BJMB
Brazilian Journal of Motor Behavior

Tutorials

Markerless tracking in sports using DeepLabCut: A methodological tutorial

TÉRCIO APOLINÁRIO-SOUZA[1] | THIAGO J. LEONARDI[2]

[1] Universidade Federal do Rio Grande do Sul, Porto Alegre, RS, Brazil
[2] Universidade Estadual de Campinas, Campinas, SP, Brazil

Correspondence to: Tércio Apolinário-Souza, address: R. Felizardo, 750 - Jardim Botânico, Porto Alegre - RS, Brasil; CEP: 90690-200
email: edf.tercio@gmail.com
https://doi.org/10.20338/bjmb.v19i1.484

ABBREVIATIONS

AI	Artificial intelligence
GUI	Graphical user interface
NaN	Not a number

PUBLICATION DATA
Received 09 05 2025
Accepted 14 11 2025
Published 24 11 2025

ABSTRACT
One of the major challenges in Motor Behavior research is translating findings from controlled environments to complex, real-world contexts such as team sports. Recent advances in machine learning, particularly deep learning, have enhanced the ability to track and analyse movement in dynamic settings. DeepLabCut, an open-source toolbox using deep neural networks, enables precise tracking of players and objects, offering valuable insights for performance analysis. This tutorial aims to provide practical guidance for implementing DeepLabCut and to evaluate its effectiveness in tracking both athletes and the ball during gameplay, thereby supporting its application in sports science research and professional practice.

KEYWORDS: Behavior Analysis | Computer Vision | Machine Learning | Motion Tracking | Pose Estimation

INTRODUCTION

A key challenge in the field of Motor Behavior is bridging the gap between insights obtained from controlled, simplified tasks and their application in complex, real-world contexts typical of professional practice [1]. This challenge is partly due to the lack of tools capable of accurately assessing Motor Behavior constructs in such dynamic and complex environments.

Team sports often reveal this challenge more prominently. Team sports constitute a complex and dynamic system shaped by the interactions of multiple elements [2]. Athletes' behavior depends on continuous adaptations to factors such as spatiotemporal demands[3]. Performance analysis research commonly focuses on individual and collective tactical behaviors—often relying on positional data—to inform training and competition strategies [4]. Over the decades, sports analysis techniques have seen remarkable advancements [5], particularly with the advent of artificial intelligence (AI). Notably, developments in AI have enabled accurate human motion tracking in dynamic and natural environments [6,7,8].

DeepLabCut, an open-source toolbox based on deep learning, exemplifies these advancements in AI [9]. DeepLabCut is a technology capable of tracking behavior across different contexts, offering a practical, low-cost, and simple-to-implement solution for the study of human movement [10]. In sports science, DeepLabCut may be used to track athletes' positions and ball dynamics during gameplay. This application of DeepLabCut enables detailed performance analysis and tactical evaluations, providing insights into player movements, strategies, and interactions within a game context. Integrating such advanced tracking technology into sports analysis represents a significant leap forward, offering researchers and coaches a deeper understanding of athletic performance and strategy.

However, a significant barrier to leveraging these methods is the complexity involved in setup, training, and evaluation, which often lacks user-friendly guidance. Consequently, an accessible, tutorial-style article walks readers through practical implementation steps and demonstrates how these tools can be deployed across diverse applications, such as sports science. This tutorial aims to outline practical implementation procedures and to assess the effectiveness of DeepLabCut in tracking both players and the ball.

METHODS

Step 1 - Recording video

The first step is to record the video. We recommended using a drone for this purpose, as it allows the camera to remain centred, thereby facilitating image processing in the subsequent steps. However, if recording with a drone is not possible, position the camera to ensure the image is as centred as possible.

Two pairs (four men aged 32± 3.0 years) played in a small-sided game to maintain possession of the ball for as long as possible over 3 minutes. Ball handling should be performed with the hands, and the participant was not allowed to move while in possession of the ball (similar to basketball). If the ball crossed the established boundaries, possession was awarded to the opposing team of the last player to touch the ball. For more details about the game, please refer to the "Original video" in the online materials. This experiment occurred on an open court within a delimited 6 x 6 meters (Figure 1). The data were recorded at 30 Hz with a resolution of 640×480 pixels using a drone (DJI Mini SE) positioned at a height of 10 meters. The moments in the video where the ball left the designated area ('ball out of play') were removed, leaving 107 seconds remaining (online material: "Original video").

Step 2 - Extract the frames

The next step is to extract the frames and label the frames of interest. These frames represent the key points that the user intends to track. DeepLabCut provides a function that allows all extracted frames to be labelled using an interactive graphical user interface (GUI). In our example, using the GUI, we localized the center of nine key points (the ball, the four players, and the four corners of the area) across 20 frames sampled using the k-means clustering approach within DeepLabCut. The 20 frames selected by the k-means clustering approach are the default setting in DeepLabCut; however, the number of frames and k-means clustering can be adjusted as needed.

Step 3 - Create training dataset

We recommend using Google Colab with GPU support to create, train, and evaluate the model. Personal computers often lack the necessary hardware configurations (e.g., dedicated graphics cards) to perform such procedures efficiently. In our example, we run the model using the Google Colab GPU. To create the model, we employed the ResNet-50 architecture, a deep residual network known for its high performance in image recognition tasks. Additionally, we utilized the `imgaug` library for data augmentation, which enhances the variability of the training data by applying random transformations such as rotations, flips, changes in brightness, and contrast (online material or supplementary material: "Script 2"). However, the parameters used to create the dataset can be modified.

Step 4 - Train network

To train the model, use the function `deeplabcut.train_network`. This function initiates the training process for the neural network dedicated to pose estimation. To enhance the model's robustness, employ the first shuffle of the dataset, providing a varied training set. To monitor the progress, configure the training to display the status every 100 iterations, offering frequent feedback on the model's performance. Additionally, set the function to save the model checkpoint every 500 iterations, ensuring that training progress is preserved and allowing for resumption from the latest checkpoint in case of interruptions. Finally, define the training process to run for a maximum of 10,000 iterations, preventing overfitting and ensuring efficient use of computational resources. Stopped the training when the loss function value reached below 0.005 with a learning rate of 0.005.

You can perform the same procedure using Script 2 available in the online material or supplementary material.

Step 5 - Evaluation of the Trained Network

The fifth step is to evaluate the performance of the trained model. We assess the model's performance by calculating the mean average Euclidean error between the manual labels and the labels predicted by DeepLabCut. The acceptable error threshold for each model depends on the level of precision required for the analysis. In our example, the results showed that the mean training error was 2.62 pixels, while the mean test error was 3.79 pixels. This error is relatively low for our application.

Step 6 - Perspective transformation

Centering the camera poses a challenge during recording; therefore, an important step is the perspective transformation. This transformation helps minimise image distortion effects. Drone footage results in a video with fluctuations in the camera's position due to the wind's effect on the drone. In this example, we used known fixed points (the four corners) to perform the perspective transformation, as described below.

We apply a perspective transformation to each frame, using the four known fixed points in the image that delineate the playing area.

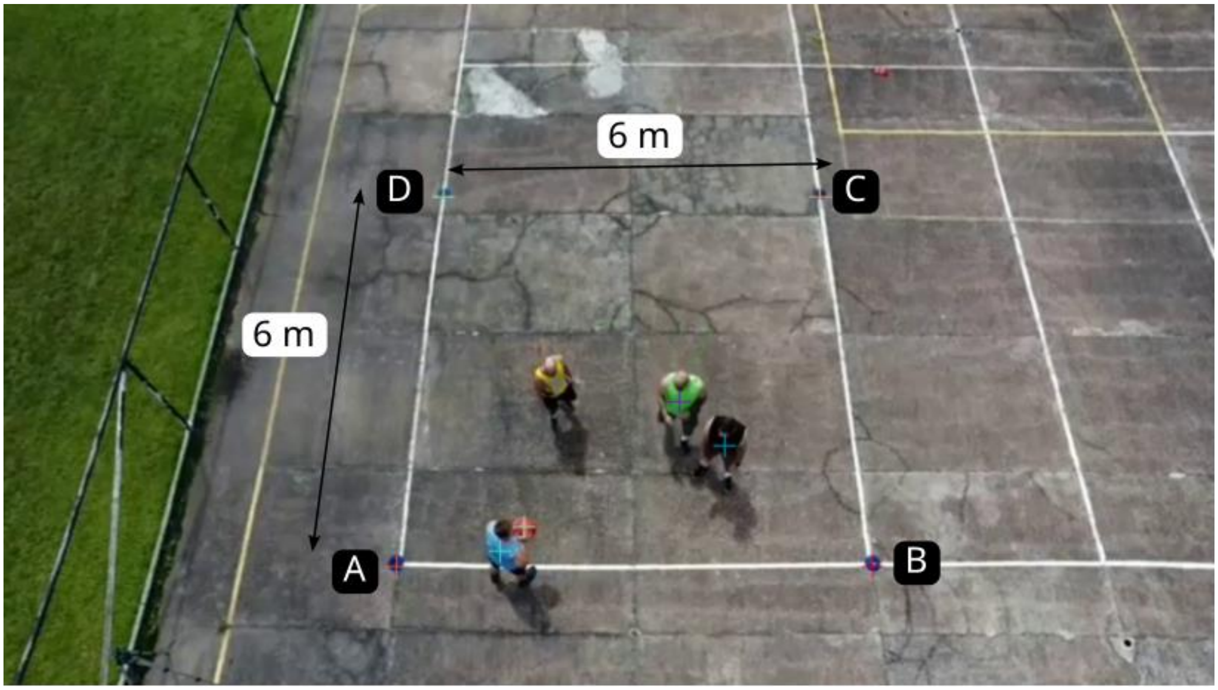

Figure 1. The known points in the image for perspective transformation are A, B, C, and D, with a distance of 6 meters between A and B, D and C. m = meters.

Our implementation computed the homography matrix (for example, at the 4-second mark in Figure 2) and applied the perspective transformation (online material or supplementary material: "Script 3"). First, we extracted each frame's four corner points:

$$\textit{Bottom} - \textit{left} = (x_A, y_A)$$
$$\textit{Bottom} - \textit{right} = (x_B, y_B)$$
$$\textit{Top} - \textit{right} = (x_C, y_C)$$
$$\textit{Top} - \textit{left} = (x_D, y_D)$$

Then, we calculated the distances between these points to find the mean width and height:

$$\text{width}_{AD} = \sqrt{(x_A - x_D)^2 + (y_A - y_D)^2}$$
$$\text{width}_{BC} = \sqrt{(x_B - x_C)^2 + (y_B - y_C)^2}$$
$$\text{meanWidth} = \frac{\text{width}_{AD} + \text{width}_{BC}}{2}$$

$$\text{height}_{AD} = \sqrt{(x_A - x_B)^2 + (y_A - y_B)^2}$$
$$\text{height}_{BC} = \sqrt{(x_C - x_D)^2 + (y_C - y_D)^2}$$
$$\text{meanHeight} = \frac{\text{height}_{AB} + \text{height}_{CD}}{2}$$

Based on the known real-world distance, we derive the pixels-per-meter ratio:

$$\text{pixelsPerMeterHorizontal} = \frac{\text{meanWidth}}{\text{realDistance}}$$
$$\text{pixelsPerMeterVertical} = \frac{\text{meanHeight}}{\text{realDistance}}$$
$$\text{pixelsPerMeter} = \frac{\text{pixelsPerMeterHorizontal} + \text{pixelsPerMeterVertical}}{2}$$

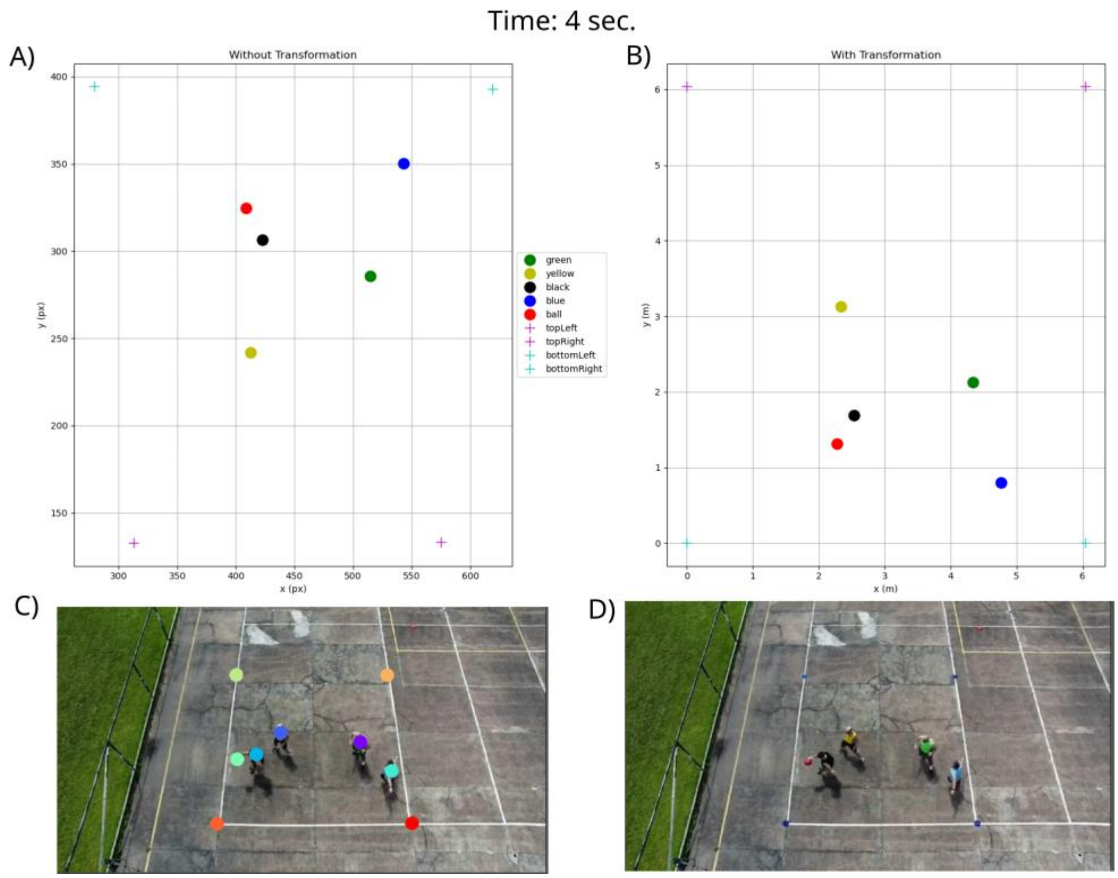

Figure 2. Graphical representation of the points in the image and the image at the 4th second. A) Graphical representation of the points in the image without the transformation. B) Graphical representation of the points in the image with the transformation. C. Image with the indication of points of interest. D. Original image. m = meters. px = pixels.

Next, we define the desired coordinates in the new image using the pixels per meter ratio. Assume d is the actual distance between the cones:

$$\text{pts\_dest} = \begin{bmatrix} 0 & 0 \\ d \cdot \text{pixels\_per\_meter} & 0 \\ d \cdot \text{pixels\_per\_meter} & d \cdot \text{pixels\_per\_meter} \\ 0 & d \cdot \text{pixels\_per\_meter} \end{bmatrix}$$

Here, pts_dest represents the coordinates in the destination plane, a transformed version of the original coordinates. Each point is represented in the format [x,y]:

1. [0, 0]: bottom-left corner;
2. [*d* · pixels_per_meter, 0]: bottom-right corner;
3. [*d* · pixels_per_meter, d · pixels_per_meter]: top-right corner;
4. [0, *d* · pixels_per_meter]: top-left corner.

we calculated the homography matrix *H* using cv2.getPerspectiveTransform fuction of OpenCV:

H = cv2.getPerspectiveTransform(pts_orig, pts_dest)

For each point p = [*x*, *y*] of the elements, it is transformed using the homography matrix *H*:

$$p' = H \cdot p$$

where p=[x,y,1] is the transformed coordinate. After, we performed by multiplying the point vector by the homography matrix *H*:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Finally, we normalize the resulting vector by dividing it by the homogeneous coordinates:

$$x' = \frac{h_{11}x + h_{12}y + h_{13}}{h_{31}x + h_{32}y + h_{33}}$$
$$y' = \frac{h_{21}x + h_{22}y + h_{23}}{h_{31}x + h_{32}y + h_{33}}$$

However, the method for performing the perspective transformation should be adjusted according to the type of recording. We always recommended using known points from the recording to carry out the perspective transformation.

Step 7 - The filter with likelihood

The data exported from the analysis includes a column indicating the probability that each signal is correct. It is advisable to disregard low-probability values. In the example below, we used probability values greater than 0.8. The following outlines the step-by-step procedure for our example.

After performing the perspective transformation of the data, we filtered the x and y position values based on the probability. Based on their likelihood values, the equation replaces values with "NaN" (Not a Number) in the x and y position rows. Given a matrix $E \in R^{3\times n}$ where E represents the positions (E_1= position x, E_2= position y) and likelihoods (E_3) of elements, as follows:

$$M_j = \begin{cases} 1 & \text{if } E_{3,j} < 0.80 \\ 0 & \text{otherwise} \end{cases} \quad \text{for } j = 1, 2, \ldots, n$$

$$E_{1,j} = \begin{cases} \text{NaN} & \text{if } M_j = 1 \\ E_{1,j} & \text{otherwise} \end{cases} \quad \text{for } j = 1, 2, \ldots, n$$

$$E_{2,j} = \begin{cases} \text{NaN} & \text{if } M_j = 1 \\ E_{2,j} & \text{otherwise} \end{cases} \quad \text{for } j = 1, 2, \ldots, n$$

Step 8 - The filter with threshold limits

Even after the final filtering, some data may not reflect reality. For instance, they may indicate positional values that do not exist in the real world. In our example, the playing area was limited to 6 by 6 meters, with an escape zone of approximately 2 meters. Therefore, any positional data above 8 meters or below -2 meters can be excluded from the analysis. The following illustrates our example.

We established two threshold limits: an upper limit of 8 and a lower limit of -2. The filtering process then replaced any values in the dataset that exceeded the upper limit or fell below the lower limit with NaN. We subsequently added the filtered data to the d matrix. This approach excludes data points outside the defined range, improving the dataset's accuracy and relevance for further analysis (Figure 3). The filtered data matrix $d_{filtered}$ is obtained by:

$$
\mathbf{d}_{\text{filtered}} = \begin{cases} \begin{bmatrix} \text{NaN} \\ \text{NaN} \end{bmatrix} & \text{if } x > \lim_{\text{up}} \text{ or } y > \lim_{\text{up}} \text{ or } x < \lim_{\text{down}} \text{ or } y < \lim_{\text{down}} \\ \begin{bmatrix} x \\ y \end{bmatrix} & \text{otherwise} \end{cases}
$$

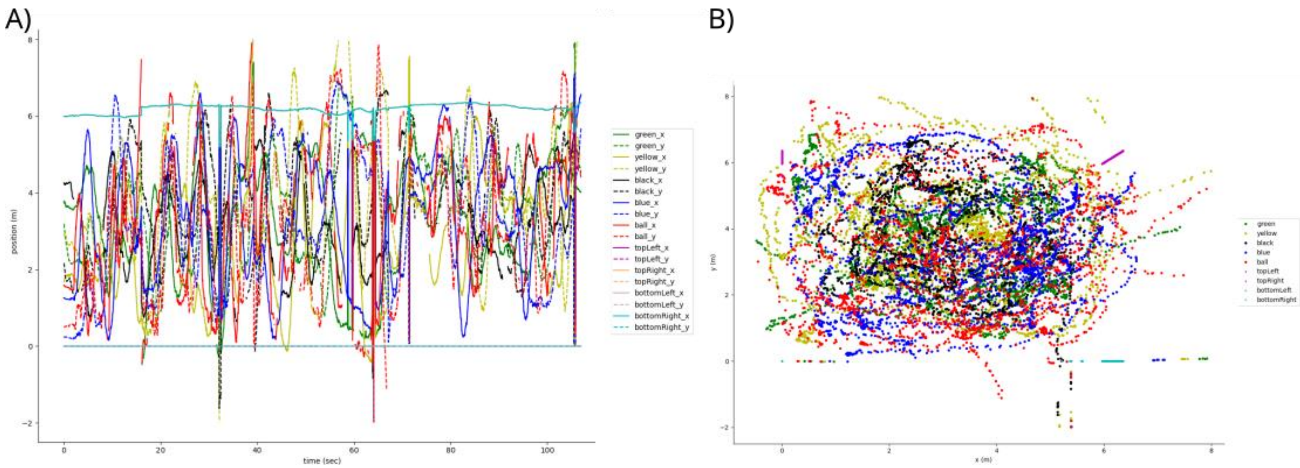

Figure 3. Graphical representation of points of interest over time (A) and their x and y positions (B). m = meters.

The data from the player wearing the black shirt and the ball showed a loss of around 50% of the data (Figure 4). This result was likely influenced by recording outdoors. The black color of the shirt did not stand out against the shadows projected by the players, making it difficult for the algorithm to recognize it. As for the ball, the sunlight reflection created a color distinct from its red hue. However, upon analyzing the complete video data (online material: "Video with DeepLabCut"), it is evident that the losses of the ball and the player in the black shirt were distributed over time, having little impact on the tracking. This indicates that, although the data loss was relatively high, the phenomena that can be investigated in this video (e.g., ball passes) occur at a relatively high frequency (around 3Hz).

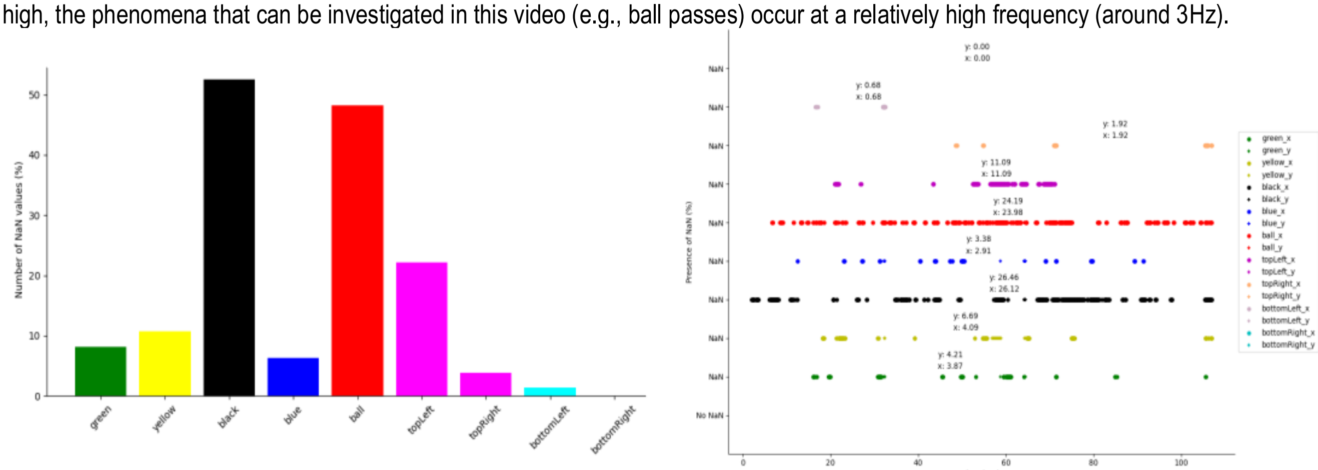

Figure 4. Percentage of frames without information. A) Comparison among the 9 elements. B) The nine elements over time. sec = seconds

FINAL CONSIDERATIONS

This tutorial aimed to outline practical implementation procedures and to assess the effectiveness of DeepLabCut in tracking both players and the ball. We hope it provides accessible guidance for researchers and practitioners unfamiliar with this tool.

Although our application focused on ball sports, DeepLabCut can be used to track any element in an image, such as body segments or implements, particularly in the field of Motor Behavior. Finally, this represents the first attempt to present a tutorial specifically tailored to DeepLabCut. We apologise if any aspect was insufficiently covered, as some steps may require basic programming knowledge. We hope this tutorial proves useful for those planning to adopt DeepLabCut in future investigations.

DATA AVAILABILITY

All scripts, videos, and other materials used in this paper can be found at the following link: https://github.com/apolinario-souza/DeepLabCut_sports or supplementary material. You can perform the same procedures using our original video ("Original video", available in the online material) and scripts (online material or supplementary material).

For definitions of the concepts of AI, please see [11].

REFERENCES

1. Wulf G, Shea CH. Principles derived from the study of simple skills do not generalize to complex skill learning. *Psychon Bull Rev*. 2002;*9*(2):185-211. doi: 10.3758/BF03196276.
2. Reilly T, Cabri J, Araújo D. Applications of dynamical systems theory to football. In *Science and Football V*. Routledge. 2005;570-572.
3. Sheehan WB, Tribolet R, Watsford ML, Novak AR, Rennie M, Fransen J. Tactical analysis of individual and team behaviour in professional australian football: Original investigation. *Science and Medicine in Football.* 2022;*6*(2):172–180. doi: 10.1080/24733938.2021.1923792.
4. Low B, Coutinho D, Gonçalves B, Rein R, Memmert D, Sampaio J. A systematic review of collective tactical behaviours in football using positional data. *Sports Med*. 2020;*50:*343–385. doi: 10.1007/s40279-019-01194-7
5. Camomilla V, Bergamini E, Fantozzi S, Vannozzi G. Trends supporting the in-field use of wearable inertial sensors for sport performance evaluation: A systematic review. *Sensors.* 2018;*18*(3):873. doi: 10.3390/s18030873.
6. Akan S, Varlı S. Use of deep learning in soccer videos analysis: survey. *Multimed Syst.* 2023;*29*(3):897–915. doi: 0.1007/s00530-022-01027-0.
7. Jia Y, Abdullah AN, Eliza H, Lu Q, Si D, Guo H, Wang W. A narrative review of deep learning applications in sports performance analysis: current practices, challenges, and future directions. *BMC Sports Sci Med Rehabil.* 2025;*17*(1):249. doi: 10.1186/s13102-025-01294-0.
8. Liu K, *Yang W, Chen M, Zheng K, Zeng X, Zhang S, Liu C*. Deep-Learning-Based Wireless Human Motion Tracking for Mobile Ship Environments, in *IEEE Internet of Things Journal. 2022;*9(23):24186-24198. doi: 10.1109/JIOT.2022.3189698.
9. Mathias A, Mamidanna P, Cury KM, Abe T, Murthy VN, Mathis MW, Bethge M. DeepLabCut: markerless pose estimation of user-defined body parts with deep learning. *Nat Neurosci.* 2018*;*21:1281–1289. doi: 10.1038/s41593-018-0209-y.
10. Nath T, Mathis A, Chen AC, Patel A, Bethge M, Mathis MW. Using DeepLabCut for 3D markerless pose estimation across species and behaviors. *Nat Protoc.* 2019*;*14:2152–2176. doi: 10.1038/s41596-019-0176-0.
11. Cossich VRA, Carlgren D, Holash RJ, Katz L. Technological Breakthroughs in Sport: Current Practice and Future Potential of Artificial Intelligence, Virtual Reality, Augmented Reality, and Modern Data Visualization in Performance Analysis. *Appl Sci. 2023;13*(23):12965. doi: 10.3390/app132312965.

Citation: Apolinário-Souza T, Leonardi TJ. (2025). Markerless Tracking in Sports using DeepLabCut: A Methodological Tutorial. *Brazilian Journal of Motor Behavior*, 19(1):e484.
Editor-in-chief: Dr Fabio Augusto Barbieri - São Paulo State University (UNESP), Bauru, SP, Brazil.
Associate editors: Dr José Angelo Barela - São Paulo State University (UNESP), Rio Claro, SP, Brazil; Dr Natalia Madalena Rinaldi - Federal University of Espírito Santo (UFES), Vitória, ES, Brazil; Dr Renato de Moraes – University of São Paulo (USP), Ribeirão Preto, SP, Brazil.

Funding: This work was supported by the Brazilian National Council for Scientific and Technological Development (CNPq, process 405316/2021-6).
Competing interests: The authors have declared that no competing interests exist.
DOI: https://doi.org/10.20338/bjmb.v19i1.484