SoftwareX 31 (2025) 102272

Contents lists available at ScienceDirect

SoftwareX

journal homepage: www.elsevier.com/locate/softx

Original Software Publication

PyBodyTrack: A python library for multi-algorithm motion quantification and tracking in videos

Angel Ruiz-Zafra [a,*], Janet Pigueiras-del-Real [b], Jose Heredia-Jimenez [c,d], Syed Taimoor Hussain Shah [e], Syed Adil Hussain Shah [f], Lionel C. Gontard [b,g]

[a] Department of Software Engineering, University of Granada, Granada, 18071, Spain
[b] Department of Condensed Matter Physics, University of Cádiz, Cádiz, 11510, Spain
[c] Department of Physical Education & Sports, University of Granada, 18071 Granada, Spain
[d] Human Behavior & Motion Analysis Lab (HubemaLab), University of Granada, 51001 Ceuta, Spain
[e] PolitoBIOMed Lab, Department of Mechanical and Aerospace Engineering, Politecnico di Torino, Turin, 10129, Italy
[f] Department of Research and Development (R&D), GPI SpA, Trento, 38123, Italy
[g] IMEYMAT, University of Cadiz, Cádiz, 11510, Spain

ARTICLE INFO

Keywords:
Motion quantification
Pose tracking
Computer vision
Human movement analysis
Python library

ABSTRACT

Human movement analysis, driven by computer vision and pose tracking technologies, is gaining acceptance in healthcare, rehabilitation, sports, and daily activity monitoring. While most approaches focus on qualitative analysis (e.g., pattern recognition), objective motion quantification can provide valuable insights for diagnosis, progress tracking, and performance assessment. This paper introduces *PyBodyTrack*, a Python library for motion quantification using mathematical methods in real-time and pre-recorded videos. It simplifies video management and integrates with position estimators like MediaPipe, YOLO, and OpenPose. *PyBodyTrack* enables seamless motion quantification through standardized metrics, facilitating its integration into various applications.

Code metadata

Current code version	2025.3.3
Permanent link to code/repository used for this code version	https://github.com/ElsevierSoftwareX/SOFTX-D-25-00184
Permanent link to Reproducible Capsule	https://colab.research.google.com/drive/1-XW_-IOAOICfwuKssuBBBAeVdZyELOTy
Legal Code License	Apache 2.0
Code versioning system used	Git
Software code languages, tools, and services used	Python
Compilation requirements, operating environments & dependencies	Python 3.9, mediapipe==0.10.21, numpy==2.2.4, opencv-python==4.11.0.86 pandas==2.2.3, scipy==1.15.2, ultralytics==8.3.78, setuptools==44.1.1
If available Link to developer documentation/manual	
Support email for questions	angelrzafra@gmail.com

Software metadata

Current software version	2025.3.3
Permanent link to executables of this version	For example: https://github.com/combogenomics/DuctApe/releases/tag/DuctApe-0.16.4
Permanent link to Reproducible Capsule	https://github.com/bihut/PyBodyTrack/
Legal Software License	Apache 2.0
Computing platforms/Operating Systems	Any platform that supports Python
Installation requirements & dependencies	mediapipe, numpy, opencv, pandas, scipy, ultralytics, setuptools
If available, link to user manual - if formally published include a reference to the publication in the reference list	https://github.com/bihut/PyBodyTrack/tree/main/docs/build/html
Support email for questions	angelrzafra@gmail.com

* Corresponding author.
E-mail address: angelr@ugr.es (A. Ruiz-Zafra).

https://doi.org/10.1016/j.softx.2025.102272
Received 19 March 2025; Received in revised form 18 June 2025; Accepted 11 July 2025

1. Motivation and significance

Human movement analysis is an expanding field with applications in biomechanics, healthcare, sports, motion-based interfaces, and daily activity tracking, among others. Motion analysis in healthcare can be employed for early detection and diagnosis of medical conditions, such as brain injuries affecting the motor system, or in rehabilitation therapies, where it serves as an indicator of patient progress. Additionally, it has applications in everyday scenarios, such as in sports, where it is used to ensure the correct execution of exercises [1,2].

From a methodological perspective, motion analysis can be approached qualitatively (e.g., identifying movement patterns or assessing whether an activity is performed correctly) or quantitatively (e.g., measuring the total or partial amount of detected movement). Although research on motion analysis is expanding, most studies focus on qualitative aspects, while standardized and accessible libraries that enable precise movement quantification in measurement units remain a challenge [3–5].

To address current limitations in movement quantification, we introduce *PyBodyTrack*, an open-source library designed for pose-based motion analysis. *PyBodyTrack* supports real-time video capture and batch processing of pre-recorded videos, providing key movement metrics: *MOS* (Movement per Second), *MOF* (Movement per Frame), *MOL* (Movement per Landmark), *NMI* (Normalized Movement Index), and *RAM* (Raw Amount of Movement). By leveraging pose estimators such as MediaPipe®,[1] YOLO®,[2] and OpenPose®,[3] the library extracts precise body coordinates, enabling structured movement analysis through distance-based and optical flow methods. Its seamless integration with Pandas facilitates efficient data handling for statistical and machine learning applications.

PyBodyTrack democratizes access to advanced motion analysis tools for researchers, developers, and professionals in fields like rehabilitation, sports science, and pediatric care. Its modular architecture allows for customization of movement metrics and adaptation to specific research needs, such as studying different populations (e.g., neonates, athletes) or integrating domain-specific metrics for motor control, rehabilitation, or biomechanics.

Thus, *PyBodyTrack* not only combines flexibility with ease of use to foster innovation in movement analysis applications, but also aims to drive research and development in the field, promoting an ecosystem of accessible, efficient and scalable tools for future applications.

1.1. Related work

Various software tools and libraries have been developed for human movement analysis, ranging from manual tracking applications to deep learning-based solutions.

One widely used tool is *Kinovea*, an open-source software for motion analysis in sports and rehabilitation. It provides functionalities such as slow-motion playback, video annotations, and motion tracking through video overlays for comparing gestures. However, its reliance on manual and semi-automated tracking, without deep learning-based pose estimation, limits its capacity for fully automated and real-time movement quantification [6]. In contrast, *PyBodyTrack* leverages pose estimation models such as MediaPipe, YOLO, and OpenPose to extract precise body coordinates and compute movement metrics without manual intervention, making it more suitable for large-scale research applications.

Several Python libraries address mobility analysis from different perspectives. *scikit-mobility* and *Trackintel* focus on large-scale mobility modeling rather than detailed motion quantification [5,7], while *MovingPandas* enables spatial trajectory analysis but lacks integrated pose estimation [8]. Unlike these, *PyBodyTrack* extracts precise body coordinates and computes movement metrics in real-time.

In addition to software-based solutions, hardware-assisted systems have been developed for movement tracking. *MOVE-HUMAN* is a specialized system for three-dimensional human movement capture in workplace environments. It employs a stereo vision setup with two synchronized digital cameras and custom software to reconstruct and analyze worker movement, primarily for ergonomic risk assessments [9]. While effective for occupational studies, its dependence on dedicated hardware limits its accessibility. In contrast, *PyBodyTrack* requires only a video input, making it a more flexible solution applicable across diverse domains such as biomechanics, healthcare, and sports.

Compared to these existing solutions, *PyBodyTrack* stands out by offering a fully automated, deep learning-based, open-source approach to human movement quantification.

2. Software description

PyBodyTrack is a Python library for quantifying human body movement from video frames, whether in real-time or pre-recorded. It utilizes deep learning-based pose tracking to detect and monitor body landmarks without physical contact.

With a high-level interface, it allows developers to capture frames, apply pose estimation, extract body coordinates, and analyze movement using built-in methods. *PyBodyTrack* is available via `pip`, released under the Apache 2.0 license, and its source code, examples, and documentation can be found on GitHub.

2.1. Software architecture

Fig. 1 illustrates the architecture of the *PyBodyTrack* library. This library consists of various modules, organized according to the required functionalities. These modules (or sub-modules) include:

(a) **bodyparts**. Defines body part specifications (e.g., nose, left knee) for different pose estimators (MediaPipe, YOLO, OpenPose) and groups landmarks into nine body regions.
(b) **enums**. Contains enumerated types for pose estimators and input modes (live capture or pre-recorded video).
(c) **methods**. Implements mathematical approaches for movement quantification.
(d) **poseestimators**. Provides pose tracking implementations for MediaPipe, YOLO, and OpenPose, encapsulated in classes for simplified usage.
(e) **observer**. Enables real-time movement quantification using an observer design pattern [10] to send results at predefined frame intervals.
(f) **posetracker**. Manages video capture and processing, abstracting complexities for developers.
(g) **utils**. Offers utility methods for dataframe cleaning, date handling, and data extraction.
(h) **BodyTracking**. The core class for executing all functionalities, allowing users to specify capture mode, pose estimator, and real-time processing while abstracting technical complexities.

The use of *PyBodyTrack* or its integration into other systems is illustrated in the workflow shown in Fig. 2.

PyBodyTrack supports two data acquisition methods: real-time recording from a camera or processing a pre-recorded video (1). The library iterates over video frames (2), applies a pose estimator (3) to extract movement data, and assigns a Unix Epoch timestamp to each entry. Supported pose estimators include MediaPipe, YOLO, and OpenPose.

Data capture operates in two modes (4): *(1) Real-time processing*, where information is transmitted every X frames (e.g., 24–30 fps), and *(2) Batch processing*, where the entire video is analyzed at once.

[1] https://ai.google.dev/edge/mediapipe/solutions/guide
[2] https://docs.ultralytics.com/es
[3] https://github.com/CMU-Perceptual-Computing-Lab/openpose

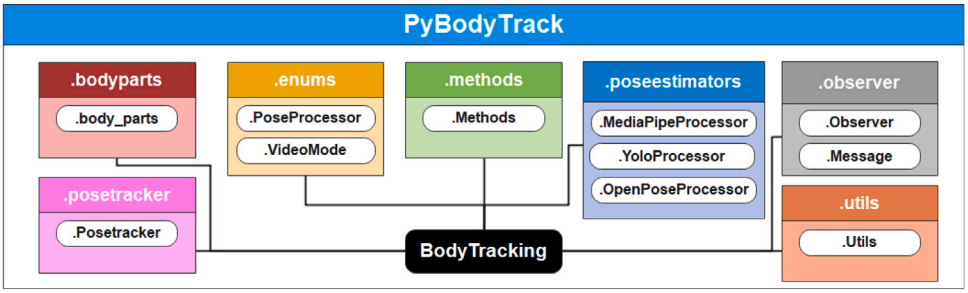


Fig. 1. PyBodyTrack architecture.

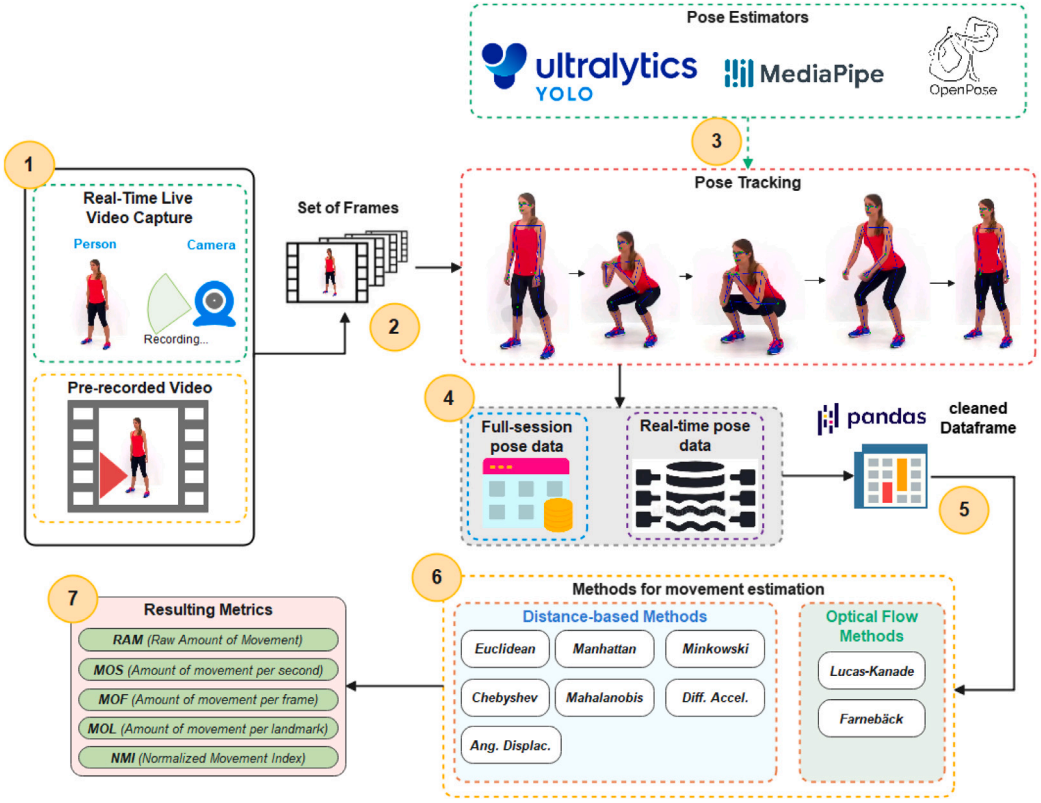


Fig. 2. PyBodyTrack workflow.

The extracted data are preprocessed and stored in a Pandas DataFrame (5), which serves as input for movement quantification methods (6). These methods fall into two categories: *(1) distance-based* and *(2) optical flow-based*. Table 1 summarizes the supported methods, including their descriptions and main equations. Distance-based methods, such as Euclidean distance, rely on simple calculations, whereas optical flow methods (Lucas–Kanade, Farnebäck) involve more complex processing.

Finally, each computational method provides a total movement value, which, when integrated with the DataFrame, enables further movement metric calculations (7), as detailed in Table 1.

2.2. Software functionalities

The library's key features include:

- **Two operation modes**: real-time recording (camera) or playback (pre-recorded video). For video playback, a specific time interval (start and end seconds) can be selected, avoiding manual trimming.
- **Multiple pose estimators**: Users can choose between YOLO, MediaPipe, and OpenPose, allowing flexibility in accuracy, speed, and computational requirements. Thus, *PyBodyTrack* does not evaluate the accuracy of these estimators, but rather uses their output directly for motion quantification.
- **Custom landmark selection**: Beyond full-body and predefined regions (e.g., torso, upper body), users can specify custom landmarks, enhancing adaptability for diverse motion analysis needs.
- **Nine movement quantification methods**: Implemented as class methods with uniform input parameters, these methods operate independently of each other and the data source, allowing multiple applications for comparison across population groups.
- **Real-time data access**: An Observer pattern enables real-time subscription to processed data from video or camera sources. In real-time mode, the FPS depends on the camera and system performance, as *PyBodyTrack* processes frames as they arrive.
- **Comprehensive output metrics**: Provides total quantified movement and raw movement data but also other useful metrics like a normalized movement index or the amount of movement per second.

Table 1
Summary of computational methods and movement metrics of PyBodyTrack.

Mathematical Methods		
Method	Description	Equation
Euclidean Distance [11]	Measures the direct displacement between two points in a three-dimensional space.	$dE = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2 + (z_2 - z_1)^2}$
Manhattan Distance [12]	Calculates the total absolute difference between coordinates, following a grid-based path.	$dM = \lvert x_2 - x_1\rvert + \lvert y_2 - y_1\rvert + \lvert z_2 - z_1\rvert$
Chebyshev Distance [13]	Determines the maximum difference along any coordinate axis.	$dC = \max\{\lvert x_2 - x_1\rvert, \lvert y_2 - y_1\rvert, \lvert z_2 - z_1\rvert\}$
Minkowski Distance [14]	Generalizes Euclidean and Manhattan distances with a parameter p.	$dM = \left(\lvert x_2 - x_1\rvert^p + \lvert y_2 - y_1\rvert^p + \lvert z_2 - z_1\rvert^p\right)^{1/p}$
Mahalanobis Distance [15]	Computes distance considering variable correlations using the covariance matrix.	$dMH = \sqrt{(A - B)^T S^{-1} (A - B)}$
Differential Acceleration [16]	Evaluates changes in acceleration of reference points across consecutive frames.	$DAcc = \sum_{i=1}^{N} \sum_{d \in \{x,y,z\}} \lvert A_{diff}^{(d)}\rvert$
Angular Displacement [17]	Measures angular variation of reference points between consecutive frames.	$DA_{angular} = \sum_{i=1}^{N} \sum \lvert\theta_i\rvert$
Lucas–Kanade [18]	A differential method for estimating optical flow based on spatial and temporal gradients.	$LK = (A^T A)^{-1} A^T b$
Farnebäck [19]	Estimates motion by analyzing optical flow based on pixel displacement.	$F = \sum_{x,y} \lvert\mathbf{F}(x, y)\rvert$
Movement Metrics		
MOS (Amount of Movement per Second)	Calculates the movement rate by dividing the total movement by the duration of the analyzed period.	MOS $= \frac{M}{t_n - t_0}$ where M is the total movement, and $t_n - t_0$ represents the duration of the analyzed period, with t_0 being the initial timestamp and t_n the final timestamp.
MOF (Amount of Movement per Frame)	Computes the movement rate by dividing the total movement by the number of intervals between analyzed frames.	MOF $= \frac{M}{N-1}$ where M is the total movement, and $N - 1$ represents the number of intervals between frames, with N being the total number of frames.
MOL (Amount of Movement per Landmark)	Computes the movement rate by dividing the total movement by the number of analyzed landmarks.	MOL $= \frac{M}{L}$ where M is the total movement, and L represents the number of landmarks used in the analysis.
NMI (Normalized Movement Index)	Computes the movement rate by dividing the total movement by the product of the analyzed period duration and the number of landmarks.	NMI $= \frac{M}{(t_n - t_0) \cdot L}$ where M is the total movement, $t_n - t_0$ represents the duration of the analyzed period, and L is the number of landmarks used in the analysis.
RAM (Raw Amount of Movement)	The original movement value returned by the method.	

3. Illustrative examples

This section presents four illustrative examples from different application areas. MediaPipe was used as the pose estimator due to its comprehensive coverage and highest number of body landmarks (33 vs. 16 in YOLO and 25 in OpenPose). Each example utilizes data from a specific dataset, which is indicated accordingly.

3.1. Motion quantification in sports applications

The movement was quantified for common gym exercises fundamental in sports. Videos from Kaggle datasets and YouTube [20,21] were used, covering plank, leg extension, bicep curl, skull crushers, bench press, chin-ups, military press, and squat. Each exercise included 10 repetitions, except the plank, which was held for 10 s.

Relevant body landmarks were selected per exercise (e.g., torso and arms for bicep curls, full body for squats). Movement was measured using the Euclidean distance method with a smoothing filter and Kalman filters.

Fig. 3 shows processed frames and a bar chart of the normalized movement index (NMI). As expected, the plank involved minimal movement, while squats had the highest. This quantification can aid in automatic repetition counting or, combined with biometric data, support calorie estimation

3.2. Movement quantification in daily activities

This example showcases *PyBodyTrack*'s ability to quantify movement in daily activities like sitting and walking using data from [22].

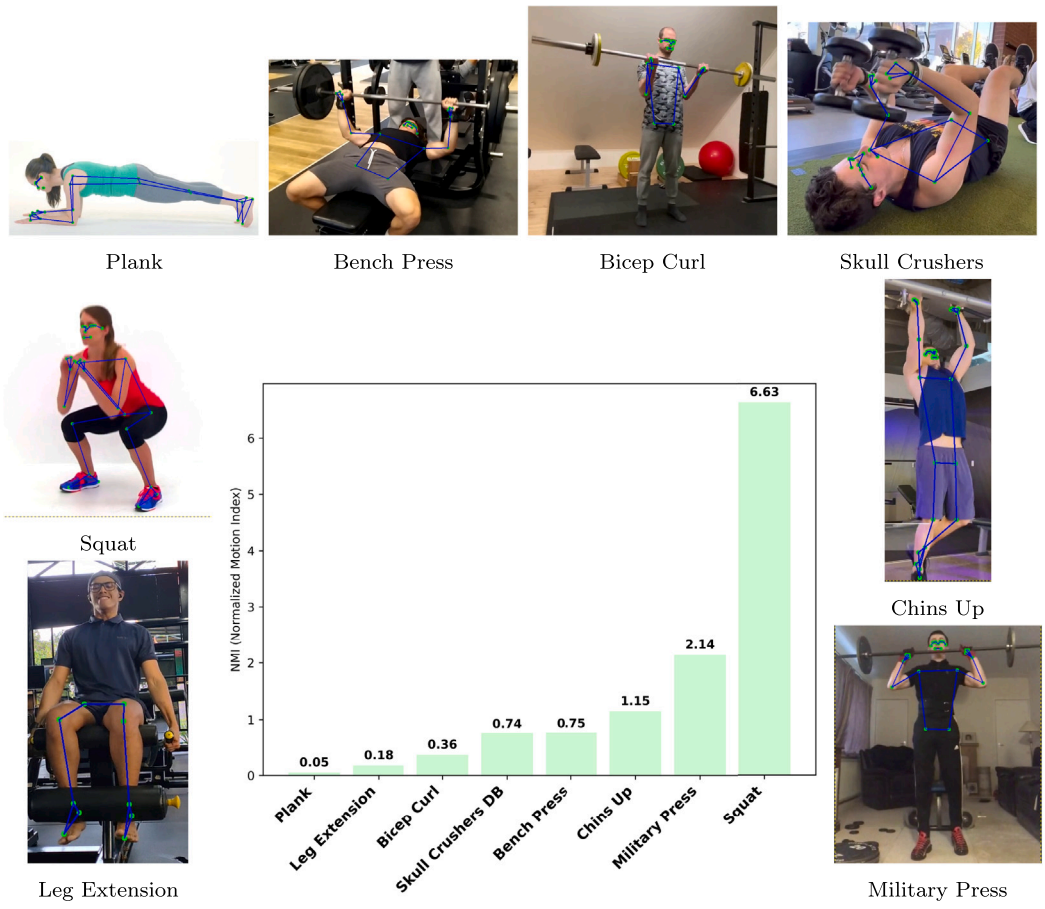


Fig. 3. Movement quantification in sports applications.

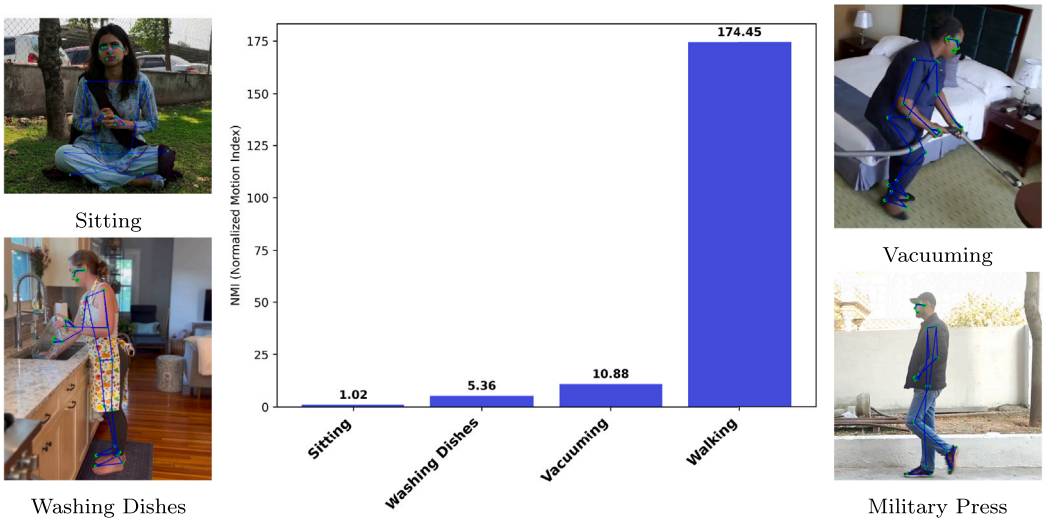


Fig. 4. Movement quantification in daily activities.

Fig. 4 presents the activities and their normalized movement index (NMI) in a bar chart.

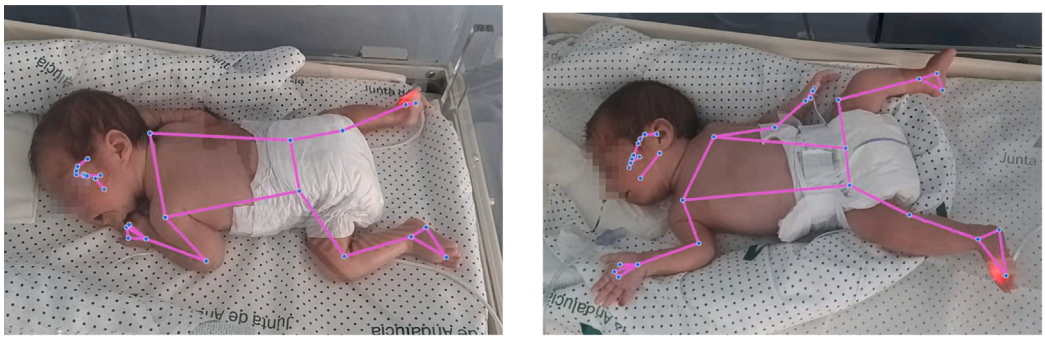
Movement was measured using the Euclidean distance method with a smoothing filter and Kalman filters.

Quantifying movement beyond sports (Example 1) has valuable applications in healthcare, such as monitoring motor function in rehabilitation, and in workplaces to assess task performance.

This approach allows contactless, device-free tracking, eliminating the need for wearable sensors like accelerometers, which have been the predominant trend in recent years [23].

3.3. Progress monitoring in preterm neonates

Neonatal progress is typically assessed through weight, size, and growth metrics, but recent studies suggest movement quantity could be an additional medical indicator [24].



(a) Frame obtained from the hospital admission video

(b) Frame obtained from the hospital discharge video

Fig. 5. PyBodyTrack for movement quantification of preterm infants.

PyBodyTrack was applied to a preterm neonate at two time points — hospital admission and discharge — using data collected with the solution presented in [25], accessible via *NeoVault* [26].[4]

Videos from both time points were analyzed, selecting random 30-second segments for processing. Fig. 5 displays frames after applying *PyBodyTrack*.

The Chebyshev method yielded 4,221 movement units (NMI: 3.917) at hospital admission and 7,249 units (NMI: 5.12) at discharge, representing a 71.74% increase.

These results highlight a significant increase in movement over time, suggesting that *PyBodyTrack* could be a valuable tool in hospital settings.

Listing 1 illustrates its application in this case.

Listing 1: Example of using PyBodyTrack for movement quantification in neonates.

```
import threading, time
from pybodytrack.BodyTracking import BodyTracking
from pybodytrack.enums.PoseProcessor import
    PoseProcessor
from pybodytrack.bodyparts import body_parts as bp
from pybodytrack.enums.VideoMode import VideoMode
from pybodytrack.methods.methods import Methods

# Define the video path
video = "PATH TO VIDEO"

# Initialize BodyTracking using Mediapipe in VIDEO
    mode with standard landmarks (whole body)
bt = BodyTracking(
  processor=PoseProcessor.MEDIAPIPE, mode=
    VideoMode.VIDEO,
  path_video=video, selected_landmarks=bp.
    STANDARD_LANDMARKS
)
bt.set_times(10, 40)  # Set start and end times in
    seconds

# Create and start the tracking thread
t = threading.Thread(target=bt.start, kwargs={})
t.start()

try:
  # Main thread loop: wait while tracking is active
  while t.is_alive():
    time.sleep(1)
except KeyboardInterrupt:
  print("Stopping tracking...")
finally:
  # Stop tracking and ensure the thread is properly
    joined
  bt.stop()
  t.join()
  if t.is_alive():
    print("Tracker thread still alive. Force
    stopping...")
    bt.stop()
'''
Note: The use of threads is not mandatory with
    PyBodyTrack in video mode (only in real-time),
    but it speeds up processing and visualization.
'''
# Retrieve data and compute movement metrics
df = bt.getData()
m = Methods.chebyshev_distance(df, filter=True,
    distance_threshold=2.0)
n = bt.normalized_movement_index(m, len(bp.
    STANDARD_LANDMARKS))
print("Raw movement:", m, "- NMI:", n)
```

3.4. Real-time fall detection

This example demonstrates *PyBodyTrack* for real-time fall detection using a camera. A 3.5-year-old child simulated a fall to test the system's effectiveness.

The Euclidean distance method was applied with a Kalman filter but without smoothing (threshold: 0 units). Movement was quantified every 8 frames, yielding three data points per second.

Fig. 6 displays movement data over time alongside the child's postural state during the simulation.

Extrapolating this example, *PyBodyTrack* could be highly useful for fall detection in the elderly and for continuous movement monitoring in rehabilitation sessions.

Since real-time quantification in *PyBodyTrack* relies on the Observer component, a subclass was implemented to demonstrate its use. Listing 2 illustrates how *PyBodyTrack* captures and processes real-time movement data.

Listing 2: Example of real-time capture and processing with PyBodyTrack for fall detection.

```

import json, threading, time, pandas as pd
from pybodytrack.BodyTracking import BodyTracking
from pybodytrack.enums.PoseProcessor import
    PoseProcessor
from pybodytrack.bodyparts import body_parts as bp
from pybodytrack.enums.VideoMode import VideoMode
from pybodytrack.methods.methods import Methods
```

[4] https://conversational.ugr.es/neovault

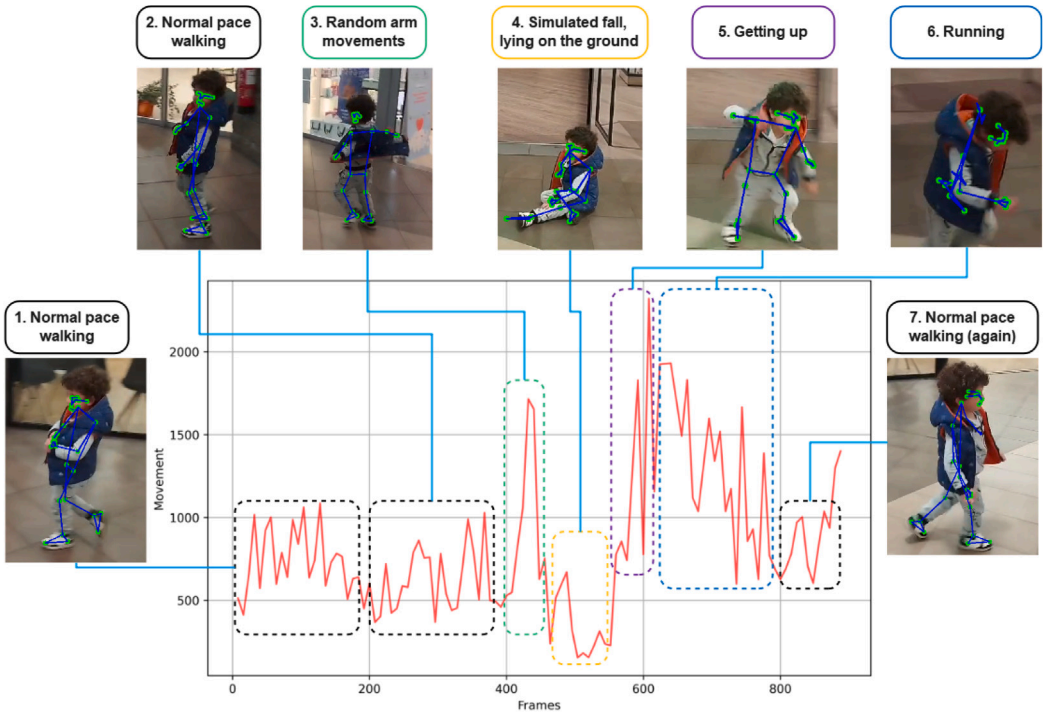


Fig. 6. PyBodyTrack applied to real-time fall detection.

```python
from pybodytrack.observer.Observer import Observer

# Global list to store results
res_json = []

class CustomObserver(Observer):
    def __init__(self, block_size=30):
        super().__init__()
        self.pkg_count = 0     # Counter for processed frames
        self.block_size = block_size  # Number of frames per block

    def handleMessage(self, msg):
        # If new landmark data is received, process it in a new thread
        if msg.what == 1:
            threading.Thread(target=self.processBuffer, args=(msg.obj,), daemon=True).start()
        else:
            print("Error:", msg.obj)

    def processBuffer(self, block):
        # Convert the incoming block to a DataFrame
        df = pd.DataFrame(block)
        # Update the package counter with the block size
        self.pkg_count += self.block_size
        # Compute the Euclidean distance as a movement measure
        mv = Methods.euclidean_distance(df, filter=True, distance_threshold=0.0)
        # Normalize the movement index using the number of standard landmarks
        nmi = body_tracking.normalized_movement_index(mv, len(bp.STANDARD_LANDMARKS))
        # Append the result with time, movement, and normalized movement index to the global results
        res_json.append({"time": self.pkg_count, "movement": mv, "nmi": nmi})

# Set up body tracking for fall detection using a camera
output = "results_falldetection.json"
body_tracking = BodyTracking(
    processor=PoseProcessor.MEDIAPIPE, mode=VideoMode.CAMERA,
    path_video=None, selected_landmarks=bp.STANDARD_LANDMARKS
)
fps = 8  # Frames per second
observer = CustomObserver(block_size=fps)
observer.startLoop()  # Start the observer loop

# Start body tracking in a separate thread, passing the observer and fps
t = threading.Thread(target=body_tracking.start, kwargs={'observer': observer, 'fps': fps})
t.start()

try:
    # Main thread idle loop while tracking is active
    while t.is_alive():
        time.sleep(1)
except KeyboardInterrupt:
    print("Stopping tracking...")
    body_tracking.stop()
finally:
    # Ensure tracking stops and the thread is properly joined
    body_tracking.stop()
    t.join()
    if t.is_alive():
        print("Force stopping...")
        body_tracking.stop()
```

```
# Write the results to a JSON file
with open(output, "w", encoding="utf-8") as f:
    json.dump(res_json, f, indent=4)
```

4. Impact

Position estimation methods based on computer vision are highly powerful tools for acquiring body movement data without physical contact, whether in real time or from pre-recorded videos. Currently, numerous studies explore these techniques, as the extracted data have a wide range of applications, including healthcare, sports, daily activity monitoring, and any field where human motion analysis is relevant [27,28].

This work introduces *PyBodyTrack*, a library that simplifies and enhances movement estimation and quantification. It allows for the analysis of either the entire body or specific regions using various mathematical approaches, both from pre-recorded videos and real-time sources. Its primary goal is to provide a set of objective, numerical motion metrics to support applications in different professional domains. To illustrate its potential, four case studies have been conducted in different contexts: sports activity monitoring, daily task analysis, neonatal development assessment, and fall detection. These examples highlight its versatility and ease of use. It is important to note that, for the presented case studies — and for any potential new application — a preliminary calibration phase was necessary to select the most appropriate combination of analysis parameters. This phase involves testing different methods, time windows, and body regions using representative examples from the target context. Such calibration ensures that the resulting movement quantification is meaningful and adapted to the specific characteristics of each use case.

PyBodyTrack is the first open-source library designed to quantify movement without requiring prior knowledge of pose tracking or computer vision. In addition to integrating multiple motion estimation methods, it allows users to customize the analyzed body regions and apply advanced data-processing filters. The library incorporates three position estimators (MediaPipe, OpenPose, and YOLO) and provides configurable options that combine different analytical approaches (Euclidean, Chebyshev, Farnebäck), body region selection (full body, torso, limbs), and specific filters (threshold distance, Kalman filter parameters). This flexibility enables motion quantification optimization based on the target population or activity under study.

One known limitation is that distance-based quantification methods may be sensitive to the subject's distance from the camera, as pose estimators typically return coordinates in pixel space. To address this, users are encouraged to maintain a fixed camera–subject distance or apply normalization techniques based on body proportions (e.g., shoulder width). Incorporating such variations during the calibration phase can also help ensure stable and meaningful measurements across different scenarios.

On the other hand, it is also worth noting that the evaluation of *PyBodyTrack's* performance should be adapted to each application domain. Comparative validation should consider population-specific benchmarks—for example, using the *General Movements Assessment* (GMA) for neonates [29] or functional mobility tests like the *Timed Up and Go* (TUG) test for older adults [30].

Due to its seamless integration with third-party software and broad applicability, *PyBodyTrack* has the potential to make a significant impact. It can be used both as a standalone tool for researchers and developers and as an integrated component in independent applications aimed at human motion analysis.

This library is part of the doctoral research of one of the authors. Initially, the project focused on neonatal movement quantification as an indicator of favorable development within the European project PARENT.[5] Over time, the research evolved into the current version of *PyBodyTrack*, expanding its applicability and scope.

5. Conclusions

PyBodyTrack is an open-source Python library designed to quantify movement in specific activities using a set of objective metrics. The library supports both real-time analysis via a camera and the processing of pre-recorded videos. It enables motion capture using different pose estimators, provides multiple adjustable movement quantification methods, and allows the selection of specific body regions for analysis.

Beyond its core functionality, *PyBodyTrack* is designed to be used either as a standalone tool or integrated into other solutions, allowing developers to incorporate motion capture and quantification capabilities without prior knowledge of computer vision. As an open-source tool, it also offers users the flexibility to modify or adapt it to specific use cases, including adding new mathematical methods or integrating additional pose estimators.

PyBodyTrack has been successfully validated in four different application scenarios, demonstrating its potential across various fields. Currently, we are working on expanding its capabilities, including motion analysis based on angle variations between body segments, as well as identifying new successful applications to further validate its usefulness and impact. In addition, we plan to incorporate qualitative analysis features into the library, enabling the detection of human behavior and more complex movement patterns.

CRediT authorship contribution statement

Angel Ruiz-Zafra: Writing – original draft, Software, Conceptualization. **Janet Pigueiras-del-Real:** Writing – review & editing, Investigation, Data curation. **Jose Heredia-Jimenez:** Validation, Investigation, Data curation. **Syed Taimoor Hussain Shah:** Writing – review & editing, Validation, Data curation. **Syed Adil Hussain Shah:** Writing – review & editing, Visualization, Data curation. **Lionel C. Gontard:** Writing – review & editing, Supervision, Methodology, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work is part of the "PremAtuRe nEwborn motor and cogNitive impairment: Early diagnosis PARENT project". The PARENT project has received funding from the European Union's Horizon 2020 research and innovation programme under the Maria Sklodowska-Curie Innovative Training Network 2020, Grant Agreement N° 956394.

References

[1] Cronin NJ, Walker J, Tucker CB, Nicholson G, Cooke M, Merlino S, et al. Feasibility of OpenPose markerless motion analysis in a real athletics competition. Front Sport Act Living 2024;5:1298003.

[2] Tacchino C, Impagliazzo M, Maggi E, Bertamino M, Blanchi I, Campone F, et al. Spontaneous movements in the newborns: a tool of quantitative video analysis of preterm babies. Comput Methods Programs Biomed 2021;199:105838.

[3] ROCHA D, AGUILERA A. Freemotionlibs una biblioteca para el análisis del movimiento. uct [online]. 2013, vol. 17, n. 68.

[5] https://parenth2020.com/

[4] Werling K, Bianco NA, Raitor M, Stingel J, Hicks JL, Collins SH, et al. AddBiomechanics: Automating model scaling, inverse kinematics, and inverse dynamics from human motion data through sequential optimization. PLoS One 2023;18(11):e0295152.
[5] Martin H, Hong Y, Wiedemann N, Bucher D, Raubal M. Trackintel: An open-source python library for human mobility analysis. Comput Environ Urban Syst 2023;101:101938.
[6] Nor Adnan NM, Ab Patar MNA, Lee H, Yamamoto S-I, Jong-Young L, Mahmud J. Biomechanical analysis using kinovea for sports application. In: IOP conference series: materials science and engineering, vol. 342, IOP Publishing; 2018, 012097.
[7] Pappalardo L, Simini F, Barlacchi G, Pellungrini R. Scikit-mobility: A python library for the analysis, generation, and risk assessment of mobility data. J Stat Softw 2022;103:1–38.
[8] Graser A. Movingpandas: efficient structures for movement data in python. GIForum 2019;1:54–68.
[9] Boné M, Ros R, Marín J, Martínez J, Álvarez J. Move-human: sistema portátil para captura y análisis tridimensional del movimiento humano en puestos de trabajo basado estéreo-visión y simulación 3D con modelos biomecánicos. In: VIII Congreso Internacional de Ingenieria de Proyectos: Bilbao 6-8 de octubre de 2004. Actas. Asociación Española de Ingeniería de Proyectos (AEIPRO); 2005, p. 131.
[10] Gamma E, Helm R, Johnson R, Vlissides J. Design patterns: elements of reusable object-oriented software. Pearson Deutschland GmbH; 1995.
[11] Liberti L, Lavor C, Maculan N, Mucherino A. Euclidean distance geometry and applications. SIAM Rev 2014;56(1):3–69.
[12] Suwanda R, Syahputra Z, Zamzami EM. Analysis of euclidean distance and manhattan distance in the K-means algorithm for variations number of centroid K. J Phys: Conf Ser 2020;1566(1):012058.
[13] Wang YC, Xing Y, Zhang J. Voronoi treemap in Manhattan distance and Chebyshev distance. Inf Vis 2023;22(3):246–64.
[14] Thant AA, Aye SM, Mandalay M. Euclidean, Manhattan and Minkowski distance methods for clustering algorithms. Int J Sci Res Sci Eng Technol 2020;7(3):553–9.
[15] Todeschini R, Ballabio D, Consonni V, Sahigara F, Filzmoser P. Locally centred Mahalanobis distance: a new distance measure with salient features towards outlier detection. Anal Chim Acta 2013;787:1–9.
[16] Maczák B, Vadai G, Dér A, Szendi I, Gingl Z. Detailed analysis and comparison of different activity metrics. PLoS One 2021;16(12):e0261718.
[17] Dubost V, Beauchet O, Manckoundia P, Herrmann F, Mourey F. Decreased trunk angular displacement during sitting down: an early feature of aging. Phys Ther 2005;85(5):404–12.
[18] Lucas BD, Kanade T. An iterative image registration technique with an application to stereo vision. In: IJCAI'81: 7th international joint conference on artificial intelligence, vol. 2, 1981, p. 674–9.
[19] Farnebäck G. Two-frame motion estimation based on polynomial expansion. In: Image analysis: 13th scandinavian conference, SCIA 2003 Halmstad, Sweden, June 29–July 2, 2003 proceedings 13. Springer; 2003, p. 363–70.
[20] Philosopher0808. Gym workout exercises video dataset. 2023, URL https://www.kaggle.com/datasets/philosopher0808/gym-workoutexercises-video. [Accessed 10 March 2025].
[21] Riccio R. Real-time exercise recognition dataset. 2023, URL https://www.kaggle.com/datasets/riccardoriccio/real-time-exercise-recognition-dataset. [Accessed 10 March 2025].
[22] Mazhar S. Human activity recognition video dataset. 2023, URL https://www.kaggle.com/datasets/sharjeelmazhar/human-activity-recognition-video-dataset. [Accessed 10 March 2025].
[23] Ruiz-Zafra A, Orantes-González E, Noguera M, Benghazi K, Heredia-Jimenez J. A comparative study on the suitability of smartphones and imu for mobile, unsupervised energy expenditure calculi. Sensors 2015;15(8):18270–86.
[24] Kadam AS, Nayyar SA, Kadam SS, Patni BC, Khole MC, Pandit AN, et al. General movement assessment in babies born preterm: Motor optimality score–revised (MOS-R), trajectory, and neurodevelopmental outcomes at 1 year. J Pediatr: X 2023;8:100084.
[25] Pigueiras-del Real J, Gontard LC, Benavente-Fernández I, Lubián-López SP, Gallero-Rebollo E, Ruiz-Zafra A. NRP: A multi-source, heterogeneous, automatic data collection system for infants in neonatal intensive care units. IEEE J Biomed Heal Informatics 2023;28(2):678–89.
[26] Pigueiras-del Real J, Ruiz-Zafra A, Benavente-Fernández I, Lubián-López SP, Shah SAH, Shah STH, et al. NeoVault: empowering neonatal research through a neonate data hub. BMC Pediatr 2024;24(1):787.
[27] Saraswat S, Malathi G. Pose estimation based fall detection system using mediapipe. In: 2024 10th international conference on communication and signal processing. IEEE; 2024, p. 1733–8.
[28] Luangaphirom T, Lueprasert S, Kaewvichit P, et al. Real-time weight training counting and correction using MediaPipe. Adv Comput Intell 2024;4(3). http://dx.doi.org/10.1007/s43674-024-00070-w.
[29] Prechtl HF, Einspieler C, Cioni G, Bos AF, Ferrari F, Sontheimer D. An early marker for neurological deficits after perinatal brain lesions. Lancet 1997;349(9062):1361–3.
[30] Greene BR, O'Donovan A, Romero-Ortuno R, Cogan L, Scanaill CN, Kenny RA. Quantitative falls risk assessment using the timed up and go test. IEEE Trans Biomed Eng 2010;57(12):2918–26.