vailá: Versatile Anarcho Integrated Liberation Ánalysis in Multimodal Toolbox

Paulo Roberto Pereira Santiago[1,2]
paulosantiago@usp.br

Abel Gonçalves Chinaglia[2]
abel.chinaglia@usp.br

Kira Flanagan[6]
kira.flanagan@unf.edu

Bruno L. S. Bedo[3]
bruno.bedo@usp.br

Ligia Yumi Mochida[4,5]
l.mochida@unf.edu

Juan Aceros[4,6]
juan.aceros@unf.edu

Aline Bononi[7]
bononialine@gmail.com

Guilherme Manna Cesar[4,5]
g.cesar@unf.edu

[1] Biomechanics and Motor Control Laboratory, School of Physical Education and Sport, USP, Brazil
[2] Graduate Program in Rehabilitation and Functional Performance, Medical School, USP, Brazil
[3] Technology and Sports Performance Analysis Laboratory, School of Physical Education, USP, Brazil
[4] Laboratory of Applied Biomechanics and Engineering, Brooks College of Health, UNF, USA
[5] Department of Physical Therapy, Brooks College of Health, UNF, USA
[6] College of Computing, Engineering and Construction, UNF, USA
[7] Municipal Pharmacy of Ribeirão Preto, Brazil

Abstract

Human movement analysis is crucial in health and sports biomechanics for understanding physical performance, guiding rehabilitation, and preventing injuries. However, existing tools are often proprietary, expensive, and function as "black boxes", limiting user control and customization. This paper introduces *vailá*—Versatile Anarcho Integrated Liberation Ánalysis in Multimodal Toolbox—an open-source, Python-based platform designed to enhance human movement analysis by integrating data from multiple biomechanical systems. *vailá* supports data from diverse sources, including retroreflective motion capture systems, inertial measurement units (IMUs), markerless video capture technology, electromyography (EMG), force plates, and GPS/GNSS systems, enabling comprehensive analysis of movement patterns. Developed entirely in Python 3.11.9, which offers improved efficiency and long-term support, and featuring a straightforward installation process, *vailá* is accessible to users without extensive programming experience. In this paper, we also present several workflow examples that demonstrate how *vailá* allows the rapid processing of large batches of data, independent of the type of collection method. This flexibility is especially valuable in research scenarios where unexpected data collection challenges arise, ensuring no valuable data point is lost. We demonstrate the application of *vailá* in analyzing sit-to-stand movements in pediatric disability, showcasing its capability to provide deeper insights even with unexpected movement patterns. By fostering a collaborative and open environment, *vailá* encourages users to innovate, customize, and freely explore their analysis needs, potentially contributing to the advancement of rehabilitation strategies and performance optimization. For more details in GitHub repository: https://github.com/vaila-multimodaltoolbox/vaila and the online documentation: https://vaila.readthedocs.io.

Keywords biomechanics, human movement analysis, open-source software, Python, multimodal data integration, batch processing, workflow

1 Introduction

Human movement analysis is essential for the understanding of physical performance, rehabilitation, and injury prevention within health and sports biomechanics [Ricamato and Hidler, 2005, Roberts et al., 2017, Cesar et al., 2016, Bedo et al., 2021, Cesar et al., 2022]. Accurate analysis of human movement patterns can lead to improved rehabilitation strategies, enhanced athletic performance, and a deeper understanding of musculoskeletal function. However, existing tools for movement analysis often present significant limitations. For instance, the Biomechanical Tool Kit (BTK) Mokka, once a popular biomechanical analysis choice, has not been updated or developed for over nine years [Barre and Armand, 2014]. Many contemporary solutions are proprietary, expensive, and function as "black boxes," restricting user control and customization and hindering the advancement of research through community contributions.

Several prominent tools in the field, such as OpenSim, are widely used for musculoskeletal modeling and simulation [Seth et al., 2018]. OpenSim allows for the development of custom toolboxes, which has been instrumental in biomechanics research. For example, custom musculoskeletal models have been developed to estimate tibiofemoral contact forces during tasks involving high knee and hip flexion [Bedo et al., 2020a]. Similarly, BOPS, a MATLAB toolbox, facilitates batch processing of musculoskeletal data in OpenSim [Bedo et al., 2020b]. Despite their utility, these tools rely heavily on programming languages like C++ and MATLAB, the latter requiring a paid license. This reliance can pose barriers for researchers without access to such software, limiting the accessibility and dissemination of advanced biomechanical analysis methods.

In contrast, Python is a high-level, open-source programming language known for its simplicity, extensive libraries, and strong community support. Python offers enhanced memory safety and portability across platforms such as Linux, macOS, and Windows [van Rossum and the Python Development Team, 2022]. Leveraging Python's capabilities, we have developed *vailá—Versatile Anarcho Integrated Liberation Ánalysis in Multimodal Toolbox*—an open-source, customizable platform designed to enhance human movement analysis by integrating data from multiple biomechanical systems. *vailá* supports data from diverse sources, including retroreflective motion capture systems, inertial measurement units (IMUs), markerless video capture technology, electromyography (EMG), force plates, and GPS/GNSS systems, enabling comprehensive and precise analysis of movement patterns.

The motivation for developing *vailá* arose from the need for a flexible, open-source tool capable of facilitating batch processing of large datasets in biomechanical research environments. During our collaboration with the University of North Florida's Laboratory of Applied Biomechanics and Engineering (LABE) under the CAPES-PAME program, we identified data processing workflows as a bottleneck in research activities. The increasing use of markerless solutions for participants with disabilities or comprehension difficulties highlighted the necessity for a tool that could simplify batch processing, integrate multimodal data, and provide researchers with the flexibility to modify the software according to their needs.

The main objective of this paper is to present and outline the development of *vailá*, which was initially designed to serve as a multimodal processing tool in biomechanical data collection for children with cerebral palsy. *vailá* acts as a facilitator, inviting individuals without extensive expertise in programming or biomechanics to contribute and enhance the software. By providing an open, modular, and inclusive platform for data analysis, *vailá* empowers the biomechanics research community to innovate, customize, and freely explore their analysis needs, thereby breaking down barriers imposed by traditional software.

2 Related Work

Several existing tools and frameworks have significantly contributed to the field of biomechanics and multimodal data analysis. OpenSim is one of the most established platforms for musculoskeletal modeling and simulation [Seth et al., 2018]. It allows users to extend its functionalities through custom toolboxes, which has been instrumental in advancing biomechanics research. For instance, custom musculoskeletal models have been developed using OpenSim to estimate tibiofemoral contact forces during activities involving high knee and hip flexion [Bedo et al., 2020a]. BOPS (Batch OpenSim Processing Scripts), a MATLAB toolbox, was also created to facilitate batch processing of musculoskeletal data [Bedo et al., 2020b]. However, the reliance on MATLAB, a proprietary platform, introduces financial and accessibility constraints, potentially limiting the tool's adoption in resource-limited settings.

Other solutions, such as BTK Mokka, provided a free and open-source alternative for biomechanics research [Barre and Armand, 2014]. Despite its initial utility, BTK Mokka has seen no active development for several years, leading to gaps in functionality and compatibility with modern technologies. Its limitations include lacking support for recent hardware advancements and data formats, which hinders its adoption in contemporary research environments.

Markerless motion capture has gained traction due to its potential to streamline data collection, especially for participants who may not tolerate traditional markers [Smith et al., 2023]. Platforms like MediaPipe offer pipelines for human pose estimation using machine learning techniques [Lugaresi et al., 2019]. While MediaPipe is a powerful tool for general pose estimation, it is not specifically tailored for biomechanics research, and its integration with biomechanical analysis pipelines remains underdeveloped.

In the realm of open-source tools, DeepLabCut [Matthis et al., 2022] has made significant strides by providing a user-friendly platform for markerless pose estimation across species and behaviors. However, it primarily focuses on pose estimation and does not offer comprehensive biomechanical analysis or multimodal data integration.

Similarly, free tools like Tracker [Wee, 2013], Kinovea, and Dvideow [Figueroa et al., 2003] have been instrumental in advancing biomechanical analyses by providing accessible platforms for motion tracking and analysis. While these tools have significantly contributed to the field, they cannot often handle multiple data modalities or require extensive manual processing, which can be time-consuming and less efficient for large datasets.

In contrast, *vailá* offers a flexible solution that integrates multiple data sources, such as motion capture, IMU, EMG, and supports markerless data collection. Developed in Python 3.11.9—which benefits from CPython acceleration—*vailá* leverages the extensive libraries and strong community support inherent to Python. This choice of programming language enhances performance and makes the tool more accessible to researchers, as Python is widely taught and used across scientific disciplines.

By providing an open-source, Python-based platform, *vailá* lowers the entry barrier for biomechanics researchers and facilitates community-driven development. This approach allows for rapid integration of new data sources and technologies, ensuring the toolbox stays relevant in a rapidly evolving research landscape. The transparency of the codebase encourages users to engage with and modify the software, fostering an environment where imagination and innovation drive the advancement of biomechanical analysis.

3 Methods

3.1 Software Engineering Principles in *vailá*

The design and development of the *vailá* toolbox are rooted in well-established software engineering principles, particularly those outlined by Frederick P. Brooks in *The Mythical Man-Month* [Brooks Jr, 1995, Brooks, 1995]. *vailá* was conceived as a modular, open-source, and transparent system to meet the needs of a biomechanics and bioengineering research lab in its early stages of operation.

In a research environment, especially in biomechanics, planned methods and equipment cannot always be applied as intended due to factors such as participants' disabilities or limited comprehension. These circumstances often require adjustments to data collection methods. To accommodate this variability, *vailá* was designed with flexibility in mind, allowing for easy modifications to the software.

One of the primary goals of *vailá* is to ensure that the system remains open, modular, and extensible, enabling users and contributors to modify or add new features easily. Every Python script in *vailá* includes mechanisms for transparency, where the code prints essential metadata—such as the name and path of the script being executed—providing users with a clear understanding of what is running and where. Additionally, whenever possible, *vailá* ensures that all important calculations and relevant procedures are displayed in the terminal, giving users full visibility into the ongoing processes.

For example, each script in *vailá* uses the following print statements to provide transparency about its execution:

```
import os
from rich import print


# Print the directory and name of the script being executed
print(f"Running script: {os.path.basename(__file__)}")
print(f"Script directory: {os.path.dirname(os.path.abspath(__file__))}")
```

This transparency enables users to track the execution flow, understand key calculations, debug more efficiently, and better comprehend the modular structure of the software. Using the `rich` library enhances terminal output, making it easier to visualize the steps in the analysis process.

By adhering to these principles, *vailá* allows researchers to adapt the software to their specific needs while maintaining clarity about what is being executed. This simplifies debugging and the development of new features and ensures that users are informed about the key operations and calculations being performed throughout the analysis.

3.2 Modular Architecture of *vailá* GUI and CLI

The core design philosophy of *vailá* is its modularity. Each major toolbox functionality is encapsulated in self-contained modules, allowing individual components to be developed, tested, and deployed independently. This approach follows the principle of separation of concerns, where each module is responsible for a specific task, such as file management, data analysis, or visualization.

The graphical user interface (GUI) of *vailá* is divided into three main frames: File Manager (Frame A), Multimodal Analysis (Frame B), and Available Tools (Frame C). Each frame is organized into rows and columns, where each button represents a specific functionality related to file handling, multimodal data analysis, or available tools for further processing and visualization. Below is the detailed explanation of each frame:

- **Frame A: File Manager** – Responsible for file operations, this section provides buttons for managing files and directories. The rows and columns are structured as follows:
    - **A_r1_c1 to A_r1_c9:**
        * A_r1_c1 – Rename Files
        * A_r1_c2 – Import Files
        * A_r1_c3 – Export Files
        * A_r1_c4 – Copy Files
        * A_r1_c5 – Move Files
        * A_r1_c6 – Remove Files
        * A_r1_c7 – Tree View
        * A_r1_c8 – Find Files
        * A_r1_c9 – Transfer Files
- **Frame B: Multimodal Analysis** – This frame includes tools for analyzing various types of biomechanical data such as IMU, motion capture (MoCap), and markerless video analysis. The rows and columns are organized as follows:
    - **B1_r1_c1 to B1_r1_c5:**
        * B1_r1_c1 – IMU Analysis
        * B1_r1_c2 – MoCap Cluster Analysis
        * B1_r1_c3 – MoCap Full Body Analysis
        * B1_r1_c4 – Markerless 2D Analysis
        * B1_r1_c5 – Markerless 3D Analysis
    - **B2_r2_c1 to B2_r2_c5:**
        * B2_r2_c1 – Vector Coding
        * B2_r2_c2 – EMG Analysis
        * B2_r2_c3 – Force Plate Analysis
        * B2_r2_c4 – GNSS/GPS Data
        * B2_r2_c5 – MEG/EEG Data
    - **B3_r3_c1 to B3_r3_c5:**
        * B3_r3_c1 – HR/ECG Data
        * B3_r3_c2 to B3_r3_c5 – Placeholder for other custom tools (*vailá*)
- **Frame C: Available Tools** – This frame includes tools for data conversion, video and image processing, and visualization. It is organized into three sections (Data Files, Video and Image, and Visualization) and structured as follows:
    - **C_A: Data Files**
        * C_A_r1_c1 – Edit CSV
        * C_A_r1_c2 – C3D <–> CSV Conversion
        * C_A_r1_c3 – Placeholder for *vailá* tools
        * C_A_r2_c1 to C_A_r3_c3 – DLT Methods (2D and 3D) and additional placeholder tools (*vailá*)
    - **C_B: Video and Image**
        * C_B_r1_c1 – Convert Video to PNG
        * C_B_r1_c2 – Cut Videos
        * C_B_r1_c3 – Draw Box
        * C_B_r2_c1 – Compress Videos (H264)

* C_B_r2_c2 – Compress Videos (H265)
* C_B_r2_c3 – Make Sync File
* C_B_r3_c1 – Get Pixel Coordinates
* C_B_r3_c2 – Metadata Information
* C_B_r3_c3 – Merge Videos

– **C_C: Visualization**

* C_C_r1_c1 – Show C3D
* C_C_r1_c2 – Show CSV
* C_C_r2_c1 – Plot 2D
* C_C_r2_c2 – Plot 3D
* C_C_r3_c1 to C_C_r4_c3 – Placeholder for additional *vailá* visualization tools

Each frame (A, B, C) is designed to handle a distinct part of the data processing and analysis workflow, allowing the user to move through data management, multimodal analysis, and tools for visualization or file conversion seamlessly. The modular layout also enables future expansion and customization by adding new tools or features in placeholders indicated as *vailá*. This structure provides flexibility and ease of use, tailored for biomechanical data processing needs.

Each area is further divided into individual components or scripts, which can be customized or expanded upon. The diagram emphasizes the modularity of the *vailá* toolbox. The process starts with file management, progresses through multimodal data processing and analysis, and concludes with generating results, which can be visualized and exported in various formats.

These capabilities enable researchers to explore new dimensions in biomechanical studies by correlating data from different sources. The synchronization functionality was effective across different systems, providing accurate temporal alignment with minimal preprocessing (Figure 1).
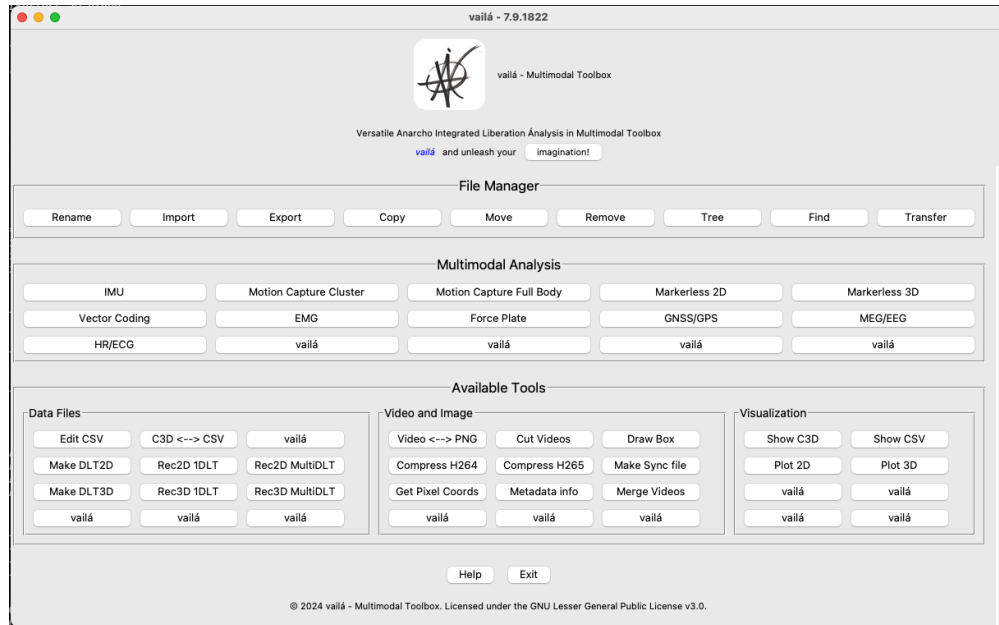

Figure 1: Graphical User Interface (GUI) of the *vailá* toolbox, built using the *Tkinter* library in Python. The interface is organized into three hierarchical levels. The first level consists of three main frames: **File Manager** (Frame A) for file operations, **Multimodal Analysis** (Frame B) for handling biomechanical data such as MoCap, IMU, and GNSS, and **Available Tools** (Frame C) for additional functionalities like visualization and file conversion. The second and third levels within each frame represent the rows and columns, where specific tools and operations are accessible. This modular structure allows for seamless data processing and analysis while providing placeholders labeled "vailá" for future tool expansions and customizations, ensuring flexibility and ease of use.

3.3   Transparent Code Execution with Print Statements

Following the principle of making code execution transparent, every script in *vailá* includes print statements that display the script's name and path. This approach ensures that users have complete visibility into the behind-the-scenes processes, contributing to a more intuitive understanding of the analysis workflow.

The print statements, integrated with the `rich` library for enhanced terminal output, provide the following benefits:

- **Track Execution Flow:** By showing the execution flow directly in the terminal, users can follow which scripts and functions are being called during their analysis.
- **Efficient Debugging:** When an issue arises, the printed file paths and script names help users pinpoint where specific operations are occurring, streamlining the debugging process.
- **Understand Modularity:** As *vailá* is designed with a modular architecture, printing the execution path allows users to see how different components interact without diving deep into the codebase.

This transparency aligns with the ethos of open-source software, empowering users to understand and control their analysis environment. The combination of print statements and structured logging ensures that users of *vailá* can monitor the progress of their analyses in real-time, which is particularly important for large-scale biomechanical studies involving batch processing.

By implementing such features, *vailá* not only simplifies the analysis process but also reinforces the concept of open-source collaboration, where code transparency and traceability are fundamental.

3.4   Terminal and Debugging Enhancements

Several tools have been integrated into the environment to streamline the development process and debugging of new modules within the *vailá* toolbox. One key aspect is the installation of Xonsh and the use of the ipdb debugger, combined with rich terminal output.

Xonsh is a Python-powered shell designed to enhance the traditional terminal experience. It allows seamless mixing of Python commands and shell primitives, combining the functionalities of Bash and Python in one environment. This provides a significant advantage for developers and users who need powerful scripting and shell interaction.

Xonsh is a cross-platform shell that works on Linux, macOS, and Windows, making it a suitable choice for the *vailá* toolbox, which targets users from different operating systems. The command-line interface can be accessed by invoking Xonsh in the terminal or by clicking the "imagination!" button in the *vailá* graphical user interface (GUI), which opens the Xonsh terminal within the active Conda environment.

3.4.1   Python Debugging with `ipdb`

To further support the debugging process, *vailá* incorporates the `ipdb` library, which provides an enhanced version of Python's standard debugger (pdb). `ipdb` includes features such as tab completion, syntax highlighting, and advanced tracebacks, improving developer productivity when tracing errors or analyzing the flow of the program. This debugging tool is available within any toolbox module and can be triggered during code execution for real-time variables and control flow inspection.

```
import ipdb
ipdb.set_trace()  # This sets a breakpoint in the code
```

3.4.2   Terminal Output with `rich`

Additionally, the terminal output in *vailá* is enhanced by the `rich` library. `rich` improves the visual output of the terminal by enabling syntax highlighting, pretty printing, and better readability of complex data structures. Every Python script executed within *vailá* prints its status using `rich`, making the terminal output clearer and more user-friendly.

```
from rich import print
print("This is an enhanced output with [bold magenta]Rich[/bold magenta]!")
```

The Figure 2 demonstrates the enhanced terminal setup with Xonsh, `ipdb`, and `rich`, showcasing how Python commands, shell operations, and debugging coexist in the same environment.

The combination of Xonsh, `ipdb`, and `rich` makes the development and troubleshooting processes in *vailá* more efficient while also providing a flexible interface for users to interact with both the shell and Python code.
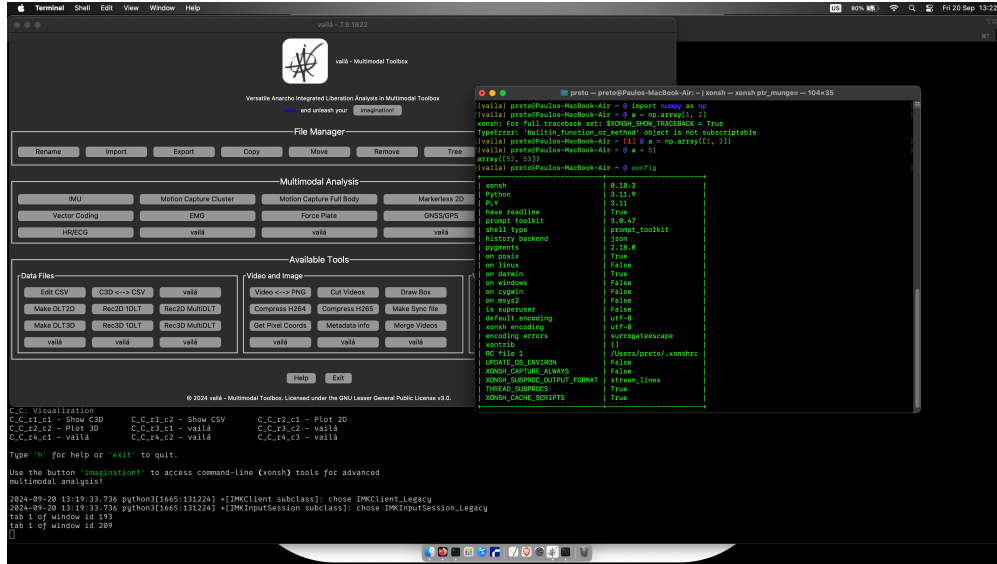
Figure 2: A terminal session running in the **Xonsh** shell, displaying enhanced output with `rich`, and real-time debugging with `ipdb`. This setup allows users to mix Python 3.11.9 and shell commands while debugging and viewing formatted terminal output.

As an open-source project, *vailá* is licensed under the GNU Lesser General Public License version 3.0 [GPL, 2007]. This license allows users to freely use, modify, and distribute the software if modifications to the core library are made available under the same license. This ensures that the community benefits from continuous improvements to the toolbox, fostering a collaborative development environment.

3.5   Code Header and Documentation

To maintain code organization and facilitate cooperation between developers, all Python scripts in the *vailá* toolbox include a standardized header. This header provides essential information, such as the script's author, version, description, and instructions for usage. Additionally, each line of the code is thoroughly commented for clarity.

Below is an example of a script header, formatted using the `verbatim` environment to showcase how documentation is included in each Python script:

```
"""
Script: markerless_2D_analysis.py
Author: Prof. Dr. Paulo Santiago
Version: 0.2.0
Last Updated: September 28, 2024

Description:
    This script performs batch processing of videos for 2D pose estimation using
    MediaPipe's Pose model. It processes videos from a specified input directory,
    overlays pose landmarks on each video frame, and exports both normalized and
    pixel-based landmark coordinates to CSV files.

    The user can configure key MediaPipe parameters via a graphical interface,
    including detection confidence, tracking confidence, model complexity, and
    whether to enable segmentation and smooth segmentation. The default settings
    prioritize the highest detection accuracy and tracking precision, which may
    increase computational cost.

New Features:
    - Default values for MediaPipe parameters are set to maximize detection and
      tracking accuracy:
        - `min_detection_confidence=1.0`
        - `min_tracking_confidence=1.0`
        - `model_complexity=2` (maximum complexity)
        - `enable_segmentation=True` (segmentation activated)
        - `smooth_segmentation=True` (smooth segmentation enabled)
    - User input dialog allows fine-tuning these values if desired.

Usage:
```

```
    - Run the script to open a graphical interface for selecting the input directory
      containing video files (.mp4, .avi, .mov), the output directory, and for
      specifying the MediaPipe configuration parameters.
    - The script processes each video, generating an output video with overlaid pose
      landmarks, and CSV files containing both normalized and pixel-based landmark
      coordinates.

How to Execute:
    1. Ensure you have all dependencies installed:
        - Install OpenCV: 'pip install opencv-python'
        - Install MediaPipe: 'pip install mediapipe'
        - Tkinter is usually bundled with Python installations.
    2. Open a terminal and navigate to the directory where 'markerless_2D_analysis.py' is located.
    3. Run the script using Python:

        python markerless_2D_analysis.py

    4. Follow the graphical interface prompts:
        - Select the input directory with videos (.mp4, .avi, .mov).
        - Select the base output directory for processed videos and CSVs.
        - Configure the MediaPipe parameters (or leave them as default for maximum accuracy).
    5. The script will process the videos and save the outputs in the specified output directory.

Requirements:
    - Python 3.11.9
    - OpenCV ('pip install opencv-python')
    - MediaPipe ('pip install mediapipe')
    - Tkinter (usually included with Python installations)
    - Pillow (if using image manipulation: 'pip install Pillow')

Output:
    The following files are generated for each processed video:
    1. Processed Video ('*_mp.mp4'):
        The video with the 2D pose landmarks overlaid on the original frames.
    2. Normalized Landmark CSV ('*_mp_norm.csv'):
        A CSV file containing the landmark coordinates normalized to a scale between 0 and 1
        for each frame. These coordinates represent the relative positions of landmarks in the video.
    3. Pixel Landmark CSV ('*_mp_pixel.csv'):
        A CSV file containing the landmark coordinates in pixel format. The x and y coordinates
        are scaled to the video's resolution, representing the exact pixel positions of the landmarks.
    4. Log File ('log_info.txt'):
        A log file containing video metadata and processing information, such as resolution, frame rate,
        total number of frames, codec used, and the MediaPipe Pose configuration used in the processing.

Example:
    1. Select a folder with videos in .mp4 format.
    2. Choose the output directory for saving processed videos and CSVs.
    3. Enter the desired values for detection confidence (e.g., 0.5), tracking confidence (e.g., 0.5),
        model complexity (0, 1, or 2), and segmentation options.
    4. The processed files will be saved with landmarks overlaid and CSVs in the chosen
        output directory.

License:
    This program is free software: you can redistribute it and/or modify it under the terms of
    the GNU General Public License as published by the Free Software Foundation, either version 3
    of the License, or (at your option) any later version.

    This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY;
    without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
    See the GNU General Public License for more details.

    You should have received a copy of the GNU GPLv3 (General Public License Version 3) along with this program.
    If not, see <https://www.gnu.org/licenses/>.
"""
```

This standardized documentation style ensures that each script's purpose, usage, and dependencies are clearly defined, promoting consistency across the *vailá* toolbox. Moreover, including extensive comments within the codebase aids in maintaining the transparency and modularity of the project, enabling both novice and experienced developers to contribute effectively.

3.6   Batch Processing and Automation

One of the fundamental features of *vailá* is its ability to run batch processes across multiple files, significantly improving efficiency in data analysis workflows. Instead of manually selecting and processing files one by one, *vailá* automatically applies the same operations to all files in a specified directory, reducing the time needed for large-scale biomechanical studies.

This functionality is particularly advantageous for researchers in biomechanics and bioengineering laboratories, where data collection occurs frequently, often yielding large volumes of files to be processed weekly. Handling these datasets manually can be time-consuming and prone to error. Using batch processing, *vailá* simplifies this process, ensuring that each file is processed similarly, with minimal user intervention.

When files are organized appropriately in the **File Manager** (Frame A), all processing tasks can be executed efficiently in "batch" mode. This method allows researchers to automate repetitive tasks, freeing up time for higher-level analysis and decision-making. The structure of *vailá* ensures that all Python scripts are designed to handle batch processing, operating in a `for` loop to process files of the same extension and internal organization.

For example, if a lab collects force platform data in CSV format or video files for motion capture analysis, *vailá* can be configured to automatically process each of these files in sequence, applying the same filters, metrics calculations, or visualizations across the entire dataset. This accelerates the workflow and minimizes the risk of human error that can arise when handling large datasets manually.

The batch processing capabilities of *vailá* are an essential feature for labs that deal with high volumes of data, enabling researchers to process their data in a fraction of the time it would take using manual methods. By automating these tasks, the toolbox enhances productivity, allowing researchers to focus on analyzing their results rather than managing file operations. The modular architecture ensures that batch operations are accessible across all tools, providing a streamlined and efficient workflow for processing biomechanical data.

4 Results

This section presents the performance results of the *vailá* toolbox when processing a complete biomechanical data collection workflow. All examples provided here can be found in the `tests/` directory of the *vailá* repository, which includes sample data and code to replicate the results shown. This workflow encompasses multiple types of biomechanical data (e.g., C3D, EMG, IMU, markerless 2D, and motion capture), demonstrating the flexibility and robustness of *vailá*.

4.1 Codebase Overview

To illustrate the current state of the *vailá* toolbox, we present an overview of its codebase. The project is predominantly written in Python 3.11.9, with contributions from Shell and PowerShell scripts, reflecting the diversity of tools used for automation and data processing (Table 1).

Table 1: Current codebase statistics of *vailá* as shown on GitHub.

Language	Percentage	Lines of Code
Python	94.8%	15,630
Shell	2.7%	445
PowerShell	2.5%	256

Table 1, the *vailá* project is composed mainly of Python code (94.8%), with Shell (2.7%) and PowerShell (2.5

4.2 Tracking Marker Points in Videos: The *Get Pixel Coordinate* Tool

One of the critical tools within *vailá* is the *Get Pixel Coordinate* tool, implemented in the script `getpixelvideo.py`. This tool allows users to manually mark and save pixel coordinates in video frames, featuring zoom functionality for precise annotations. It is particularly useful when working with markerless video data, where manual annotation or verification of automatically detected key points is necessary.

The tool supports various video formats, such as `.mp4`, `.avi`, `.mov`, and `.mkv`, and allows users to load pre-marked points from CSV files. It controls frame navigation, zooming, and marking or adjusting points. Marked coordinates are saved in CSV format, facilitating subsequent biomechanical analysis within the *vailá* workflow.

An example of the tool's interface is shown in Figure 3. The interface is designed for efficient annotation, enabling users to interactively navigate through frames and adjust accurately to keypoints. This flexibility enhances the usability of the *vailá* toolbox for analyzing biomechanical data from markerless systems.

Figure 3: Screenshot of the *Get Pixel Coordinate* tool within *vailá*. The interface allows users to navigate video frames, zoom in for precise point selection, and mark or adjust keypoints. This tool facilitates manual annotation and verification of keypoints in video data, enhancing the flexibility of the *vailá* toolbox.

The *Get Pixel Coordinate* tool also integrates with the *Edit CSV* tool within *vailá*, which provides functionality for processing and converting keypoints or markers from various CSV formats. This integration enables users to seamlessly incorporate manually annotated data or adjust existing datasets for compatibility with other analysis workflows.

Incorporating tools like `getpixelvideo.py`, *vailá* enhances the ability to process and analyze markerless video data, which is increasingly important in biomechanical studies involving participants who may not tolerate traditional markers or in field-based research where markerless systems are more practical. Using these workflows, the ability to quickly process large batches of video data allows researchers to adapt to unexpected research opportunities without losing valuable data collections.

Furthermore, the inclusion of such tools demonstrates the modularity and extensibility of *vailá*. Users can customize and expand the toolbox by adding scripts tailored to their specific needs, promoting a collaborative environment where new functionalities can be shared and integrated into the main framework. This flexibility is crucial for accommodating the diverse requirements of biomechanics research, where data types and analysis methods can vary widely between studies.

Another significant advantage of *vailá* is its cross-platform compatibility, particularly in providing tools for visualizing C3D files on macOS and Linux. Currently, most free or open-source tools for C3D visualization, such as Mokka and Visual3D Reader, are limited to Windows. While Mokka is available for macOS, it only runs on older OSX versions, leaving many users without accessible options.

To address this, *vailá* includes powerful 3D visualization tools, allowing users to animate and interact with C3D and CSV files using Python libraries such as Matplotlib and Plotly. These visualizations can be displayed in the system's default browser, offering features like marker selection and forward/backward playback controls. This functionality enhances the user experience by providing interactive, real-time visualizations on any operating system.

The visualization tools are easily accessible under the "Available Tools > Visualization" section within the *vailá* interface, enabling users to generate 3D plots and explore biomechanical data without the need for specialized, platform-specific software.

An example of a 3D animated C3D data visualization using Plotly within *vailá* is shown in Figure 4.
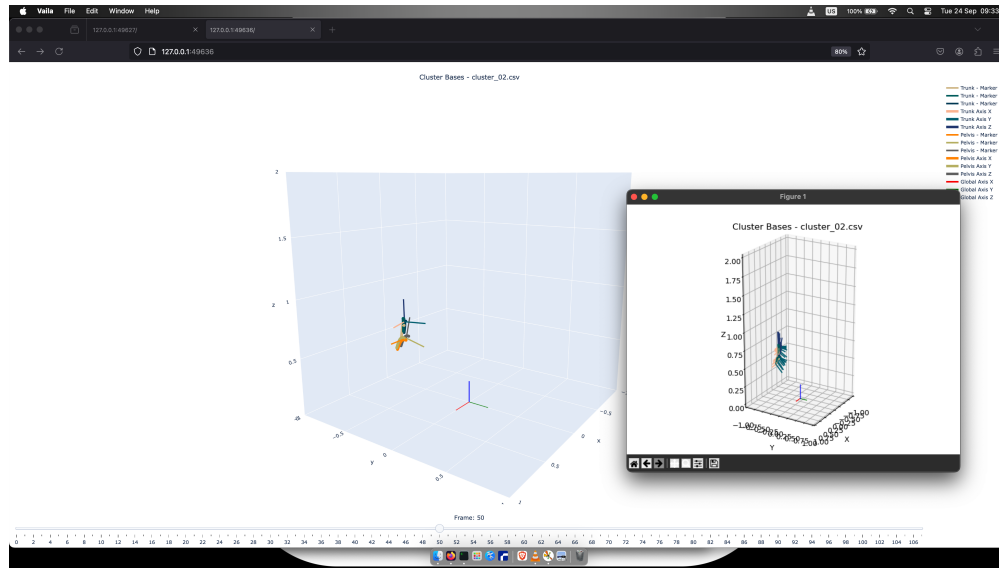

Figure 4: Example of a 3D animated visualization of C3D data using Plotly in *vailá*. The interface allows users to interact with markers, play the animation forward or backward, and adjust the display animation. These tools are located in the "Available Tools > Visualization" section.

4.3   Markerless 2D Workflow in *vailá*

The *vailá* toolbox provides an optimized and efficient workflow for processing markerless 2D video data, significantly improving performance compared to traditional methods. The workflow is represented in Figure 5, which shows the sequence of steps used in the processing pipeline:

Figure 5: Markerless 2D processing pipeline in *vailá*, with each step briefly describing the operation performed.

The workflow starts by extracting metadata from the video using the *Metadata info* tool. This step identifies any potential video compression or codec issues before further processing.

The *Video <–> PNG* tool is applied to extract all frames from the original video and recompress them into a new h264 video format, which is optimal for analysis. Next, the *Make Sync File* tool generates synchronization files to align the frames of two videos for subsequent analysis.

The *Cut Videos* step trims the synchronized videos using the sync file, and the *Draw Box* tool masks irrelevant regions in the video, focusing on the subject of interest. Finally, the *Markerless 2D* Multimodal, powered by MediaPipe [Lugaresi et al., 2019], tracks and overlays 2D pose landmarks on the processed video.

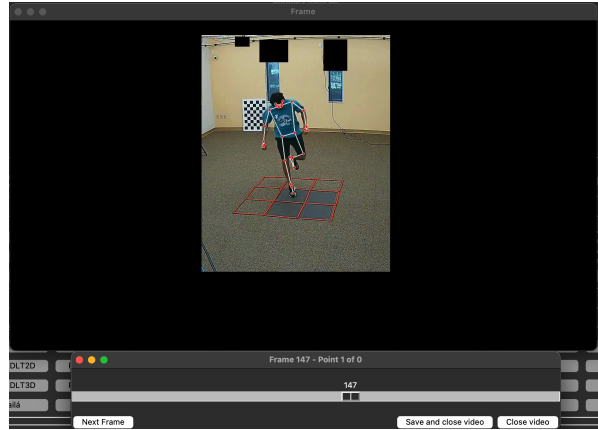This workflow generates the following output files for each processed video:

- **Processed Video** (`*_mp.mp4`): The video with 2D pose landmarks overlaid on the original frames.
- **Normalized Landmark CSV** (`*_mp_norm.csv`): A CSV file containing normalized landmark coordinates.
- **Pixel Landmark CSV** (`*_mp_pixel.csv`): A CSV file containing pixel coordinates for the landmarks, scaled to the video's resolution.
- **Log File** (`log_info.txt`): A log file detailing video metadata, such as resolution, frame rate, total frames, and codec used.

The Figure 6 presents a mosaic of four frames from the markerless workflow to visually demonstrate the results. The top-left (Figure 6a) shows the raw input video from Camera 1, and the top-right (Figure 6b) displays the processed

video with the 2D pose landmarks overlaid. Similarly, the bottom-left (Figure 6c) shows the raw video from Camera 2, and the bottom-right (Figure 6d) displays the final processed result with landmarks.
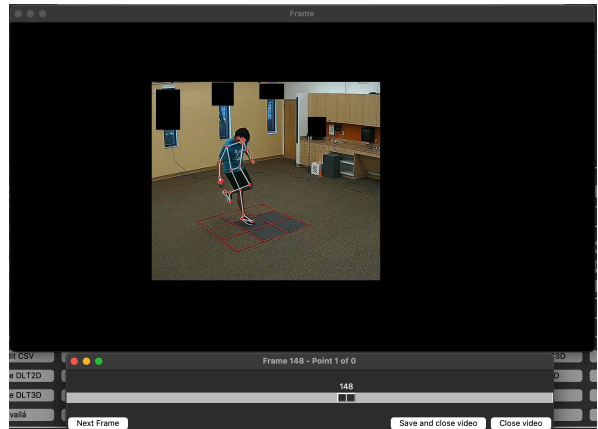

(a) (a) Raw video camera 1

(b) (b) *vailá* processed video of camera 1 with Landmarks

(c) (c) Raw video camera 2

(d) (d) *vailá* processed video of camera 2 with Landmarks

Figure 6: Markerless 2D analysis in *vailá*. Top: raw video frames. Bottom: processed videos with 2D pose landmarks overlaid.

In addition to the markerless workflow, *vailá* also includes tools for creating 2D and 3D calibrations using the Direct Linear Transformation (DLT) method, both for static and dynamic (moving) cameras. These scripts are available under *Available Tools* > *Data Files* > *Make DLT 2D* and *Make DLT 3D*. Moreover, 2D and 3D reconstructions based on single or multiple DLT calibrations for each frame are supported, allowing dynamic calibrations. These tools can be used with screen coordinate data, either from the Markerless 2D system or any other tracking system.

4.4 Motion Capture and IMU Cluster Workflows in *vailá*

Using a cluster-based approach, the *vailá* toolbox offers robust workflows for analyzing motion capture and IMU (Inertial Measurement Unit) data. These techniques allow the processing of 3D coordinates and sensor data to compute joint rotations and derive Euler angles. The two modalities—traditional motion capture clusters and IMU-based clusters—share a similar analysis pipeline but differ in the data sources used.
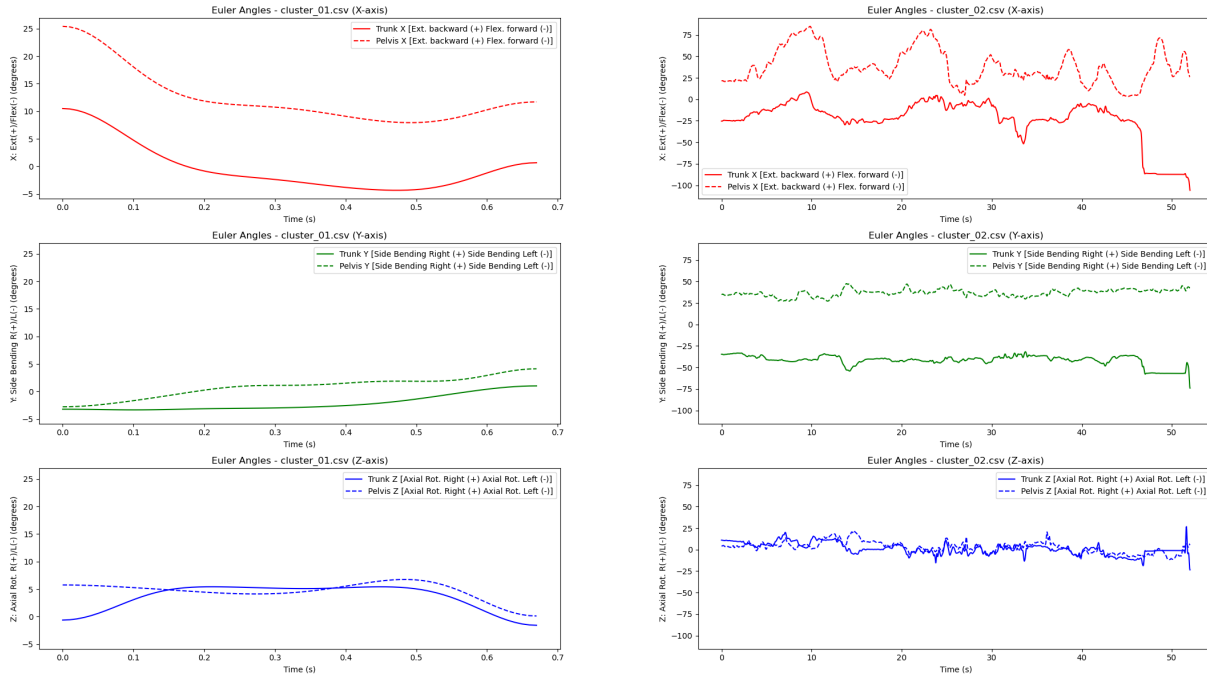
4.4.1 Motion Capture Cluster Workflow

In the motion capture cluster workflow, 3D coordinates of reflective markers attached to specific clusters (e.g., trunk and pelvis) are processed to compute orthonormal bases and derive Euler angles that describe joint rotations. The workflow organizes the outputs into a structured directory, including folders for figures and processed data files.

The output includes:

- **Cluster Data CSV** (`*_cluster.csv`): This file contains the computed Euler angles for each cluster over time.
- **Figures** (`*_figure.png`): Visual representations of the orthonormal bases and Euler angles for each cluster.

The results of this workflow are demonstrated in Figure 7, showing the Euler angles for both the trunk (Cluster 1) and pelvis (Cluster 2) during a Sit To Stand Test.

(a) Euler Angles for Trunk (Cluster 1) - Typically Developing Child

(b) Euler Angles for Pelvis (Cluster 2) - Child with Cerebral Palsy

Figure 7: Euler angles over time for the trunk and pelvis clusters during a Sit To Stand Test in *vailá*. Clusters were positioned on the trunk and pelvis. (a) Data from a typically developing child. (b) Data from a child with cerebral palsy. The use of clusters is beneficial when skin issues or altered sensitivity may lead to children resisting the placement of kinematic markers.

4.4.2 IMU Cluster Workflow

The IMU cluster workflow in *vailá* offers a similar analysis structure but utilizes data from Inertial Measurement Units (IMUs) instead of reflective markers. IMUs are typically placed on the same body regions as the clusters (e.g., trunk and pelvis) and provide accelerometer and gyroscope data along the X, Y, and Z axes. Euler angles are derived from this data to describe the same joint rotations computed in the motion capture workflow.

The output of the IMU workflow is stored in a CSV file with the following structure:

```
Time,Gyro_X,Gyro_Y,Gyro_Z,Acc_X,Acc_Y,Acc_Z,Euler_X,Euler_Y,Euler_Z, Tilt_X,Tilt_Y,Tilt_Z,Quat_W,Quat_X,Quat_Y,Quat_Z
0.000,52.874,17.166,0.961,-0.066,0.910,0.858,0.235,0.024,0.000,-3.038,46.609,46.770,1.000,0.002,0.000,0.000
...
10.000,52.872,17.166,0.962,-0.066,0.910,0.858,0.468,0.047,0.000,-3.038,46.608,46.770,0.999,0.004,0.000,-0.000
```

This file contains timestamps and a range of computed values, including:

- **Gyroscope Data** (X, Y, Z): Captures the angular velocity of the IMU in three axes.
- **Accelerometer Data** (X, Y, Z): Measures the linear acceleration of each IMU in three axes.
- **Euler Angles** (X, Y, Z): Derived angles representing the rotation of the body segments.
- **Tilt Angles** (X, Y, Z): Calculations of the tilt of each segment.

- **Quaternion Data** (W, X, Y, Z): Provides an alternative representation of the orientation of each segment.

The key difference between the motion capture cluster and IMU workflows lies in the data sources: while motion capture uses physical markers and camera systems, the IMU workflow leverages wearable sensors to gather acceleration and angular velocity data. This allows for more flexible motion tracking, particularly in environments where camera systems may not be feasible or when working with subjects who may resist wearing reflective markers.

The following figure compares Euler angles derived from a typically developing child and a child with cerebral palsy, showcasing how both modalities can assess joint movements across different populations and conditions.

4.5 EMG Workflow in *vailá*

The processing workflow for EMG data in *vailá* begins with selecting the directory containing CSV files with EMG data, defining the sampling rate, and identifying the analysis windows. The key outputs include filtered EMG signals, rectified signals, RMS values, and frequency domain analysis, which are visualized in Figure 8.

Select path
of files CSV
Choose directory
with EMG data
→
Define parameters
Sampling rate, etc.
→
Check analysis
window
Specify start
and end
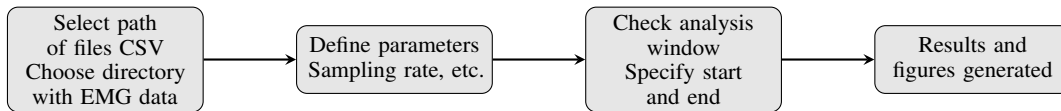→
Results and
figures generated

Figure 8: EMG signal processing workflow in *vailá*.

To demonstrate the results of EMG analysis, Figure 9 presents a mosaic of key outputs for the file `emg_01.csv`. These include the bandpass filtered EMG signal, which is used to remove noise; the full-wave rectified EMG signal for envelope detection; the Root Mean Square (RMS) values, which reflect muscle activity; the median frequency, showing trends over time and indicating muscle fatigue; and Welch's Power Spectral Density (PSD), which highlights the frequency components of the signal.

4.6 Force Cube Analysis in *vailá*

The *vailá* system supports a robust workflow for analyzing force platform data using the Force Cube method. This process involves analyzing the vertical ground reaction force (Fz) and calculating key biomechanical metrics, such as peak forces, rate of force development (RFD), and stiffness. The system is designed to allow batch processing of multiple files, and each analysis is saved automatically.

The Force Cube Analysis workflow consists of several steps described in Figure 10.

Select Input
Directory
→
Select Fz
in GUI
→
Define Body
Weight Window
→
Select Force
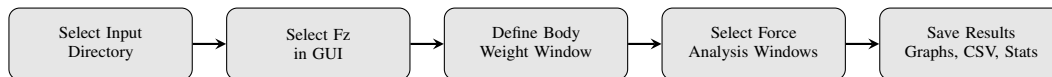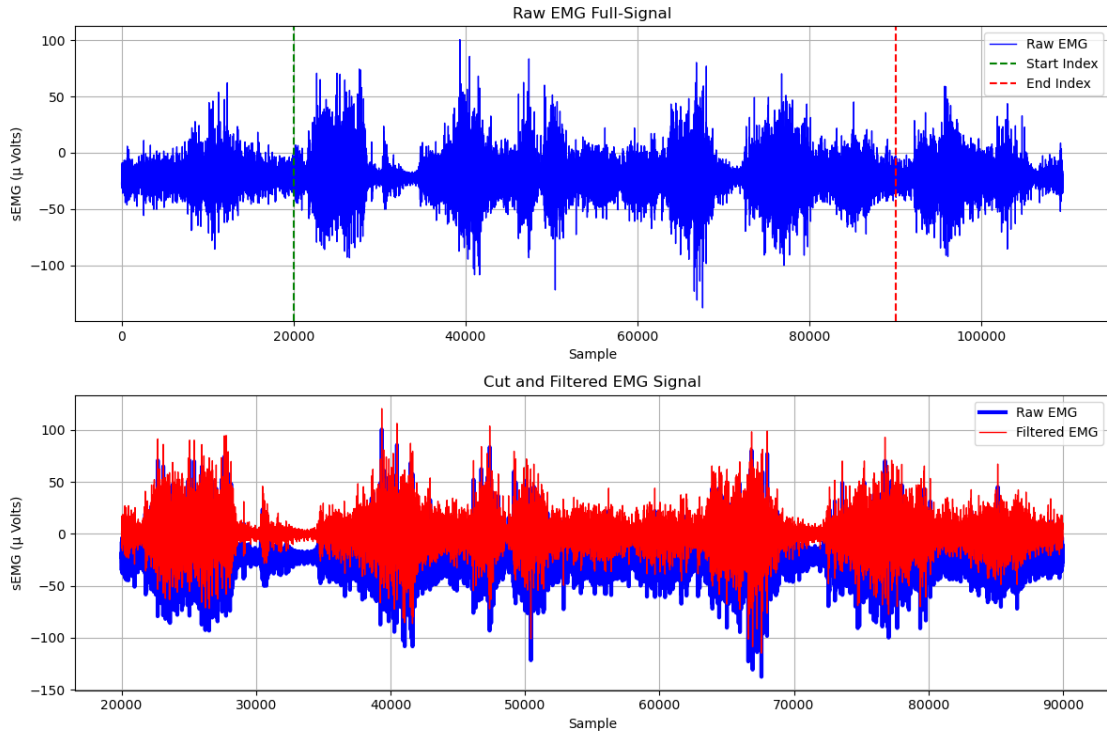Analysis Windows
→
Save Results
Graphs, CSV, Stats

Figure 10: Force Cube Analysis workflow in *vailá*, outlining each step from input directory selection to saving results.
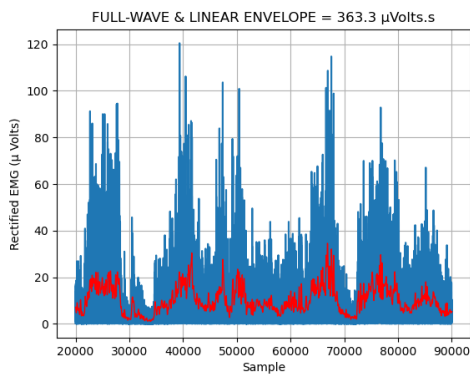
The following Figure 11 illustrates the peak selection process (a) and the generated force-time curve (b), which are key outputs of the Force Cube Analysis.

Each output file generated by the Force Cube Analysis contains key biomechanical metrics that are crucial for understanding the force dynamics during specific time windows. The metrics include:
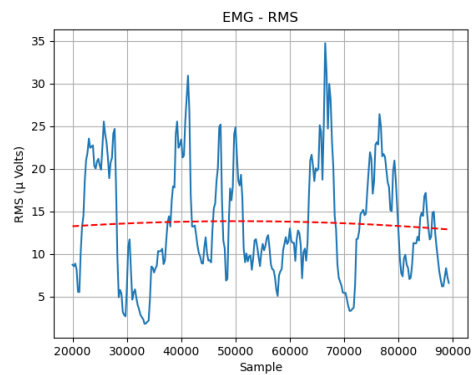File name (`FileName`), the date and time of the analysis (`TimeStamp`), the trial number (`Trial`), and the body weight in kilograms (`BW_kg`). The analysis also records which foot was used (right or left) (`SideFoot_RL`) and foot dominance (`Dominance_RL`). A user-provided quality rating (`Quality`) and the number of samples in the trial (`Num_Samples`) are also included. Indices for specific time points such as 40 ms (`Index_40ms`), 100 ms (`Index_100ms`), the impact transient (`Index_ITransient`), the vertical impact peak (`Index_VIP`), and the maximum force peak (`Index_Max`) are provided. The trial's duration (`Test_Duration_s`), cumulative time sum (`CumSum_Times_s`), contact time (`Contact_Time_s`), and specific times for 40 ms, 100 ms, impact transient, vertical impact peak, and maximum force (`Time_40ms_s`, `Time_100ms_s`, `Time_ITransient_s`, `Time_VIP_s`, and `Time_Peak_VMax_s`) are also included. Key force-related metrics such as peak forces at different times (normalized to body weight) are recorded: `VPeak_40ms_BW`, `VPeak_100ms_BW`, `Peak_VITransient_BW`, `Peak_VIP_BW`, and `Peak_VMax_BW`. The total impulse (`Total_Imp_BW.s`) and impulse at various stages (`Imp_40ms_BW.s`, `Imp_100ms_BW.s`, `Imp_ITransient_BW.s`, `Imp_Brake_VMax_BW.s`, `Imp_Propulsion_BW.s`) are calculated, alongside the rate of force development (RFD) at
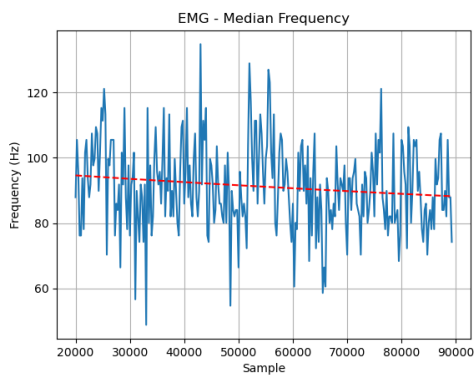
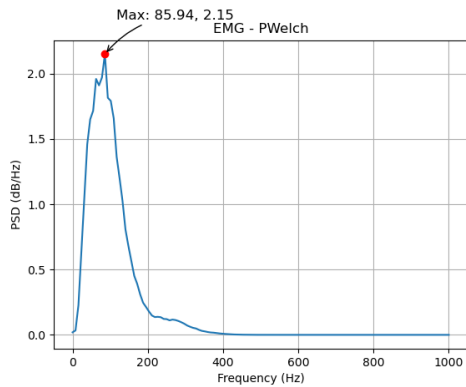(a) Filtered EMG signal

(b) Rectified EMG

(c) RMS values

(d) Median Frequency

(e) Welch's PSD

Figure 9: Key results of EMG analysis from *vailá* for file `emg_01.csv`. Top: filtered EMG signal. Bottom: rectified signal, RMS values, median frequency, and Welch's PSD.
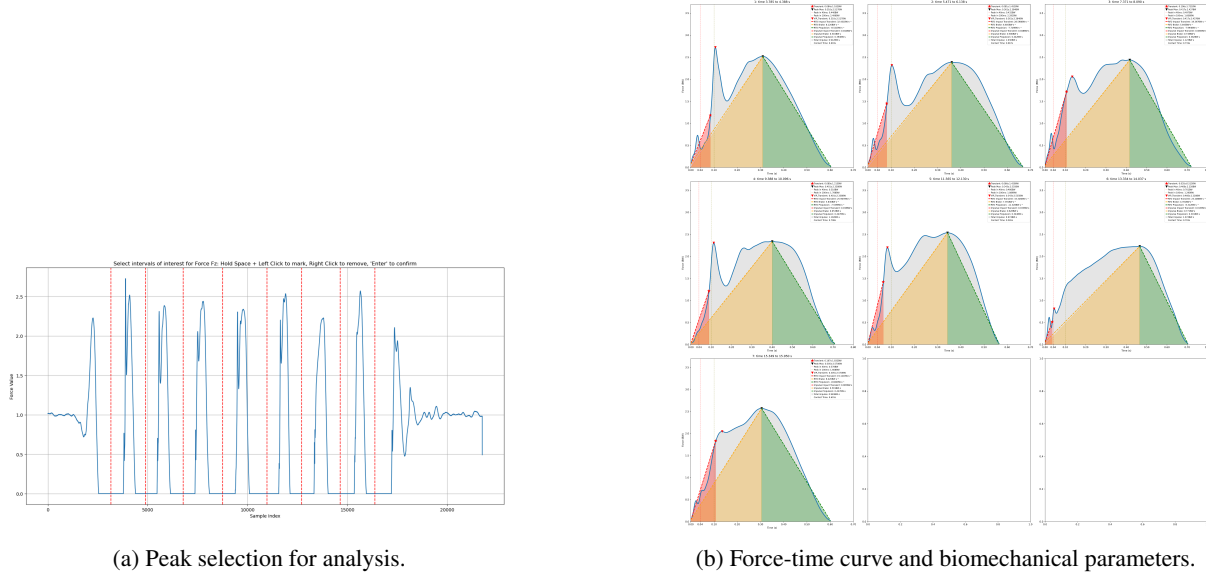
(a) Peak selection for analysis.

(b) Force-time curve and biomechanical parameters.

Figure 11: Force Cube Analysis outputs: (a) peak selection and (b) force-time curve.

different intervals (`RFD_40ms_BW.s`$^{-1}$, `RFD_100ms_BW.s`$^{-1}$, `RFD_ITransient_BW.s`$^{-1}$, `RFD_Brake_VMax_BW.s`$^{-1}$, and `RFD_Propulsion_BW.s`$^{-1}$). Finally, stiffness parameters (`Simple_stiffness_constant`, `High_stiffness`, `Low_stiffness`, `Transition_time`) and the average loading rate (`Average_loading_rate`) complete the set of metrics.

4.7 CoP Balance Analysis in *vailá*

The *vailá* system includes a comprehensive workflow for performing center of pressure (CoP) balance analysis based on force plate data. The analysis begins by selecting the CSV file directory with force plate data. After confirming the default parameters, the user selects the relevant CoP data columns, such as Cx (mediolateral) and Cy (anteroposterior) coordinates, through the graphical interface. Once confirmed, all figures and metrics are generated and saved automatically Figure 12.

Figure 12: Workflow for CoP Balance Analysis in *vailá*, excluding the confirmation step.

The key outputs generated by *vailá* include time-domain and frequency-domain metrics, as discussed in the literature by Duarte and Freitas [Duarte and Freitas, 2010], such as the CoP path length, mean speed, and power spectral density (PSD) for both mediolateral (ML) and anteroposterior (AP) directions. These metrics provide insights into the subject's balance stability and postural control.

The outputs generated from the CoP analysis include the following: an Overview Image, which provides a summary of key metrics such as total path length and mean speed; the Final Figure, showcasing detailed results including the CoP path plotted over time; a Metrics CSV File, which contains numerical values for key balance metrics such as mean ML and AP positions, RMS, and total path length; a PSD Plot, illustrating the power spectral density (PSD) and the frequency distribution of CoP movement; and a Stabilogram, depicting the CoP trajectory over time in both ML and AP directions.

The key balance metrics and frequency-domain characteristics (e.g., total power and frequency dispersion) are detailed in the CSV file. The results enable a robust posturographic analysis in line with the methodology proposed by Duarte and Freitas [Duarte and Freitas, 2010].

Figure 13 shows an example of the CoP analysis outputs: the stabilogram, PSD, general CoP overview, and the final figure showing the CoP path over time.

16

(a) CoP stabilogram.

(b) Power spectral density (PSD).

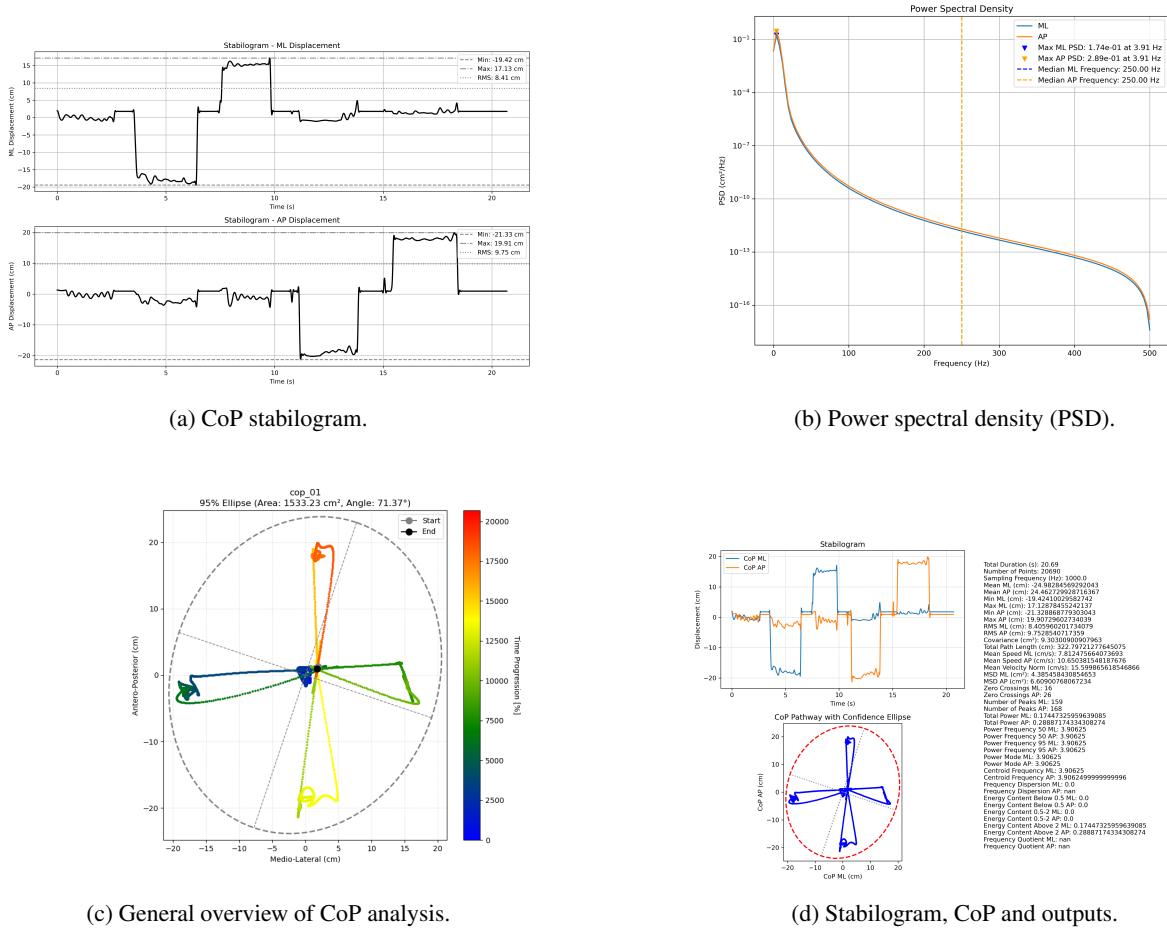(c) General overview of CoP analysis.

(d) Stabilogram, CoP and outputs.

Figure 13: Center of Pressure (CoP) analysis outputs: stabilogram (a), PSD (b), general overview (c), and final Stabilogram and CoP path with ellipse 95% figure (d).

5   Discussion

vailá represents a significant advancement in biomechanical data analysis by offering a comprehensive, multimodal, open-source toolbox that integrates various data sources into a single environment. Its development addresses the complexities and time-consuming nature of processing kinematic data, particularly from video sources, as highlighted in recent studies [Palucci Vieira et al., 2022, Bini et al., 2023, dos Santos Banks et al., 2024, Monteiro et al., 2024]. For instance, Palucci Vieira et al.Palucci Vieira et al. [2022] demonstrated the substantial time investment required for 3D movement kinematic analysis in soccer using markerless motion detection methods compared to traditional digitization. Similarly, Bini et al.[Bini et al., 2023] discussed the challenges in assessing lower limb motion during cycling with neural networks, emphasizing the need for specialized algorithms and computational resources.

In addition to kinematic data, the integration of kinetic data is crucial for a comprehensive understanding of human movement biomechanics. Kinetic analyses provide insights into the forces and moments acting on the body, which are essential for evaluating balance, stability, and functional performance. Berg-Poppe et al. [Berg-Poppe et al., 2018] examined the concurrent validity between a portable force plate and an instrumented walkway when measuring limits of stability, highlighting the importance of accurate kinetic measurements in identifying balance deficits that can underlie various disorders. Their findings emphasize the need for accessible tools capable of processing both kinematic and kinetic data to fully assess human movement.

These studies underscore the necessity for efficient, accessible, and user-friendly tools in biomechanics research. Free and open-source tools facilitate research by reducing costs and have the potential for greater citation impact due to their accessibility and adaptability [Vieira et al., 2017]. For example, Vieira et al. [Vieira et al., 2017] developed a

method for tracking futsal players using a wide-angle lens camera, providing their code freely at link and GitHub. Such availability promotes transparency and allows other researchers to build upon their work.

Several free tools like Tracker [Wee, 2013], Kinovea [Abd El-Raheem et al., 2015], and Dvideow [Figueroa et al., 2003] have been instrumental in advancing biomechanical analyses by providing accessible platforms for motion tracking and analysis. Tracker, for instance, is widely used for physics education and research due to its simplicity and effectiveness in tracking motion from videos. Kinovea offers user-friendly video analysis features suitable for sports and medical applications. Dvideow [Figueroa et al., 2003] provides a flexible software solution for tracking markers in human motion analysis.

While these tools have significantly contributed to the field, *vailá* has the potential to advance beyond them by leveraging modern programming languages and technologies. Developed in Python 3.11.9, which includes CPython acceleration, *vailá* benefits from improved performance and the extensive libraries available in the Python ecosystem. Unlike tools developed in C or C++, which may deter users due to their complexity and less widespread use among the general research community, Python's popularity and readability make *vailá* more accessible to a broader audience. The large and active Python community further strengthens the development and adoption of *vailá*, as users can easily find support, resources, and collaborators.

Moreover, *vailá* shares similarities with tools like DeepLabCut [Nath et al., 2019], which package scripts in a user-friendly manner to facilitate markerless pose estimation across species and behaviors. Like DeepLabCut, *vailá* leverages open-source development to make advanced computational tools accessible to a broader audience. However, *vailá* extends this approach by integrating multiple modalities of biomechanical data, including kinetic data, offering a more holistic analysis platform.

A distinctive feature of *vailá* is its commitment to open-source principles and transparency. Unlike proprietary software that often uses executable installers (`.exe`), which can obscure the underlying code and discourage user modification, *vailá* keeps its codebase transparent and accessible on GitHub vaila-multimodaltoolbox. This openness fosters community contributions and encourages users to engage with the code, potentially inspiring them to develop their programming skills in Python. By avoiding closed installers, *vailá* ensures that users can see and modify the code, promoting a deeper understanding of the software's functionality and facilitating customization to meet specific research needs.

The ease of installation and use of *vailá* has been validated by practitioners in laboratory settings. The comprehensive documentation *vailá* in Read the Docs guides users through the installation process and demonstrates how to leverage the tool's capabilities effectively. Originally developed to support a specific research project on classifying single-limb motor development using soccer kicking in young children with cerebral palsy, funded by the CAPES Foundation Brazil (CAPES process: 88887.936639/2024-00), *vailá* has since grown into a versatile tool ready for broader application and community contribution.

Despite its current capabilities, *vailá* acknowledges its limitations and areas for future development. While it offers functionalities such as the Markerless 2D Workflow, Motion Capture Cluster Workflow, EMG Workflow, and Center of Pressure (CoP) Balance Analysis, there is ample room for expansion. Future perspectives include enhancing the user interface for even greater intuitiveness, incorporating advanced machine learning algorithms for data analysis, and extending support to additional data types and devices. Continuous development and community engagement are essential to realize these enhancements.

Furthermore, in this article, we have presented examples of workflows and pipelines within *vailá* that facilitate rapid processing of large datasets in batch mode, independent of the type of data collection. This flexibility is particularly valuable in research scenarios where unexpected opportunities arise and there is a need to process data quickly without losing valuable collections. By enabling swift adaptation to varying data types and collection methods, *vailá* helps researchers efficiently manage and analyze data, even under time constraints or unforeseen circumstances.

The creation of *vailá* was born out of a practical need in a laboratory environment where researchers required modified and versatile code daily for each data collection. This necessity mirrors the sentiment expressed in Albert Einstein's often-misunderstood quote, "Imagination is more important than knowledge." Contrary to common misinterpretation, Einstein did not downplay the importance of knowledge but emphasized that imagination is the driving force behind innovation and the application of knowledge in new ways. In the context of *vailá*, imagination has been the catalyst for developing a tool that challenges traditional norms, offering a flexible and open platform that empowers researchers to explore, experiment, and innovate without constraints.

By embracing imagination and creativity, *vailá* has evolved beyond its initial scope, providing a foundation for researchers and laboratories eager to experiment with new computational approaches. Its design encourages users not

only to utilize existing functionalities but also to contribute new modules and workflows, fostering a collaborative environment where knowledge and imagination synergize to advance the field of biomechanics.

vailá offers a flexible, accessible, and comprehensive solution to the challenges of biomechanical data analysis. By simplifying data processing, integrating multiple data sources—including kinetic data as highlighted by Berg-Poppe et al. [Berg-Poppe et al., 2018]—and promoting open-source development, *vailá* has the potential to empower researchers and practitioners to conduct more efficient and inclusive biomechanical studies. Its contributions could significantly advance the field, leading to improved rehabilitation strategies, performance optimization, and a deeper understanding of human movement. As the tool develops, it invites the community to participate actively in its evolution, embodying the spirit of collaborative innovation essential for scientific progress.

6　Conclusion

vailá offers a flexible, accessible, and comprehensive solution to the challenges of biomechanical data analysis. By simplifying data processing, integrating multiple data sources, and promoting open-source development, *vailá* empowers researchers and practitioners to conduct more efficient and inclusive biomechanical studies. Its contributions have the potential to significantly advance the field, leading to improved rehabilitation strategies, performance optimization, and a deeper understanding of human movement. As the tool develops, it invites the community to participate actively in its evolution, embodying the spirit of collaborative innovation essential for scientific progress.

Acknowledgements

Disclosure statement

The authors declare that they have no competing financial interests or personal relationships that could have appeared to influence the work reported in this paper. The project is open-source, entirely free, and non-commercial, aimed solely at those interested in studying and advancing biomechanical analysis.

Funding

Availability of Data and Materials

The *vailá* toolbox is available at our GitHub repository: <https://github.com/vaila-multimodaltoolbox/vaila>. All code is open-source, and we encourage contributions from researchers, developers, and users. The repository includes detailed documentation, example datasets, and installation scripts for Linux, macOS, and Windows.

We welcome feedback, suggestions, and code contributions to help improve and expand *vailá* to meet the needs of the biomechanics community. Please feel free to open issues, submit pull requests, or join discussions to contribute to the project.

References

Anthony L Ricamato and Joseph M Hidler. Quantification of the dynamic properties of emg patterns during gait. *Journal of electromyography and kinesiology*, 15(4):384–392, 2005.

M. Roberts, D. Mongeon, and F. Prince. Biomechanical parameters for gait analysis: a systematic review of healthy human gait. *Phys. Ther. Rehabil*, 4(6), 2017. doi:10.7243/2055-2386-4-6.

Guilherme M Cesar, Curtis L Tomasevicz, and Judith M Burnfield. Frontal plane comparison between drop jump and vertical jump: implications for the assessment of acl risk of injury. *Sports Biomechanics*, 15(4):440–449, 2016.

Bruno Luiz Souza Bedo, Guilherme Manna Cesar, Renato Moraes, Fábio Pamplona Mariano, Luiz Henrique Palucci Vieira, Vitor Luiz Andrade, and Paulo Roberto Pereira Santiago. Influence of side uncertainty on knee kinematics of female handball athletes during sidestep cutting maneuvers. *Journal of applied biomechanics*, 37(3):188–195, 2021.

Guilherme M Cesar, Thad W Buster, Arash Mohammadzadeh Gonabadi, and Judith M Burnfield. Muscle demand and kinematic similarities between pediatric-modified motor-assisted elliptical training at fast speed and fast overground walking: Real-world implications for pediatric gait rehabilitation. *Journal of electromyography and kinesiology*, 63: 102639, 2022.

A. Barre and S. Armand. Biomechanical toolkit: Open-source framework to visualize and process biomechanical data. *Comput Methods Programs Biomed*, 114(1):80–87, 2014.

Ajay Seth, Jennifer L Hicks, Thomas K Uchida, Ayman Habib, Christopher L Dembia, James J Dunne, Carmichael F Ong, Matthew S DeMers, Apoorva Rajagopal, Matthew Millard, et al. Opensim: Simulating musculoskeletal dynamics and neuromuscular control to study human and animal movement. *PLoS computational biology*, 14(7): e1006223, 2018.

Bruno L. S. Bedo, Danilo S. Catelli, Mario Lamontagne, and Paulo R. P. Santiago. A custom musculoskeletal model for estimation of medial and lateral tibiofemoral contact forces during tasks with high knee and hip flexions. *Computer Methods in Biomechanics and Biomedical Engineering*, 23(10):658–663, 2020a.

Bruno L. S. Bedo, A Mantoan, D. S. Catelli, W. Cruaud, M. Reggiani, and M. Lamontagne. 474 bops: a matlab toolbox to batch musculoskeletal data processing for opensim. *Computer 475 Methods in Biomechanics and Biomedical Engineering*, pages 1–11, 2020b.

Guido van Rossum and the Python Development Team. Python 3.11.0 release. https://docs.python.org/3/whatsnew/3.11.html#whatsnew311-faster-cpython, 2022. Accessed: 2024-09-24.

J. M. Smith, A. K. Thompson, and et al. Markerless motion capture using deep learning techniques: A systematic review. *Computer Methods in Biomechanics and Biomedical Engineering*, 2023. doi:10.1016/j.compbiomed.2023.104210.

Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Guang Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, and Matthias Grundmann. Mediapipe: A framework for building perception pipelines, 2019. URL https://arxiv.org/abs/1906.08172.

Jonathan Matthis, Aaron Cherian, and Trent Wirth. The freemocap project - and - gaze/hand coupling during a combined three-ball juggling and balance task. *Journal of Vision*, 22(14):4195, 2022. doi:10.1167/jov.22.14.4195.

Loo Kang Wee. Open source physics. *arXiv preprint arXiv:1308.2614*, 2013.

Pascual J Figueroa, Neucimar J Leite, and Ricardo ML Barros. A flexible software for tracking of markers used in human motion analysis. *Computer methods and programs in biomedicine*, 72(2):155–165, 2003.

Frederick P Brooks Jr. *The mythical man-month: essays on software engineering*. Pearson Education, 1995.

Frederick P. Brooks. The mythical man-month: After 20 years. *IEEE Software*, 12(5):57–60, 1995. doi:10.1109/MS.1995.10042.

Gnu general public license, version 3. http://www.gnu.org/licenses/gpl.html, June 2007. Last retrieved 2020-01-01.

Marcos Duarte and Sandra MSF Freitas. Revision of posturography based on force plate for balance evaluation. *Brazilian Journal of physical therapy*, 14:183–192, 2010.

Luiz H Palucci Vieira, Paulo RP Santiago, Allan Pinto, Rodrigo Aquino, Ricardo da S Torres, and Fabio A Barbieri. Automatic markerless motion detector method against traditional digitisation for 3-dimensional movement kinematic analysis of ball kicking in soccer field context. *International journal of environmental research and public health*, 19 (3):1179, 2022.

Rodrigo Rico Bini, Gil Serrancoli, Paulo Roberto Pereira Santiago, Allan Pinto, and Felipe Moura. Criterion validity of neural networks to assess lower limb motion during cycling. *Journal of sports sciences*, 41(1):36–44, 2023.

Luíza dos Santos Banks, Paulo Roberto Pereira Santiago, Ricardo da Silva Torres, Donizete Cicero Xavier de Oliveira, and Felipe Arruda Moura. Accuracy of a markerless system to estimate the position of taekwondo athletes in an official combat area. *International Journal of Performance Analysis in Sport*, pages 1–16, 2024.

Rafael Luiz Martins Monteiro, Carlos Cesar Arruda Dos Santos, Patrick Blauberger, Daniel Link, Tiago Guedes Russomanno, Ariany Klein Tahara, Abel Gonçalves Chinaglia, and Paulo Roberto Pereira Santiago. Enhancing soccer goalkeepers penalty dive kinematics with instructional video and laterality insights in field conditions. *Scientific Reports*, 14(1):10225, 2024.

Patti Berg-Poppe, Guilherme M Cesar, Hanz Tao, Cal Johnson, and Jessica Landry. Concurrent validity between a portable force plate and instrumented walkway when measuring limits of stability. *International Journal of Therapy and Rehabilitation*, 25(6):272–278, 2018.

Luiz HP Vieira, Emilio A Pagnoca, Fabio Milioni, Ricardo A Barbieri, Rafael P Menezes, Luis Alvarez, Luis G Déniz, Daniel Santana-Cedrés, and Paulo RP Santiago. Tracking futsal players with a wide-angle lens camera: accuracy analysis of the radial distortion correction based on an improved hough transform algorithm. *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, 5(3):221–231, 2017.

Reham M Abd El-Raheem, Ragia M Kamel, and Mohammad F Ali. Reliability of using kinovea program in measuring dominant wrist joint range of motion. *Trends in Applied Sciences Research*, 10(4):224, 2015.

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