Drop Vertical Jumps to Assess Risk for Knee Degeneration and Future Total Knee Arthroplasty

Group Members and Mentor

- Juliana El Rayes
- Eswar Kumar Katragadda
- Vani Pailla

Mentor/advisor: Nathan Schilaty, DC, PhD

Introduction

Knee osteoarthritis is a common condition that affects mobility and often leads to joint replacement surgeries such as total knee arthroplasty (TKA). Identifying movement-related risk factors early on can help reduce the need for these procedures. Our project explores this by analyzing knee mechanics during Drop Vertical Jumps (DVJs), using data from both electromyography (EMG) and 3D motion capture. The surface EMG data is decomposed into individual motor units to gain deeper insight into neuromuscular control strategies. These measurements aim to reveal how age-related changes in muscle control may affect joint mechanics and contribute to the progression of osteoarthritis.

EMG and 3D motion capture are widely used in research to measure muscle activity and joint motion, and they are typically synchronized using a hardware trigger. However, a key challenge in this study is that there was a failure of the expected trigger signal to mark jump start and stop. Our role in the project is to assist with the synchronization of these two datasets. We developed a set of tools to extract relevant timing information from the EMG data, to visually inspect signal features, and to align the EMG and motion capture recordings.

The ultimate goal is to enable accurate alignment, allowing researchers to analyze relationships between motor unit control, biomechanical metrics, and age. This may help identify early markers of joint degeneration and inform preventative or rehabilitative interventions.

Problem Statement & Objectives

In a study led by Dr. Nathan Schilaty, surface electromyography (EMG) and 3D motion capture data were collected to analyze neuromuscular control and joint mechanics during four types of jump landings: Drop Vertical Jumps (DVJ), Left Single-Leg Landings (LSLL), Right Single-Leg Landings (RSLL), and Countermovement Jumps (CMJ). These tasks were designed to assess how motor control may change with age and potentially contribute to knee degeneration. However, due to a failure of the trigger signal, the EMG and motion capture recordings are not synchronized, making it difficult to analyze how muscle activity corresponds to joint movement during each jump trial.

Our objective was to develop a post-processing method to align these two datasets without relying on a trigger signal. Specifically, we:

- Extracted timing metadata from EMG and motion files.
- Developed visual tools to identify signal features such as bursts or spikes that could serve as alignment points.
- Matched EMG recordings with corresponding motion trials for each jump type.
- Organized the workflow into a reusable process for future research or clinical use.

This alignment is a critical step in enabling reliable downstream analysis of biomechanical and neuromuscular variables across age groups, which may ultimately support early detection of risk factors for knee degeneration.

Functional Components & Code Logic

This project consisted of four main functional components written in MATLAB and Python. Each script supported different aspects of the data preprocessing and synchronization process.

1. MATLAB Batch Plotting Script: Implemented in the '***plot_emg_batch.m***' file (included in the submission folder), this script processed a folder of EMG CSV files, identified relevant signal columns for the Vastus Lateralis muscles, and generated high-resolution PNG plots for visual inspection. It was designed to handle variations in column naming, skip invalid files, and support fast batch review across multiple trials. The output figures consistently displayed side-by-side signals for

the left and right muscles, helping to visually identify patterns that could support alignment.

For each file, this script uses *readtable* to load the data while preserving header formatting. Since column names can vary slightly between files, the script checks against multiple known variations for the Vastus Lateralis EMG signals to ensure it captures the correct data. If either the left or right EMG signal columns are missing, the script logs a warning and skips that file rather than stopping the whole batch process. Once valid columns are found, the corresponding time and EMG data are extracted and prepared for plotting.

To generate clear visual comparisons, each figure shows two subplots: one for the right and one for the left muscle. The layout is consistent across all plots and each figure is saved as a high-resolution PNG file using *exportgraphics*. The figure below shows a generated PNG plot for an EMG file. The plot contains two subplots, one for the right Vastus Lateralis muscle and one for the left. Each signal is visualized over the entire duration, providing a clear, side-by-side comparison of the muscle activity.

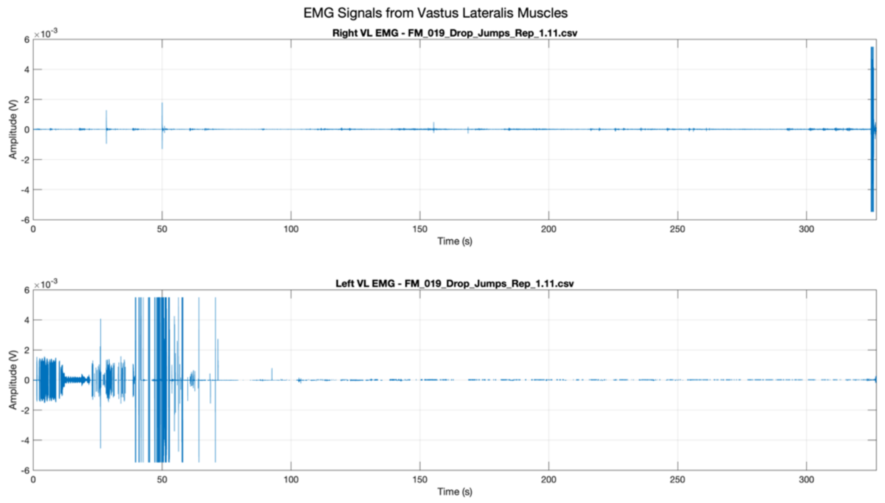


Figure 1: Generated PNG plot for EMG file named FM_019_Drop_Jumps_Rep_1.11.csv

2. <u>MATLAB Live Script for Visual Inspection:</u> This component was implemented in the '***emg_viewer.mlx***' script (included in the project submission folder). It provided an interactive way to examine individual EMG recordings in greater detail. Users were prompted to select a specific EMG CSV file, after which the script automatically generated full-duration plots for both the left and right Vastus Lateralis muscles. In addition to the full view, it also produced zoomed-in plots focused on a defined two-second window, helping to highlight bursts, spikes, or other signal features that could serve as alignment points.

This script begins by prompting the user to select a single CSV file using a file browser. Once the file is selected, the script loads the data using *readtable* while preserving the original headers. Like the first script, it accounts for variations in column names when identifying the EMG signal and time data for the Vastus Lateralis muscles. After loading the data, the script generates a full-length plot of the EMG signals for both the right and left muscles. In addition, the script then creates a second set of plots focused on a zoomed-in time window to support precise inspection. This is done using logical indexing to filter the time and signal vectors.

The figure below shows a window displaying the interactive output of the Live Script. It includes a zoomed-in view of the EMG signal for both the right and left Vastus Lateralis muscles, focusing on a 2-second window.

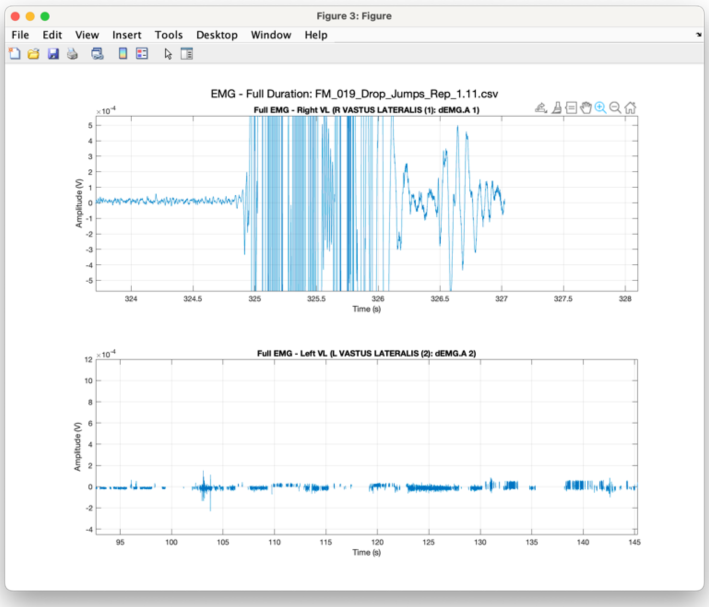
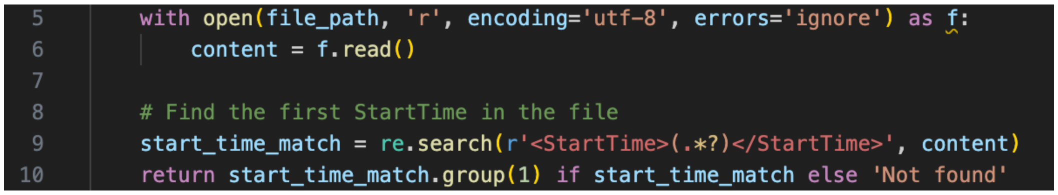


Figure 3: Figure
File Edit View Insert Tools Desktop Window Help
EMG - Full Duration: FM_019_Drop_Jumps_Rep_1.11.csv
Full EMG - Right VL (R VASTUS LATERALIS (1): dEMG.A 1)
Amplitude (V)
Time (s)
Full EMG - Left VL (L VASTUS LATERALIS (2): dEMG.A 2)
Amplitude (V)
Time (s)

Figure 2: Generated Live Plot for EMG file named FM_019_Drop_Jumps_Rep_1.11.csv

3. <u>Python Script for Extracting UTC Timestamps:</u> This functionality was implemented in the '***extract_utc_from_hpf.py***' script (included in the project submission folder). The script processed a folder of .hpf EMG files and extracted the UTC start time embedded in each file's metadata. It used regular expressions to locate the <StartTime> tag within the file content, which followed a loosely structured XML format.

This script starts by scanning all .hpf files in a specific directory and opening each one to read its content. Since the files are not standard XML but still contain identifiable tags, this script uses regular expressions to locate the <StartTime> field.

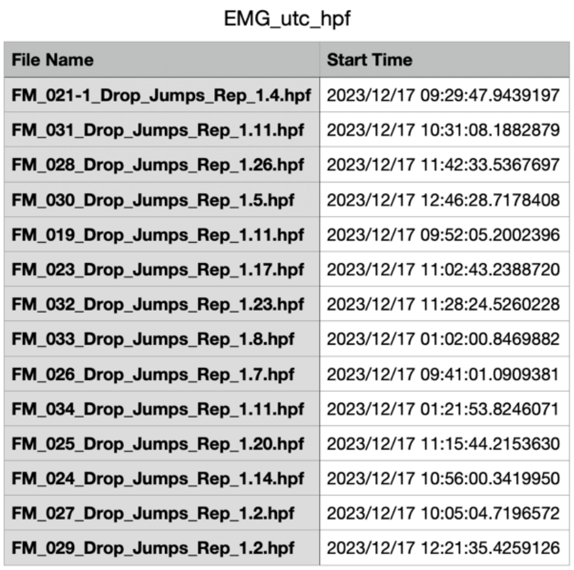


```
 5      with open(file_path, 'r', encoding='utf-8', errors='ignore') as f:
 6          content = f.read()
 7
 8      # Find the first StartTime in the file
 9      start_time_match = re.search(r'<StartTime>(.*?)</StartTime>', content)
10      return start_time_match.group(1) if start_time_match else 'Not found'
```

Figure 3: Lines 5-10 of Python Script for Extracting UTC Timestamps – Scanning for Start Time Field

If a <StartTime> tag is found, the script captures the timestamp and stores it along with the filename. If the tag is missing or malformed, the script skips the file and moves on to the next one. This prevents the program from crashing and ensures it can still extract data from the rest of the files. Once all files have been processed, the script writes the results to a summary CSV file. This file contains two columns: the name of each .hpf file and its corresponding extracted UTC start time.

The figure below shows the summary CSV generated by the Python script. This output provides the extracted UTC start times for each EMG file and is intended to support future alignment with the 3D motion captured recordings.



EMG_utc_hpf

File Name	Start Time
FM_021-1_Drop_Jumps_Rep_1.4.hpf	2023/12/17 09:29:47.9439197
FM_031_Drop_Jumps_Rep_1.11.hpf	2023/12/17 10:31:08.1882879
FM_028_Drop_Jumps_Rep_1.26.hpf	2023/12/17 11:42:33.5367697
FM_030_Drop_Jumps_Rep_1.5.hpf	2023/12/17 12:46:28.7178408
FM_019_Drop_Jumps_Rep_1.11.hpf	2023/12/17 09:52:05.2002396
FM_023_Drop_Jumps_Rep_1.17.hpf	2023/12/17 11:02:43.2388720
FM_032_Drop_Jumps_Rep_1.23.hpf	2023/12/17 11:28:24.5260228
FM_033_Drop_Jumps_Rep_1.8.hpf	2023/12/17 01:02:00.8469882
FM_026_Drop_Jumps_Rep_1.7.hpf	2023/12/17 09:41:01.0909381
FM_034_Drop_Jumps_Rep_1.11.hpf	2023/12/17 01:21:53.8246071
FM_025_Drop_Jumps_Rep_1.20.hpf	2023/12/17 11:15:44.2153630
FM_024_Drop_Jumps_Rep_1.14.hpf	2023/12/17 10:56:00.3419950
FM_027_Drop_Jumps_Rep_1.2.hpf	2023/12/17 10:05:04.7196572
FM_029_Drop_Jumps_Rep_1.2.hpf	2023/12/17 12:21:35.4259126

Figure 4: Generated Summary CSV for all EMG files provided

4. <u>Python Script for Final Alignment:</u> Code was implemented in the '***align_all_subject_trials.py***' script (included in the project submission folder),

this component handled the alignment of EMG and 3D motion capture data using previously extracted UTC timestamps to match each EMG recording with its corresponding motion file. This matching process was based entirely on timing, allowing for accurate pairing even in the absence of a trigger signal.

Once the files were matched, the script split the motion data into individual jumps and aligned them with EMG bursts occurring in the same time window. Each aligned trial was saved as a new file in a structured folder named '*aligned_subject_trials*'. This script completed the synchronization pipeline by producing a clean and organized set of output files, ready for further visualization or analysis.

All four scripts worked together to align EMG and 3D motion capture data without a trigger signal. The output of the full pipeline was reviewed manually to ensure that EMG bursts and motion segments were correctly aligned across trials. While a formal quantitative evaluation was not performed, the resulting files appeared consistent and ready for downstream biomechanical analysis.

Design Choices and Development Methods

We chose to use a combination of MATLAB and Python to take advantage of the strengths of each platform. MATLAB was selected for signal plotting and visual inspection due to its built-in support for matrix operations and high-quality plotting functions, which made it ideal for handling EMG data. Python was used for file parsing and automation tasks because of its flexibility in handling different file formats and ease of scripting for batch operations. Development was done using MATLAB R2023b and Python 3.11, and all scripts were organized into clearly labeled files and folders to support collaboration and reusability. While we did not use formal version control, we maintained consistent naming conventions and tested scripts individually before integrating them into the full workflow.

Team Contributions

- Created MATLAB batch plotting script for EMG visualization: Juliana
- Developed MATLAB live script for detailed EMG inspection: Juliana
- Wrote Python script to extract UTC timestamps from EMG .hpf files: Juliana
- Implemented Python script to align EMG and motion data: Eswar, Vani
- Standardized and cleaned formatting of motion capture .csv files: Eswar, Vani
- Evaluated aligned output files for consistency and reliability: Eswar, Vani

References

1. Hunter, D. J., & Bierma-Zeinstra, S. (2019). Osteoarthritis. *The Lancet, 393*(10182), 1745–1759. https://doi.org/10.1016/S0140-6736(19)30417-9

2. Padua, D. A., DiStefano, L. J., Beutler, A. I., de la Motte, S. J., & DiStefano, M. J. (2015). The landing error scoring system as a screening tool for an anterior cruciate ligament injury–prevention program in elite-youth soccer athletes. *Journal of Athletic Training, 50*(6), 589–595. https://doi.org/10.4085/1062-6050-50.1.10

3. MathWorks. (2023). Readtable: Read tabular data from file. MATLAB Documentation. https://www.mathworks.com/help/matlab/ref/readtable.html

4. Python Software Foundation. (2023). re — Regular expression operations. https://docs.python.org/3/library/re.html