras (shown in Fig. 7)

	cam1-cam2	cam1-cam3	cam1-cam4	cam2-cam3	cam2-cam4	cam3-cam4
Front Paw	0.00000004	0.00001030	0.00684739	0.07784151	0.00055666	0.05207830
Hind Paw	0.01414776	0.01414776	0.08251063	0.00210960	0.04616507	0.07077881

The results show that the 3D reconstruction errors for the majority of cameras were significantly different between cameras. However, the highlighted results in grey show the ones which were not significantly different with a relatively small margin

To find the best number of superpixels, we presented the details in [32]. Briefly, the number of superpixels is determined by the number of pixels in an image divided by the twice number of pixels in the object needed to be tracked. The reason is that the SLIC makes superpixels with twice or half size of the object [1]. We showed that if the size of a marker would be a known parameter, then, the following equation can find the best value for the number of superpixels:

$$\text{number of superpixels} = \frac{\text{row} \times \text{col}}{\text{mnop} \times 2} \tag{10}$$

where mnop is the minimum number of object (paws) pixels. The terms row and col are the image width and the image height, respectively. The front paw was the smallest object, and it was getting as small as 50 pixels (even sometimes getting hidden). Therefore, we considered the number of superpixels equal to 15,000. However, this number can be adjusted based on setup and animal size.

It should be noted that we did not investigate the required time for the SLIC because Achanta et al. comprehensively studied the required time for the SLIC method to generate a different number of superpixels [1].

4.4 3D reconstruction error

The 2D *UV* coordinates (on an image plane of camera) were mapped to the 3D *XYZ* coordinate system using the DLT coefficients to investigate the 3D reconstruction error. The 3D reconstruction shown in Fig. 3 is an ideal scenario while the result of mapping to 3D is two lines (ray beams of cameras) not intersecting with each other. A point should be found which is located between the two lines, and it is the closest point to both lines. Therefore, if a point in 3D was remapped to the 2D *UV* coordinate systems (on the image plane of camera) using the DLT inverse, there can be a difference between the tracked points and remapped points (which shows ideal points on the 2D images for having that point in 3D) on the image planes of cameras. Therefore, the difference between the ideal coordinates of reprojected points and the original 2D coordinates can be a factor showing how accurate the 3D reconstruction process was done. All tracked frames for each method were used to evaluate the 3D reconstruction. The resuFig. 7 and Table 2.

4.5 Adjustment function effect

As briefly discussed, the front paw is prone to more shape, size, and color changes compared with hind paw. These changes are more dramatic during the swing phase while the late stance or early swing phase of the front paw is when the "collision type II" happens. We noticed that when "collision type II" occurs, as illustrated in Algorithm 3, we can adjust the weight reducing the major error by almost 10%. The major error was 2.82 before using the function and reduced to 2.54. This was studied for three trials our of ten trials (the other trials were not evaluated).

The logic behind the selection of weight changes, illustrated in Algorithm 4, is as follow for front paw weights: (1) Incrementing the first weight by one representing F_1 (previously tracked green value) because this can help us to avoid finding dark treadmill (green spots on treadmill) as front paw. (2) Reducing the fourth weight by one representing F_4 (previously tracked hue value) because one mislabeling in the front paw could lead to a constant mislabeling. In other words, the previously tracked hue value can be critical if the fur of body is detected as the front paw (which the real front paw is dark showing the fur parts during the swing phase). (3) Setting the seventh weight equal to zero representing F_7 (bottom left coordinate of image) because the spline prediction algorithm rules the prediction of coordinates.

The severity of this issue for the hind paw was less and we adjusted one weight: incrementing the first weight by two representing F_3 (the initial hue value) because the pink parts of hind paw are completely obvious, and the hue channel carries the color information. This incrimination is done for the initial hue value because we are sure about the color information from the first frame.

Table 2 A summary of tracking results

	Our Algorithm	SLIC + 2D	SLIC + 3D	Manual
Total Number of Frames	36,700	8000	8000	4000
Time Avg	1267	2701	2650	7848
Time STD	129	236	171	1320
Major Error Avg	2.54	13.23	11.43	0
Major Error STD	1.87	5.63	3.48	0
Minor Error Avg	5.29	8.06	10.29	0
Minor Error STD	2.38	3.48	5.39	0

This table summarizes the results presented in Fig. 7

5 Discussion

It is often important to track the tips of appendages, e.g., paws, feet, hands, because they often carry extensive information relevant to a task [37]. But they are often prone to difficult and periodic occlusions.

We presented a method to solve this problem by segmenting and tracking markerless paws of running rodents on the treadmill. The treadmill arena was captured using four side view cameras; two cameras were located on each side of the treadmill, and the setup can be seen in Fig. 1. SLIC superpixel method was used for segmentation. After one or two rounds of initial tracking, which was MT, we predicted the location of paw using the 3D KF. A "sub-image" was extracted from the frame. Then, eight features described in Eq. 3 were extracted for each superpixel. These features were normalized and used to find the best match for the paw based on the coordinates and color features of the paw in the previous frame, and the color features of the paw in the first frame tracked manually. This was done using the fusion function based on a weight vector presented in Eq. 5. The proposed method requires calibration of the capture volume to extract the DLT coefficients and a spline function calculation for finding the paw motion pattern. Both of these calibrations are needed only once for a given setup and specific animal.

The main difficulties to develop a method were occlusion, three types (I, II, and III) of collisions for the paws, relatively rapid variations in the size and shape of paws, distinguishing the paws from the other parts of the body, and differentiating the paws from background and treadmill. One of the innovations that we employed was using the scipy fitting tools to match the previously tracked paw coordinates with the spline function from a typical motion of paws. It helped us to fix the most severe collision, "collision type II."

A software to modify any possible mistakes was also presented. The software gives the option of showing the results for any sequence number of frames, saving the results at any time while running the code, having the possibility to adjust the parameters, visualizing the saved results, and applying an edition at any time of tracking or visualization. Figure 7 demonstrates that our proposed method could track the paws with fewer mistakes in labeling and faster than other methods. To track a complete trial, we needed an average of 2.54 editions (to fix major errors) to have an acceptable tracked coordinates for the paws. There was 5.29 mistakes, the minor error, after fixing the major errors. Therefore, by a total of 7.83 editions, clean tracking can be achieved. However, a low-pass filter can almost remove the adverse effect of 5.29 mistakes among 1000 frames in a trial which means an average of 2.54 editions can have acceptable results. These reported values are estimates of the expected errors.

For tracking of the paws from four cameras, our presented method was about ten times faster than the manual clicking and more than two times faster than three possible combinations of SLIC with 2D KF and 3D KF. This has been illustrated in Fig. 7c. The reason that the other three methods were slower was the need for manual clicking to avoid the major error. Fig. 7a suggests that the major errors could lead to nonrecoverable mistakes and they should be edited to complete a trial which the editions were slowing down the speed of processing. However, Fig. 7b shows about one half of minor errors for our method. In addition, although KF is a powerful tool for predicting states, the results of SLIC with 2D KF and 3D KF suggest that KF in our setup would not be enough for tracking of a paw. We believe that more parameters can be involved for KF to improve its performance in future works.

We also investigated the 3D reconstruction error shown in Fig. 7d. The calibration object had an error of about 2 pixels offset between the original points and reprojected points when the 25 markers were transferred to the 3D domain, and then, they remapped to the 2D image planes. This error was varying between 5 and 8 pixels for the paws for the four cameras. The interesting point is that the two front cameras, camera 1 and 4, had a slightly better 3D reconstruction results for the front paws compared to the hind paws. While the opposite trend can be seen for the back cameras, camera 2 and 3.

The proposed algorithm is limited to quadruped animals, and it requires availability of a color difference between the paws and the body of animal. It is independent of the speed and gait; however, the animal cannot move backward (turning back and move toward back), climb the plexiglass walls, or stand on hind foods. In other words, the animal should run (just not in the reverse direction) or remain stationary. For rats and mice, these conditions happen most of the recording time. If the color features of the paw are not enough to differentiate the paws from body for an animal, researchers can develop a similar method using superpixels with extracting more higher order features (like texture features or higher order statistical color features).

It should be noted that the available methods for comparison were limited to manual segmentation or using a combination of SLIC with 2D and 3D information in this manuscript However, we have tried to employ a neural network or optical flow for tracking which we had much lower success in tracking; therefore, the results were not presented here.

As our method relies only on the general assumption that the individual legs will recirculate through space and time, but not on any specific set of phase relationships between legs, it will likely handle arbitrary gaits, or issues with individual legs (hesitation, perturbation,

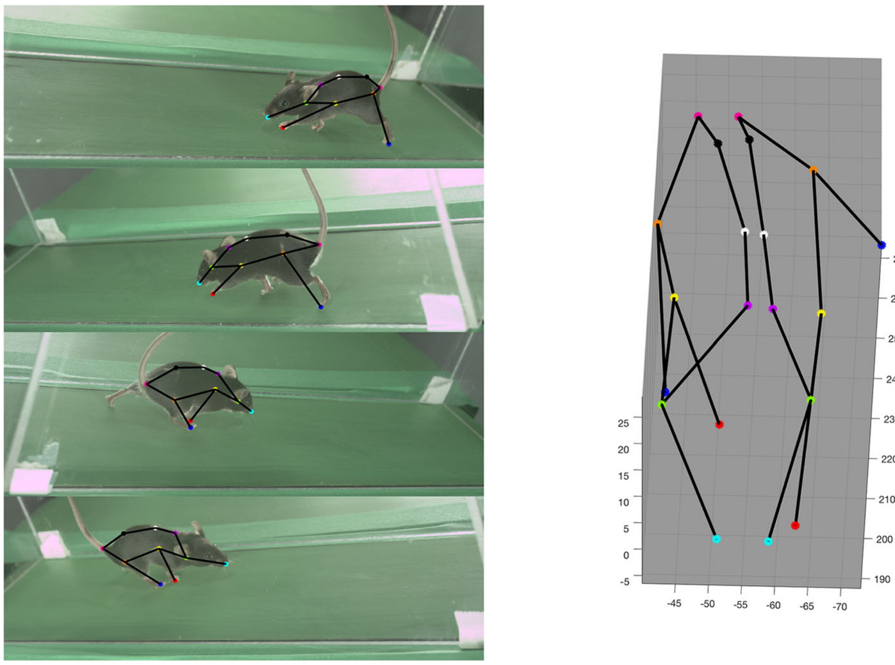


Fig. 9 3D reconstruction for a few landmarks. On the left, four camera frames are shown with ten markers. We bleached four markers on the body of the mouse to make tracking easier for the body. In addition, ear, tail, nose, two paws, and top of the curve of the body are the other six landmarks. The 3D reconstruction of these ten markers is illustrated on right

injury, stuttering), without issue. While the generality of the tracking is useful for tracking raw kinematic data from recirculating paws, future work will build on these data to construct a 3D model of the mouse body (Fig. 9). Body roll, pitch, and yaw could be investigated, along with spinal curvature, for example, to answer interesting questions in biomechanics and disease models.

6 Conclusion

We have in fact developed a method based on kinematic structure, in the sense that 3D trajectories through space and time are considered, but not kinematic in the sense of a "kinematic chain" or "skeleton" with multiple "bones" or "segments" and joints, that are part of the tracker state. The main assumption for our tracker is that the legs recirculate through a relatively flexible path in space and time. As such, it will likely capture abnormal gait, or different speeds, without difficulty. We believe that utilizing the kinematic structure of the rodent, in the sense of a detailed "skeleton" model of the animal, fit to the data, is beyond the scope of this paper, due to the difficulty of gathering raw kinematics from these animals without markers. Future work will build on these data to build such models of the rodent kinematic structure. Gait and kinematic trajectory features vary by speed and animal to animal, and it would make the method limited to specific speed or gait. This can be done using a probabilistic joint movement predictor position based on physical limitation [2] and applying more sophisticated trackers [57], [41]. Finally, we would like to develop software for kinematics study for markerless tracking as we presented for marker-based studies[34].

Abbreviations

DLT: Direct linear transform; GUI: Graphical user interface; KF: Kalman filter; MT: Manual tracking; OPM: Our proposed method SIFT: Scale-invariant feature transform: SLIC: Simple linear iterative clustering

Acknowledgements

We would like to thanks Helen Freitas for all her helps as our lab manager.

Authors' contributions

OHM conducted and analyzed the study, and he was a major contributor in writing the manuscript. AV helped for preparing the setup and gathering the data. AS supervised and guided the study. In addition, he was a contributor in writing the manuscript. All authors read and approved the final manuscript.

Funding

This work is supported by Shriners Hospitals for Children Grant #85115 to Andrew Spence. This work is further supported by Neilsen Foundation Senior Research Grant #546798 to A. Spence.

Availability of data and materials

It can be provided based on request.

Competing interests

The authors declare that they have no competing interests.

Author details

[1]Omid Haji Maghsoudi Department of Radiology, University of Pennsylvania, Philadelphia PA, 19104, USA. [2]Annie Vahedipour Department of Pediatric Neurology, Yale University, Connecticut CT, 06520, USA. [3]Department of Bioengineering, College of Engineering, Temple University, Philadelphia PA, 19122, USA.

Received: 23 April 2019 Accepted: 12 August 2019
Published online: 14 September 2019

References

1. R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, S. Süsstrunk, SLIC superpixels compared to state-of-the-art superpixel methods. IEEE Trans. Pattern. Anal. Mach. Intell. **34**(11), 2274–2282 (2012)
2. V. Belagiannis, S. Amin, M. Andriluka, B. Schiele, N. Navab, S. Ilic, 3d pictorial structures revisited: Multiple human pose estimation. IEEE Trans. Pattern. Anal. Mach. Intell. **38**(10), 1929–1942 (2016)
3. J. L. Boulland, F. M. Lambert, M. Züchner, S. Ström, J. C. Glover, A neonatal mouse spinal cord injury model for assessing post-injury adaptive plasticity and human stem cell integration. PLOS One. **8**(8), e71,701 (2013)
4. G. Bradski, A. Kaehler, *Learning OpenCV: Computer Vision with the OpenCV library*. (O'Reilly Media Inc., 2008)
5. R. G. Brown, P. Y. Hwang, et al., *Introduction to Random Signals and Applied Kalman Filtering, vol. 3*. (Wiley, New York, 1992)
6. A. M. Choo, T. R. Oxland, Improved rsa accuracy with dlt and balanced calibration marker distributions with an assessment of initial-calibration. J. Biomech. **36**(2), 259–264 (2003)
7. J. Song, H. Luo, T. L. Hedrick, Three-dimensional flow and lift characteristics of a hovering ruby-throated hummingbird. Journal of The Royal Society Interface. **11**(98), 20140541 (2014)
8. K. Clarke, J. Still, Gait analysis in the mouse. Physiol. Behav. **66**(5), 723–729 (1999)
9. D. Comaniciu, P. Meer, Mean shift: a robust approach toward feature space analysis. IEEE Trans. Pattern. Anal. Mach. Intell. **24**(5), 603–619 (2002)
10. K. Deisseroth, Optogenetics. Nat. Methods. **8**(1), 26–29 (2011)
11. R. Deumens, R. J. Jaken, M. A. Marcus, E. A. Joosten, The catwalk gait analysis in assessment of both dynamic and static gait changes after adult rat sciatic nerve resection. J. Neurosci. Methods. **164**(1), 120–130 (2007)
12. C. W. Dorman, H. E. Krug, S. P. Frizelle, S. Funkenbusch, M. L. Mahowald, A comparison of digigait^{TM} and treadscan^{TM} imaging systems: assessment of pain using gait analysis in murine monoarthritis. J. Pain. Res. **7**, 25 (2014)
13. P. F. Felzenszwalb, D. P. Huttenlocher, Efficient graph-based image segmentation. Int. J. Comput. Vis. **59**(2), 167–181 (2004)
14. V. M. Filipe, J. E. Pereira, L. M. Costa, A. C. Maurício, P. A. Couto, P. Melo-Pinto, A. S. Varejão, Effect of skin movement on the analysis of hindlimb kinematics during treadmill locomotion in rats. J. Neurosci. Methods. **153**(1), 55–61 (2006)
15. K. K. Gadalla, P. D. Ross, J. S. Riddell, M. E. Bailey, S. R. Cobb, Gait analysis in a mecp2 knockout mouse model of rett syndrome reveals early-onset and progressive motor deficits. PLOS One. **9**(11), e112,889 (2014)
16. O. Haji-Maghsoudi, A. Talebpour, H. Soltanian-Zadeh, N. Haji-maghsoodi, in *2012 16th CSI International Symposium on Artificial Intelligence and Signal Processing (AISP), pp. 116–121*. Automatic organs' detection in wce (IEEE, 2012)
17. O. Haji Maghsoudi, A. Vahedipour, A. Spence, Three-dimensional-based landmark tracker employing a superpixels method for neuroscience, biomechanics, and biology studies. Int. J. Imaging. Syst. Technol. (2019)
18. O. HajiMaghsoudi, A. Talebpour, H. Soltanian-zadeh, H. A. Soleimani, in *IEEE International Conference on Imaging Systems and Techniques (IST), pp. 18–23*. Automatic informative tissue's discriminators in wce (IEEE, 2012)
19. O. HajiMaghsoudi, A. VahedipourTabrizi, B. Robertson, P. Shamble, A. Spence, in *IEEE Symposium in Signal Processing in Medicine and Biology (SPMB), pp. 1–3*. A rodent paw tracker using support vector machine (IEEE, 2016)
20. F. P. Hamers, A. J. Lankhorst, T. J. van Laar, W. B. Veldhuis, W. H. Gispen, Automated quantitative gait analysis during overground locomotion in the rat: its application to spinal cord contusion and transection injuries. J. Neurotrauma. **18**(2), 187–201 (2001)
21. H. Hatze, High-precision three-dimensional photogrammetric calibration and object space reconstruction using a modified dlt-approach. J. Biomech. **21**(7), 533–538 (1988)
22. T. L. Hedrick, Software techniques for two-and three-dimensional kinematic measurements of biological and biomimetic systems. Bioinspiration. Biomim. **3**(3), 034,001 (2008)
23. P. Huehnchen, W. Boehmerle, M. Endres, Assessment of paclitaxel induced sensory polyneuropathy with catwalk automated gait analysis in mice. PLOS One. **8**(10), e76,772 (2013)
24. R. E. Kalman, et al., A new approach to linear filtering and prediction problems. J. Basic. Eng. **82**(1), 35–45 (1960)
25. D. H. Theriault, N. W. Fuller, B. E. Jackson, E. Bluhm, D. Evangelista, Z. Wu, M. Betke, T. L. Hedrick, et al., A protocol and calibration method for accurate multi-camera field videography, jeb–100529 (1960)
26. T. L. Hedrick, B. W. Tobalske, I. G. Ros, D. R. Warrick, A. A. Biewener, et al., in *Proceeding of Royal Society B: Biological Sciences*. Morphological and kinematic basis of the hummingbird flight stroke: scaling of flight muscle transmission ratio, vol. 279 (The Royal Society, 2012), pp. 1986–1992
27. T. Karakostas, S. Hsiang, H. Boger, L. Middaugh, A. C. Granholm, Three-dimensional rodent motion analysis and neurodegenerative disorders. J. Neurosci. Methods. **231**, 31–37 (2014)
28. C. Kim, F. Li, A. Ciptadi, J. M. Rehg, in *Proceedings of the IEEE International Conference on Computer Vision, pp. 4696–4704*. Multiple hypothesis tracking revisited, (2015)
29. A. Kyme, S. Se, S. Meikle, G. Angelis, W. Ryder, K. Popovic, D. Yatigammana, R. Fulton, Markerless motion tracking of awake animals in positron emission tomography. IEEE Trans. Med. Imaging. **33**(11), 2180–2190 (2014)
30. Z. Ma, A. B. Chan, Counting people crossing a line using integer programming and local features. IEEE Trans. Circ. Syst. Video. Technol. **26**(10), 1955–1969 (2016)
31. O. H. Maghsoudi, M. Alizadeh, M. Mirmomen, in *IEEE Symposium in Signal Processing in Medicine and Biology (SPMB), pp. 1–6*. A computer aided method to detect bleeding, tumor, and disease regions in wireless capsule endoscopy (IEEE, 2016)
32. O. H. Maghsoudi, A. V. Tabrizi, B. Robertson, A. Spence, in *51st Asilomar Conference on Signals, Systems, and Computers, pp. 209–213*. Superpixels based marker tracking vs. hue thresholding in rodent biomechanics application (IEEE, 2017)
33. O. H. Maghsoudi, A. Vahedipour, J. Gerstenhaber, S. P. George, T. Hallowell, B. Robertson, M. Short, A. Spence, Matlab software for impedance spectroscopy designed for neuroscience applications. J. Neurosci. Methods. **307**, 70–83 (2018)
34. O. H. Maghsoudi, A. Vahedipour, T. Hallowell, A. Spence, Open-source python software for analysis of 3d kinematics from quadrupedal animals. Biomed. Sig. Process. Control. **51**, 364–373 (2019)
35. O. H. Maghsoudi, A. Vahedipour, B. Robertson, A. Spence, Application of superpixels to segment several landmarks in running rodents. J. Pattern Recog. Image Anal. **28**, 468–483 (2018)
36. A. Mahdi, H. Omid, S. Kaveh, R. Hamid, K. Alireza, T. Alireza, Detection of small bowel tumor in wireless capsule endoscopy images using an adaptive neuro-fuzzy inference system. J. Biomed. Res. (2017)
37. H. M. Maus, S. Revzen, J. Guckenheimer, C. Ludwig, J. Reger, A. Seyfarth, Constructing predictive models of human running. J. R. Soc. Interface. **12**(103), 20140,899 (2015)
38. A. A. Migliaccio, R. Meierhofer, C. C. Della Santina, Characterization of the 3d angular vestibulo-ocular reflex in c57bl6 mice. Exp. Brain. Res. **210**(3-4), 489–501 (2011)
39. R. J. Moore, G. J. Taylor, A. C. Paulk, T. Pearson, B. van Swinderen, M. V. Srinivasan, Fictrac: a visual method for tracking spherical motion and generating fictive animal paths. J. Neurosci. Methods. **225**, 106–119 (2014)
40. S. Nori, Y. Okada, S. Nishimura, T. Sasaki, G. Itakura, Y. Kobayashi, F. Renault-Mihara, A. Shimizu, I. Koya, R. Yoshida, et al., Long-term safety issues of ipsc-based cell therapy in a spinal cord injury model: oncogenic transformation with epithelial-mesenchymal transition. Stem Cell Rep. **4**(3), 360–373 (2015)
41. S. Oh, S. Russell, S. Sastry, in *Decision and Control, 2004. CDC. 43rd IEEE Conference on, vol. 1, pp. 735–742*. Markov chain monte carlo data association for general multiple-target tracking problems (IEEE, 2004)
42. T. E. Oliphant, Python for scientific computing. Comput. Sci. Eng. **9**(3) (2007)
43. G. N. Orlovski, T. Deliagina, S. Grillner, *Neuronal control of locomotion: from mollusc to man*. (Oxford University Press, 1999)
44. S. S. Parvathy, W. Masocha, Gait analysis of c57bl/6 mice with complete freund's adjuvant-induced arthritis using the catwalk system. BMC. Musculoskelet. Disord. **14**(1), 14 (2013)
45. R. Penjweini, S. Deville, O. Haji Maghsoudi, K. Notelaers, A. Ethirajan, M. Ameloot, Investigating the effect of poly-l-lactic acid nanoparticles carrying hypericin on the flow-biased diffusive motion of hela cell organelles. J. Pharm. Pharmacol. (2017)
46. D. F. Preisig, L. Kulic, M. Krüger, F. Wirth, J. McAfoose, C. Späni, P. Gantenbein, R. Derungs, R. M. Nitsch, T. Welt, High-speed video gait analysis reveals early and characteristic locomotor phenotypes in mouse

models of neurodegenerative movement disorders. Behav. Brain. Res. **311**, 340–353 (2016)
47. B. Přibyl, P. Zemčík, M. Čadík, Absolute pose estimation from line correspondences using direct linear transformation. Comp. Vision. Image. Underst. (2017)
48. S. Revzen, J. M. Guckenheimer, Finding the dimension of slow dynamics in a rhythmic system. J. R. Soc. Interface. **9**(70), 957–971 (2012)
49. S. H. Rezatofighi, A. Milan, Z. Zhang, Q. Shi, A. R. Dick, I. D. Reid, in *ICCV, pp. 3047–3055*. Joint probabilistic data association revisited, (2015)
50. B. L. Roth, Dreadds for neuroscientists. Neuron. **89**(4), 683–694 (2016)
51. J. Shi, J. Malik, Normalized cuts and image segmentation. IEEE Trans. Pattern. Anal. Mach. Intell. **22**(8), 888–905 (2000)
52. B. Smith, in *AdvancED ActionScript 3.0: Design Patterns, pp. 1–25*. Object-oriented programming (Springer, 2011)
53. J. L. Smith, P. Carlson-Kuhta, T. V. Trank, Forms of forward quadrupedal locomotion. iii. a comparison of posture, hindlimb kinematics, and motor patterns for downslope and level walking. J. Neurophysiol. **79**(4), 1702–1716 (1998)
54. M. Sonka, V. Hlavac, R. Boyle, *Image processing, analysis, and machine vision*. (Cengage Learning, 2014)
55. A. J. Spence, G. Nicholson-Thomas, R. Lampe, Closing the loop in legged neuromechanics: an open-source computer vision controlled treadmill. J. Neurosci. Methods. **215**(2), 164–169 (2013)
56. A. D. Straw, K. Branson, T. R. Neumann, M. H. Dickinson, Multi-camera real-time three-dimensional tracking of multiple flying animals. J. R. Soc. Interface., p. rsif20100230 (2010)
57. R. L. Streit, T. E. Luginbuhl, in *Proceedings of the Sixth Joint Service Data Fusion Symposium, pp. 1015–1024*. A probabilistic multi-hypothesis tracking algorithm without enumeration and pruning, (1993)
58. C. J. Veenman, M. J. Reinders, E. Backer, Resolving motion correspondence for densely moving points. IEEE Trans. Pattern. Anal. Mach. Intell. **23**(1), 54–72 (2001)
59. O. Veksler, Y. Boykov, P. Mehrani, in *European Conference on Computer Vision, pp. 211–224*. Superpixels and supervoxels in an energy optimization framework (Springer, 2010)
60. S. Van der Walt, S. van der walt, jl schönberger, j. nunez-iglesias, f. boulogne, jd warner, n. yager, e. gouillart, t. yu, and the scikit-image contributors, peerj 2, e453 (2014). PeerJ. **2**, e453 (2014)
61. A. B. Wiltschko, M. J. Johnson, G. Iurilli, R. E. Peterson, J. M. Katon, S. L. Pashkovski, V. E. Abraira, R. P. Adams, S. R. Datta, Mapping sub-second structure in mouse behavior. Neuron. **88**(6), 1121–1135 (2015)
62. Q. Xu, C. Cai, H. Zhou, H. Ren, in *International Conference on Optoelectronics and Image Processing (ICOIP), vol. 1, pp. 181–184*. A video tracking system for limb motion measurement in small animals (IEEE, 2010)
63. C. Yan, Y. Tu, X. Wang, Y. Zhang, X. Hao, Y. Zhang, Q. Dai, Stat: spatial-temporal attention mechanism for video captioning. IEEE Trans. Multimed. (2019)

Publisher's Note