Streamlining C3D File Processing and Visualization: A User-Friendly Approach Using Google Colab and Open-Source Python Packages

Authors:

- Hossein Mokhtarzadeh, Senior Research Fellow at the University of Melbourne
  - Github: hmok (Hossein Mokhtarzadeh) · GitHub
  - Email: Hossein Mokhtarzadeh
- Soroosh Bagheri, CEO - Founder at BSNlab.com
  - Website: www.BSNlab.com - www.C3Dtools.com
  - Github : https://github.com/etoshey
  - Email: soroosh.b.k@gmail.com

Access Our Code: https://github.com/etoshey/colabC3D

Contents:

Abstract:

This abstract presents a comprehensive approach to leverage Google Colab, a cloud-based Python development environment, for processing and visualizing C3D files. By combining the power of C3DTools, Three.js, and Pygwalker, we provide researchers, practitioners, and enthusiasts with a flexible and efficient platform to gain valuable insights from C3D files across diverse domains. We outline the setup process, including the installation of C3DTools, Three.js, and Pygwalker, within the Google Colab environment. With these tools, users can apply filtering, transformation, and interpolation techniques to manipulate the data. We enhance visualization using Three.js, creating interactive representations to explore and analyze motion capture data. Pygwalker facilitates the visualization of graphs and we export data in different formats such as pandas, trc, mot or csv. Comparing our results with the established Python package ezc3d for Center of Pressure (CoP) calculations, we achieve a high correlation ($R \geqslant 0.99$). The integration of these tools within Google Colab provides a collaborative and accessible platform, eliminating the need for local installations and extensive hardware requirements. Access to the notebook on GitHub allows users to freely explore C3D files at https://github.com/etoshey/colabC3D. By leveraging Google Colab, Python, and open-source libraries, this approach enables efficient processing, visualization, and analysis of C3D files, empowering users to uncover valuable insights in the field of biomechanics and motion capture.

Keywords: C3D, Google Colab, Python, Biomechanics, Visualization, Open-source, Motion capture

I. Introduction

Motion capture data holds immense value in numerous domains, including animation, biomechanics, sports analysis, and robotics. To make sense of this data, efficient processing and visualization techniques are vital, particularly for the widely adopted C3D file format. This paper introduces a novel approach that leverages the power of Google Colab, an online Python development environment, to process and visualize C3D files. By incorporating libraries like C3DTools, Three.js, Pygwalker, and others, we aim to overcome the challenges associated with traditional local installations and hardware constraints and and cater to non-programmer users.

The main driving force behind this research is to address the limitations posed by cumbersome setups and resource-intensive requirements when working with motion capture data. With Google Colab, researchers, practitioners, and enthusiasts can access a collaborative and accessible platform that provides the necessary computational capabilities and pre-installed libraries, eliminating the need for extensive configuration.

Our proposed methodology revolves around the seamless installation and configuration of essential libraries, such as C3DTools, Three.js, and Pygwalker, within the Google Colab environment. C3DTools equips users with a comprehensive toolkit for manipulating C3D files, enabling essential functionalities like data transformation, and interpolation. Integrating Three.js empowers users to create immersive 3D visualizations, allowing for interactive exploration and analysis of motion capture data within a web-based environment. Moreover, Pygwalker facilitates the effortless export and display of graphs, enabling users to visualize analysis results directly within the Google Colab environment.

In our experimental setup, we provide insights into the dataset used for evaluation and walk through the process of configuring the Google Colab environment. The results and analysis section showcases the visualizations of processed C3D data using Three.js and the display of exported graphs using Pygwalker. Finally, we conclude the paper by discussing the strengths and limitations of our proposed approach, as well as the potential applications and future directions for C3D file processing and visualization.

II. Related Work

A. Review of existing research and tools for C3D file processing and visualization

Several existing tools and software solutions are available for C3D file processing and visualization. Commercial and non-commercial options such as Mokka (non-commercial but not updated any more), Nexus from Vicon (commercial), ezc3d and kineticstookkit, OpenSim C3D adapter (based on ezc3d package), pyc3d [1-5], and other motion capture system providers offer comprehensive features but are not freely available and may lack easy integration with programming languages like Python, despite providing APIs. While these tools are commonly used in research and clinical settings, they often require software installations on local machines (Table 1).

However, the fundamental task of reading and visualizing the content of C3D files should not be a challenging endeavor for researchers and clinicians (Table 1, e.g. ease of use and user learning curve as well as skill level). It is crucial for them to have an easy and accessible means of examining and manipulating the data without the burden of software installation on their personal computers or laptops.

In this regard, the proposed approach aims to address these concerns by utilizing online and cloud-based systems like Google Colab [6, 7]. By leveraging this platform, users can access the necessary computational resources without worrying about software installations or hardware limitations. The cloud-based environment ensures a seamless experience where researchers and clinicians can focus on data analysis rather than the technicalities of setting up the required software.

Nevertheless, it is important to consider the risks associated with using online and cloud-based systems. Security and privacy concerns should be taken into account to safeguard sensitive

data and ensure compliance with relevant regulations. Despite these considerations, the benefits of an easily accessible and user-friendly environment for C3D file processing and visualization outweigh the potential risks, making the proposed approach a valuable solution for researchers and clinicians in need of seamless data analysis capabilities.

Table 1: This table presents a comparison of Software Options: Google Colab (Cloud-Based) e.g., c3dColab in this paper, Web-Based Option such as C3Dtools.com, and Locally Installed such as ezc3d. The table highlights various aspects including accessibility, installation requirements, dependency management, execution environment, collaboration capabilities, performance, data storage, visualization possibilities, customization options, cost, user learning curve, ease of use, and required skill level.

Aspect	**Google Colab (Cloud-Based)**	**Web-Based Option**	**Locally Installed Option**
Accessibility	Anywhere with internet	Web browser	Specific device
Installation	No installation required	No installation required	Installation required
Dependencies	Install via pip, conda	Install via web package manager	Install via package manager
Execution Environment	Remote servers	Remote servers	Local machine
Collaboration	Easy sharing and collaboration	Varies	Varies
Performance	Depends on internet speed and server load	Depends on internet speed and server load	Depends on local machine resources
Data Storage	Google Drive or cloud storage	Varies	Local storage
Visualization	Libraries available (e.g., pygwalker, three.js)	Web-based libraries	Local visualization libraries
Customization	Limited customization options	Varies	Greater customization
Cost	Free with optional paid resources	Varies	One-time purchase or subscription-based
User Learning Curve	Moderate	Varies	Steeper
Ease of Use	User-friendly interface	Varies	Depends on familiarity
Skill Level	Basic programming skills	No programming skills required	Intermediate to advanced programming skills
Development Possibility by User	High	None	High

B. Identification of gaps or limitations in current approaches

While there are existing tools available for C3D file processing and visualization, there are certain gaps and limitations that need to be addressed. One key limitation is the lack of freely available and open-source solutions. Many widely-used commercial software packages such as Nexus from Vicon, require substantial financial investments, making them less accessible to researchers and clinicians with limited budgets.

In response to this limitation, some open-source Python packages like ezc3d and pyc3d have been developed. These packages provide functionalities for working with C3D files, but they often require a certain level of expertise to effectively utilize and visualize the data. The learning curve associated with these packages can be a barrier for researchers and clinicians who may not have extensive programming experience.

The proposed approach aims to simplify the process of C3D file visualization, even with the utilization of existing open-source packages. By leveraging the user-friendly framework provided by Google Colab, researchers and clinicians can take advantage of these packages without needing extensive expertise in programming (Table 1). Moreover, researchers could benefit from focusing on their own research questions to address clinical and research hypotheses, such as feature detection of pathological gait, by leveraging the vast array of free and open-source packages available, such as TensorFlow, within the Colab platform. This allows for easy analysis and utilization of their C3D files. The goal is to streamline the process and enable users to easily visualize and manipulate C3D data within the Colab environment.

By addressing the gaps and limitations in current approaches, the proposed solution offers a more accessible and simplified workflow for C3D file processing and visualization. It provides researchers and clinicians with the flexibility to utilize both commercial and open-source tools, ensuring that they can leverage the power of these packages without significant barriers to entry.

III. Methodology

A. Introduction to Google Colab and its benefits for C3D file processing:
Google Colab, an online Python development environment, offers numerous benefits for C3D file processing. It provides access to powerful computational resources, eliminating the need for local installations. Collaboration is seamless, and users can work on projects from anywhere with an internet connection (Table 1).

B. Installation and configuration of required libraries and dependencies:
The proposed approach involves installing and configuring necessary libraries and dependencies within the Google Colab environment. This process ensures that C3DTools, Three.js, Pygwalker, and other relevant packages are readily available for data processing and visualization tasks.

C. Using C3DTools for data preprocessing and manipulation:
C3DTools, a Python API designed for C3D file manipulation, plays a vital role in the proposed approach. Researchers can utilize its functionalities for data preprocessing, calibration, and metadata extraction. It showcases how C3DTools enhances the analysis of C3D files.

D. Integration of Three.js for 3D visualization of C3D files:
The integration of Three.js, a JavaScript library, enables immersive 3D visualization of C3D files. Three.js leverages WebGL to provide real-time rendering capabilities within web browsers. This highlights the seamless integration and showcases the interactive 3D visualizations that can be generated for C3D files within the Colab environment.

E. Utilization of Pygwalker for exporting and displaying graphs:
Pygwalker, a Python package, allows users to export and display graphs derived from C3D data. It offers various graph types, including kinematic graphs, dynamic analysis results, and statistical plots. Our proposed process, outlined in Figure 1, empowers users to read C3D files in Google Colab, visualize data, and export it in common formats like Excel and Pandas, leveraging the capabilities of open-source Python packages like Pygwalker.

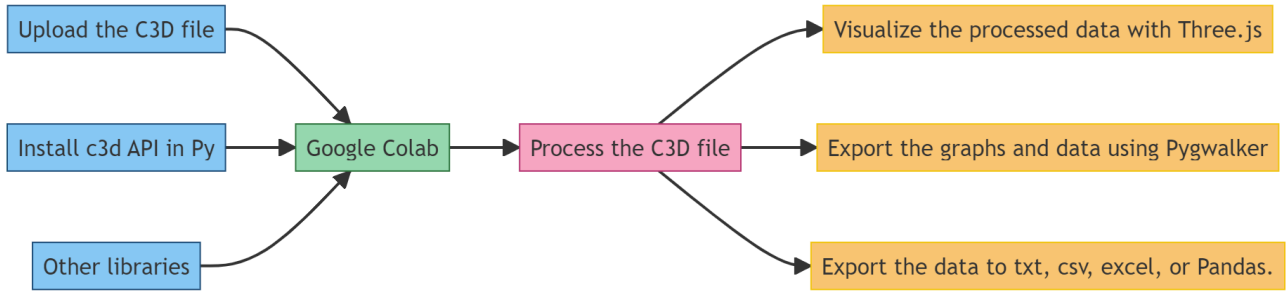

Figure 1. A framework for visualizing and reading C3D files in Google Colab, plotting data, and exporting it in standard formats such as Excel and Pandas, utilizing open-source Python packages.

IV. Experimental Setup

A. Description of the dataset used for evaluation:
The dataset used for evaluation has been sourced from various reliable sources, including https://www.c3d.org/sampledata.html [8]. These datasets comprise data from different force plate types. The C3D data consists of marker data, ground reaction forces (GRFs), and other analog data such as EMG and similar measurements. To demonstrate the capabilities, we utilized several files available for download and further analysis on GitHub [9]. You can replicate all the results by using the provided notebooks at https://github.com/etoshey/colabC3D. Specifically, for this particular analysis, we utilized the file "walking2.c3d" from the following link. Please be aware that during our testing, we encountered certain C3D files that failed to work

correctly with either C3DTools or ezc3d. This is an important observation that needs to be taken into consideration. If you have any of these problematic files, we kindly request you to share them with us. This will enable us to perform debugging and address the issues accordingly. For assistance, please contact Soroosh Bagheri via email: soroosh.b.k@gmail.com.

B. Configuration of the Google Colab environment:
The specific configuration of the Google Colab environment has been outlined in the Jupyter notebook provided by this article and on GitHub. The Python version used in Colab was 3.10.12 (main, June 7 2023, 12:45:35) [GCC 9.4.0], and the codes were tested successfully under this version. In the notebook, instructions on how to install all the required libraries are provided. It's worth noting that no GPU or TPU acceleration is required for the experiments to ensure the reproducibility of the results.

C. Details of the experiments conducted and parameters used for validation:
The proposed approach underwent rigorous evaluation, with detailed coding in Colab experiments well documented. To validate the outcomes, two alternative methods i.e. C3DTools and the ezc3d packages in Python were compared. The center of pressure, extracted from imported C3D files, served as a validation parameter. Additionally, correlation analyses were conducted to assess the level of agreement between the two methods quantitatively (Refer to Figures 1S and 2S, along with Tables 1S and 2S, found in the Appendices for additional details).

Preprocessing steps, such as filtration, were not performed. However, detailed visualization and graph export procedures were carried out using Three.js and Pygwalker, which were explained in the accompanying Jupyter notebooks provided with this paper. All tasks performed on the C3D data, including visualization, and graph export, were meticulously documented within the code. Additionally, the specific parameters used for each experiment, such as visualization options, were well-documented to facilitate the replication of results. To reproduce our results, users can execute the Jupyter notebooks found in our GitHub repository directly on Google Colab.

V. Results and Analysis

A. Presentation of processed C3D data and visualizations using Three.js:
The processed C3D data are presented, showcasing the effectiveness of the proposed approach in transforming, and visualizing the motion capture data. Interactive 3D visualizations created using Three.js are demonstrated, highlighting the ability to explore complex motions and spatial relationships (Figure 2).

B. Showcase of exported graphs and their interpretation using Pygwalker:

The graphs exported using Pygwalker are showcased, illustrating the analysis results derived from the C3D data. Various graph types, such as kinematic graphs or plots, are presented. The interpretation of these graphs and their insights into the motion capture data will be discussed (Figure 3).

C. Comparison and discussion of the results with existing approaches:
We have performed an extensive comparison to assess the efficacy of our proposed method, particularly in contrast with the ezc3d Python package. Please refer to the appendices for the detailed comparative analysis. The strengths and limitations of the proposed approach will be discussed (Appendix 1, 2), along with its potential advantages over traditional or commercial solutions. The results can be contextualized within the broader field of C3D file processing and visualization.

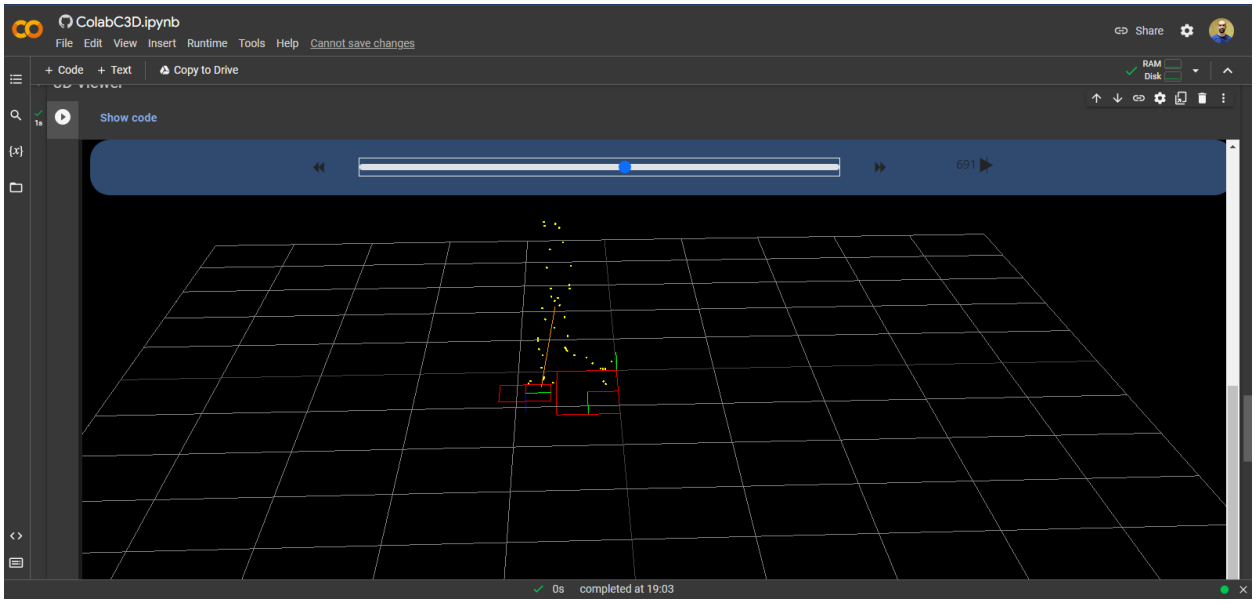

Figure 2: Visualization of C3D file in Colab: In this image, markers and GRF (Ground Reaction Force) are visible, allowing for a comprehensive view of the data. The user has the ability to interact with the visualization by using their mouse and keyboard to make basic rotation and displacement changes in various anatomical planes.

In Figure 3, we present a figure generated by pygwalker, highlighting marker and GRF data. Although we have only provided a sample of marker data and GRF in the figure, it's important to emphasize that the Colab environment enables visualization, export, and exploration of any other data points present within the C3D file.

Furthermore, it's worth mentioning that the detailed discussion with ezc3d is available in the appendix. As demonstrated in the appendix, the qualitative and quantitative match of the CoP and GRFs results was remarkably accurate (Figures 1S, 2S and Tables 1S, 2S).

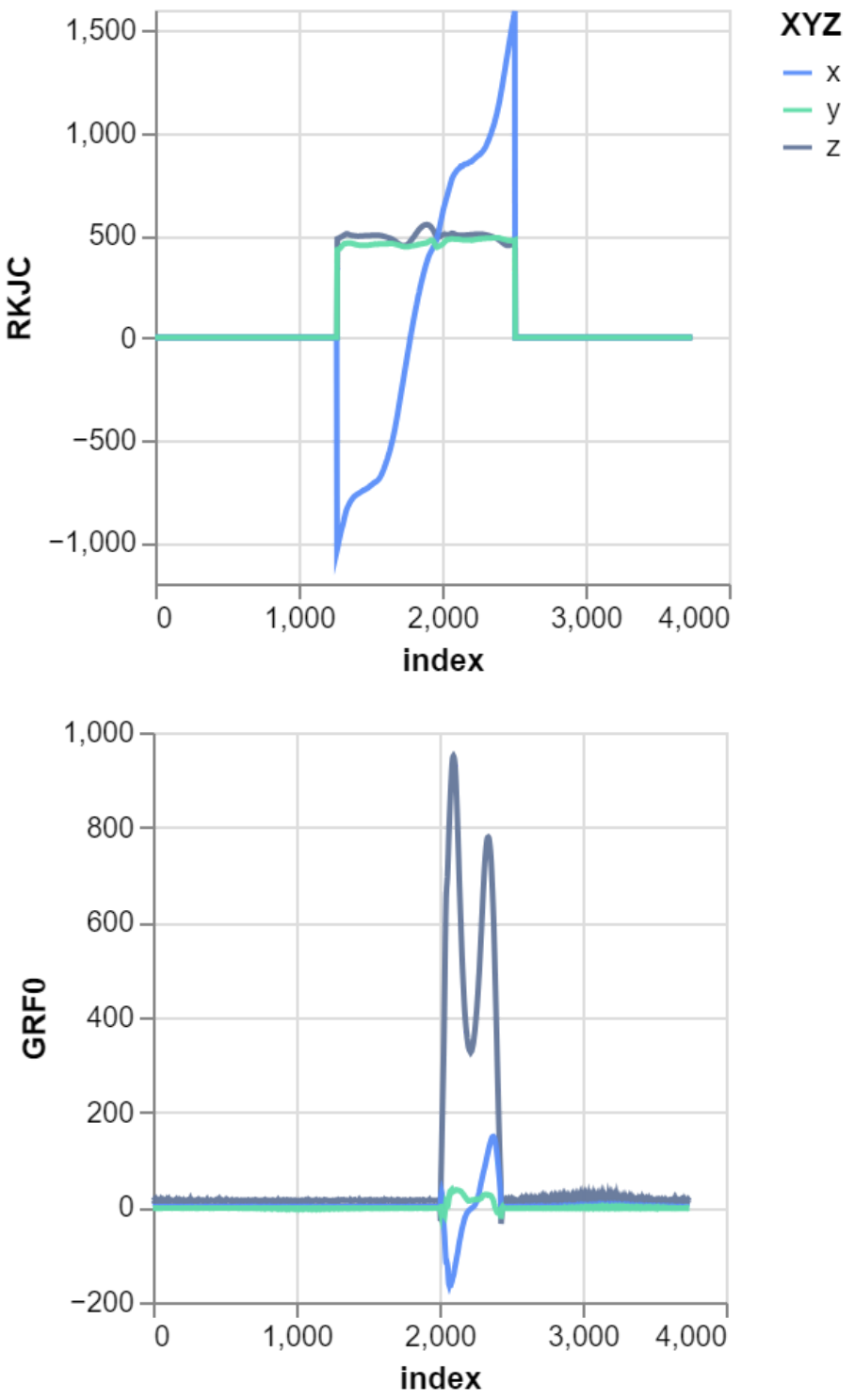

Figure 3: Graph Visualization of C3D File with Pygwalker in Colab. This image showcases a comprehensive view of a C3D file using Pygwalker. It displays a marker and Ground Reaction Force (GRF) graph, enabling interactive exploration within Colab. Users can export graphs as png or svg formats and perform data analyses seamlessly within the Colab environment.

VI. Discussion

A. An Evaluation of the proposed approach's strengths and limitations:

The proposed method for streamlining and visualizing C3D files in Colab stands out for several strengths. It is the first of its kind, offering a unique and innovative approach. The method provides significant benefits to both researchers and clinicians. Users can efficiently organize their data within minutes without the need for system or PC installations, requiring only Gmail access and an internet connection. The outcomes generated through this method can be easily shared, fostering seamless collaboration with colleagues while ensuring the reproducibility of simulations and processes. Another notable strength is the framework's ability to visualize C3D files directly on the web in Colab, eliminating the need for PC installations for visualization purposes. The framework enables direct processing of data points within C3D files, allowing users to visually inspect their outcomes. Additionally, users can effortlessly export their files to other Musculoskeletal (MSK) models such as OpenSim, or convert them to ASCII, CSV, and pandas formats. The inclusion of the "pgwalker" plotting option further enhances users' understanding of the data by enabling the plotting of different data points for marker data, GRF, or COP with ease. These strengths collectively contribute to the effectiveness and usability of the proposed method.

When utilizing C3D files in Colab for processing, it is crucial to consider and address the following limitations. First, there are security risks associated with cloud-based data sharing, which can be mitigated through encryption, access controls, and regular security updates. Secondly, the current framework only supports processing a single C3D file, but it can be expanded to handle multiple files by making necessary modifications. Additionally, Colab has limitations in terms of computational resources, file size, and upload limits, which may require preprocessing or working with smaller subsets of data. Internet connectivity is crucial for uninterrupted processing. Ensuring compatibility with dependencies and addressing data privacy and compliance requirements are also essential. By addressing these limitations, researchers can optimize the use of C3D files in Colab for efficient processing and analysis.

B. Potential applications and future directions:

The proposed approach holds promise for numerous applications in diverse domains, including animation, biomechanics, sports analysis, and robotics. Its potential can be further expanded by incorporating advanced machine learning techniques, which would enable more sophisticated analysis and interpretation of the data. Additionally, exploring compatibility with other motion capture file formats would enhance the framework's versatility and usability. Furthermore, this framework has the potential to contribute to the standardization of biomechanical analyses by providing a simple and efficient solution to read and visualize various C3D-like format files. This would facilitate easy sharing and collaboration among researchers in the field. Continual exploration of these directions will pave the way for exciting advancements and discoveries in the future.

VII. Conclusion

A. Summary of the contributions and findings:
We have developed a user-friendly framework to read, manipulate, and visualize C3D files on the cloud, specifically on Google Cloud. Our framework utilizes free and open-source Python packages. The validity of our methods has been demonstrated through comparison with an established approach. This framework has the potential to advance biomechanics and related research areas by providing a simple and user-friendly method for C3D file processing and visualization in the cloud.

B. Importance of the proposed approach in C3D file processing and visualization:
To the best of our knowledge, this framework stands out as a simple yet powerful solution for exploring, processing, and visualizing C3D files in the field of C3D file processing. By harnessing the capabilities of Google Colab, C3DTools, ezc3d, Three.js, and Pygwalker, it offers effortless accessibility. This framework empowers users to perform comprehensive analyses of C3D data, facilitating a deeper understanding of motion capture data in a user-friendly manner.

VIII. References

[1]. Michaud, B. and Begon, M., 2021. ezc3d: An easy c3d file i/o cross-platform solution for c++, python and matlab. *Journal of Open Source Software*, *6*(58), p.2911.
[2]. Delp, S.L., Anderson, F.C., Arnold, A.S., Loan, P., Habib, A., John, C.T., Guendelman, E. and Thelen, D.G., 2007. OpenSim: open-source software to create and analyze dynamic simulations of movement. *IEEE transactions on biomedical engineering*, *54*(11), pp.1940-1950.
[3]. Pyc3d GitHub - mkjung99/pyc3dserver: Python interface of C3Dserver software for reading and editing C3D motion capture files
[4]. Barre, A. and Armand, S., 2014. Biomechanical ToolKit: Open-source framework to visualize and process biomechanical data. *Computer methods and programs in biomedicine*, *114*(1), pp.80-87.
[5]. KineticsToolKit GitHub - felixchenier/kineticstoolkit: An Open-Source Python Package to Facilitate Research in Biomechanics
[6]. Mokhtarzadeh, H., Revolutionizing Biomechanics: Integrating AI, Collaborative Coding Environments, and Simulation Software for Enhanced Model Development and Analysis. DOI: https://doi.org/10.31224/3036
[7]. Mokhtarzadeh, H., Jiang, F., Zhao, S. and Malekipour, F., 2022. OpenColab project: OpenSim in Google colaboratory to explore biomechanics on the web. *Computer Methods in Biomechanics and Biomedical Engineering*, pp.1-9.
[8]. C3D Organization. (n.d.). C3D - Standard for the Biomechanics Community. Retrieved from https://c3d.org/ & The C3D File Format User Manual
[9]. Mokhtarzadeh, H., Forte, J.D. and Lee, P.V.S., 2021. Biomechanical and cognitive interactions during visuo motor targeting task. *Gait & Posture*, *86*, pp.287-291.

Acknowledgments:

We thank the developers of Three.js, and Pygwalker for their invaluable contributions to open-source packages in Python. Thanks to the Mermaid team and the creators of its Google Docs extension for enabling us to create a graph effortlessly. Additionally, we acknowledge OpenAI and ChatGPT for their assistance in writing, drafting, and editing this paper.

Appendix **1:**

This appendix shows the comparison between ezc3d and C3Dtools outcomes esp. Center of pressures and ground reaction forces. The user could replicate these and any other calculations within Colab. The C3D files used can also be found here on Github. The comparisons and figures presented in this section are based on the file "walking2.c3d". However, users have the flexibility to compare other files as well [9]. It is important to note that certain files may encounter reading issues with either ezc3d or C3DTools. If you come across any problems, please don't hesitate to reach out to us. We are committed to resolving any issues promptly. For assistance, please contact Soroosh Bagheri via email: soroosh.b.k@gmail.com.

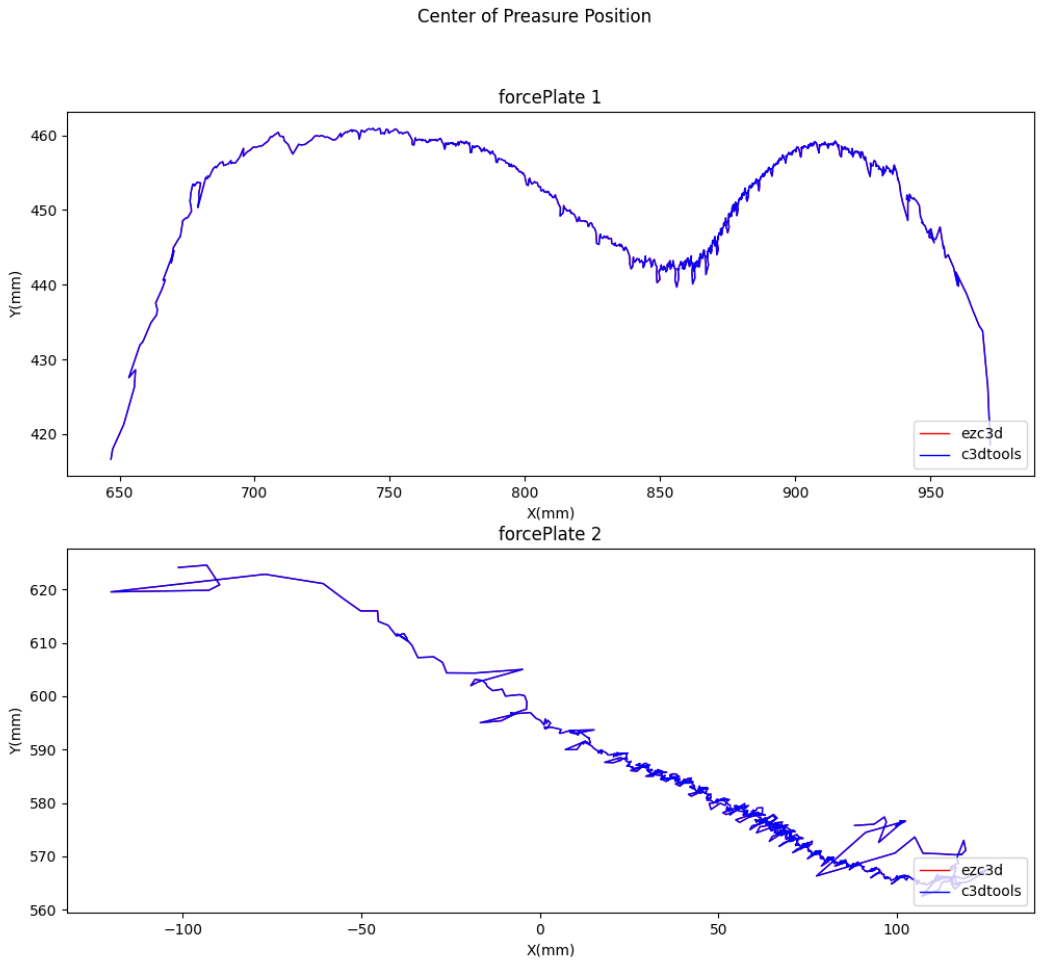

Figure 1S: Center of Pressures calculated by either ezc3d or C3Dtools in Colab. The comparison shows quite perfect match

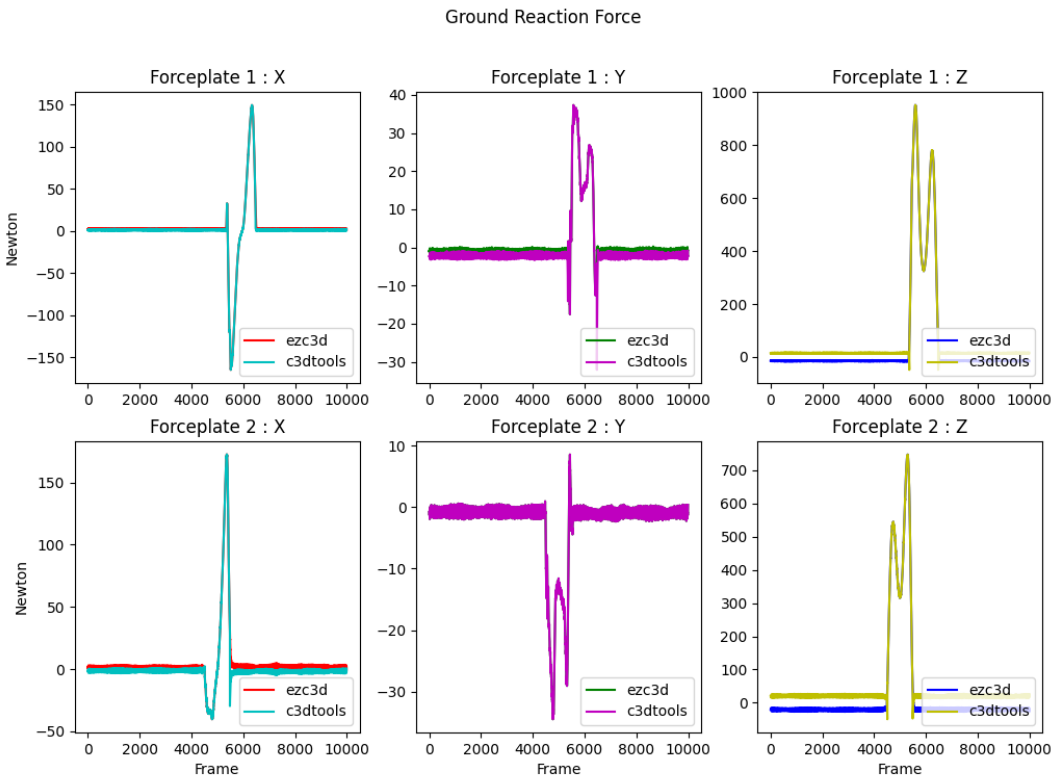

Figure 2S: Ground reaction forces read by either ezc3d or C3Dtools in Colab. The comparison shows quite perfect match

Correlations table:

Correlation coefficients in the tables below, generated using NumPy's correlation coefficient function, to provide a comprehensive comparison between the two methods:

Table 1S: CoP Comparison between Two Methods for Each Force Plate: Pearson Correlation Analysis

Parameter		Pearson Correlation
0	forcePlate 1 : COPX (ezc3d - c3dtools)	1.0
1	forcePlate 1 : COPY (ezc3d - c3dtools)	1.0
2	forcePlate 2 : COPX (ezc3d - c3dtools)	1.0
3	forcePlate 2 : COPY (ezc3d - c3dtools)	1.0

Table 2S: Comparison of Forces between Two Methods for Each Force Plate: Pearson Correlation Analysis

Parameter		Pearson Correlation
0	Forceplate 1 : FX (ezc3d - c3dtools)	1.00
1	Forceplate 1 : FY (ezc3d - c3dtools)	0.99
2	Forceplate 1 : FZ (ezc3d - c3dtools)	0.99
3	Forceplate 2 : FX (ezc3d - c3dtools)	1.00
4	Forceplate 2 : FY (ezc3d - c3dtools)	0.99
5	Forceplate 2 : FZ (ezc3d - c3dtools)	0.99

All of these outcomes can be verified and reproduced in the relevant notebook, which is accompanied by this paper and available on GitHub.

Appendix **2:**

The notebooks in this paper are freely available in the github link.

1. "c3dColab" notebook: Learn to read c3d files, analyze marker or ground reaction force (GRF) data, and gain insights through visualization for research or clinical work.

2. "Validation" or “Comparision” notebook: Compare our current c3dTools method with the Python tool "ezc3d" for validation. Results show a perfect match, ensuring accuracy in data processing.

Explore these notebooks for a deeper understanding of C3D content and reliable data analysis.