Decomposition of Muscle Activity for Sensorimotor Neuroscience
Final Report

Daniel King, Jasmine Ortega, Rada Rudyak, Rowan Sivanandam

Mentor: Dr. Alexi Rodríguez-Arelis
Partner: Dr. Jean-Sébastien Blouin, Sensorimotor Physiology Laboratory, UBC

2022-06-29

Contents

Executive Summary

The decomposition of electromyography (EMG) signals into individual motor units is a critical technique for understanding the relationship between the nervous and muscular systems. The UBC Sensorimotor Physiology Laboratory's research currently decomposes raw EMG signals using free software, `OTBioLab+`, whose code is based on a paper from Negro et al. (2016). As this software has several shortcomings, we have replicated the decomposition algorithm and packaged it in an open-source Python package called `EMGdecomPy`. Additionally, we have included a reproducible tool to visualize the decomposed output of the algorithm.

The decomposition algorithm described by Negro et al. (2016) was broken up into individual functions. These functions were individually tested and extensively documented for the sake of transparency. Finally, per the partner's request, `EMGdecomPy` was made flexible enough to run experiments of arbitrary time duration. Preliminary qualitative results show that `EMGdecomPy` identifies 3 out of 5 of the same motor units as Hug et al. (2021) when run on their raw EMG data obtained from the *Gastrocnemius lateralis* muscle with 10% contraction intensity.

While the sequential steps to decompose raw EMG signals were given by Negro et al. (2016), there was room for interpretation and improvement in the implementation of each step. Future recommendations for improvement include decreasing the algorithm's run-time through optimization, increasing decomposition accuracy through domain knowledge, and implementing a relearning feature into the algorithm.

Introduction

The Sensorimotor Physiology Lab at the UBC School of Kinesiology studies the nervous system's role in muscle movement through understanding the neurological-muscular mechanisms involved in human motion, specifically the role that these mechanisms play in humans' ability to balance. By studying the link between the nervous system and movement, researchers can further understand the effects of neuromuscular damage.

Findings can be used to establish preventative measures and create more effective treatments for those with chronic neuromuscular disorders and age-related neural degeneration (Purves 2018).

The brain, spinal cord, and attached nerves form the nervous system, which is responsible for initiating and propagating electrical movement signals to the correct muscles. These electrical bursts, known as action potential, are initially fired off from the brain. Action potential propagates throughout the body via neurons, which are cells specialized in transmitting electrical signals across long distances.

Motor neurons are a specific subclass of neurons that create a junction between the central nervous system and the muscular system. Motor neurons attach to fibres in the muscle, forming an entity known as a motor unit (MU), as seen in **figure 1**. When struck by action potential, the MU generates motor unit action potential (MUAP), a burst of electrical activity specific to the MU that produced it. The MUAP signal causes the muscle fibres attached to the MU to contract (Purves 2018).

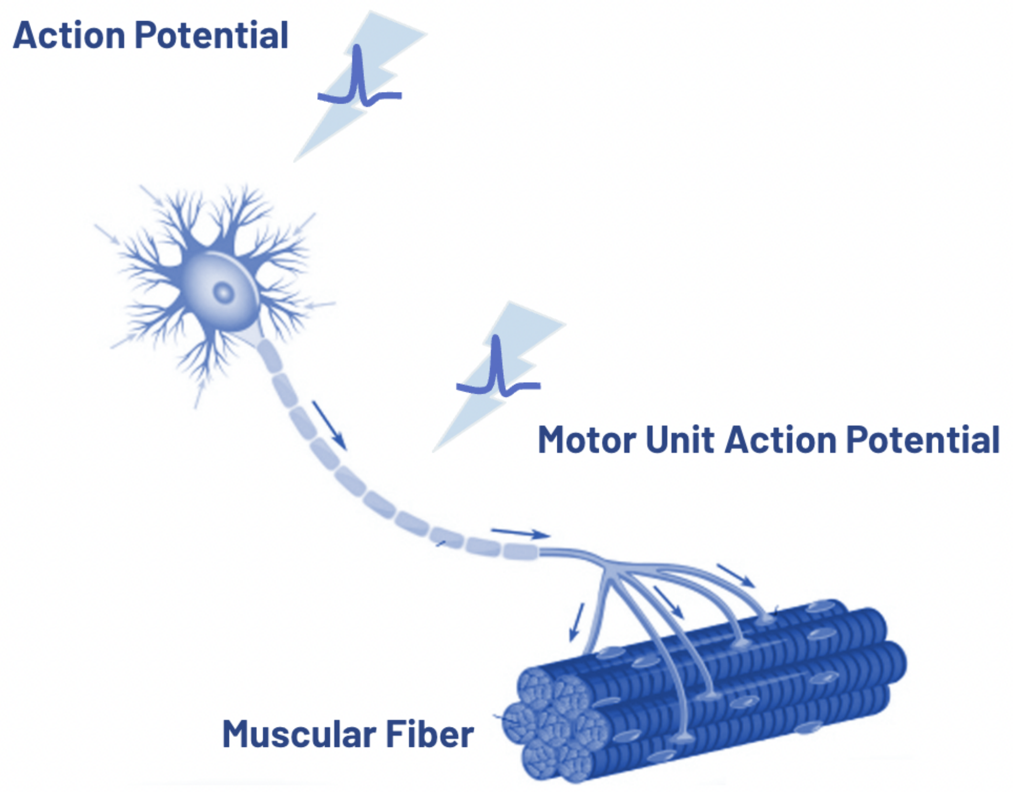


Figure 1: Diagram of a single motor unit. Modified from McLaughlin (2020).

The activation of several MUs (and their accompanying MUAPs) is required to move a single muscle. Therefore, the number of MUs involved in an isolated muscle movement can be determined by measuring the net electrical charge of a single muscle.

The partner uses non-invasive electromyography (EMG) to measure the net firing of MUAPs across a single muscle, seen in **figure 2a** as opposed to invasive EMG, seen in **figure 2b**. Surface electrodes are placed on the skin and voltage is measured as participants flex and relax a muscle. The partner uses a grid of 64 surface electrodes, as seen in **figure 3**, allowing the collection of 64 streams of MUAP data across a single muscle. The raw signal collected by EMG is the result of many MUAP peaks constructively and destructively interfering with each other.

A blind source separation algorithm can decompose raw EMG signals into several individual electrical signals

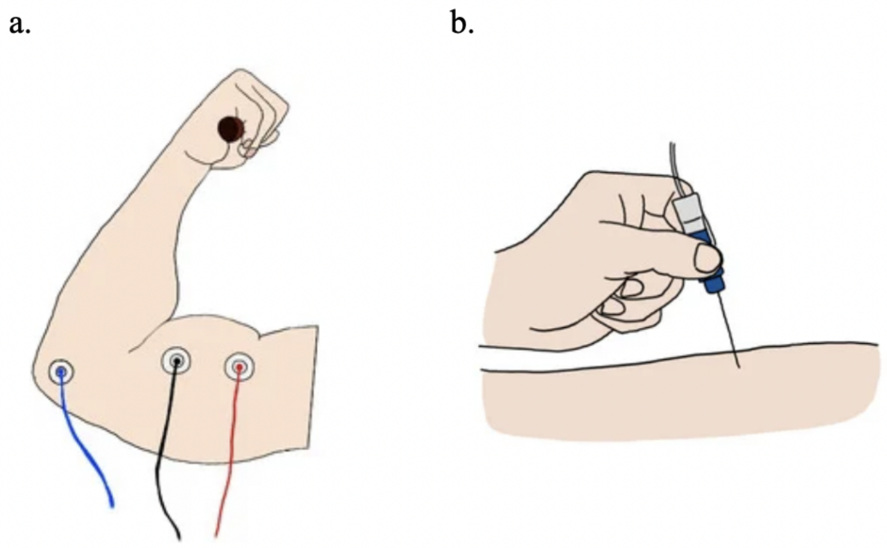


Figure 2: Obtaining EMG with a. surface electrodes and b. needle electrodes. Invasive EMG using needle electrodes is the more commonly used technique, as it allows access to deeper muscle groups and higher muscle region selectivity which decreases noise. In comparison, non-invasive EMG using surface electrodes are less selective and can only sample superficial muscles, as they sit on the skin far away from the muscle fibres ("EMG Decomposition Tutorial" 2006). However, non-invasive EMG is a more flexible and easier-to-apply technique with higher long-term stability than invasive EMG, which are desirable attributes to the partner (Farina and Holobar 2015). Figure modified from Nam, Cha, and Park (2021).

that can be ascribed to different MUs. The partner currently decomposes EMG signals using free software from OT Bioelettronica SRL (2022) called `OTBioLab+`. This software's graphical user interface (GUI) can be seen in **figure 4**. `OTBioLab+` determines the individual MUAP spike trains using a closed-source algorithm based on a paper published by Negro et al. (2016).

As is, `OTBioLab+` is an unideal solution for decomposing EMG signals for the partner's needs. The creation of a custom Python package, `EMGdecomPy`, can address the issues with `OTBioLab+` and contains custom functionalities specific to the partner's use. `OTBioLab+` is closed-source, which obscures how the partner's experimental results are derived. By making `EMGdecomPy` open-source, the algorithm's code can be easily inspected, circumventing this problem. Additionally, `OTBioLab+` has a 100-second time limit on experiments, which hinders the partner as they run experiments up to 5 minutes in length. In `EMGdecomPy`, there is no hard-coded time limit on experiments that can be run.

The transparency of `EMGdecomPy`'s code and the open-source licensing support other parties in expanding on and improving the existing functionalities. While made for the partner, we hope that `EMGdecomPy` will be a useful package for other researchers to explore the mechanisms underlying the brain's interactions with the body's muscles.

Data Science Methods

The partner needed to replicate the blind source separation algorithm used in Negro et al. (2016) to decompose raw EMG signals into their constituent MU spike trains (MUSTs). This algorithm provides advantages to other blind source separation algorithms in that it is an experimentally validated approach that allows for the decomposition of multi-channel invasive EMG and non-invasive EMG data.

Essentially, the blind source separation algorithm is an unsupervised machine learning model that iteratively extracts separation vectors from the data. Separation vectors are vectors that, when applied to the pre-processed data, separate a single MUST from unwanted noise and other MUSTs, as seen in **figure 5**. An

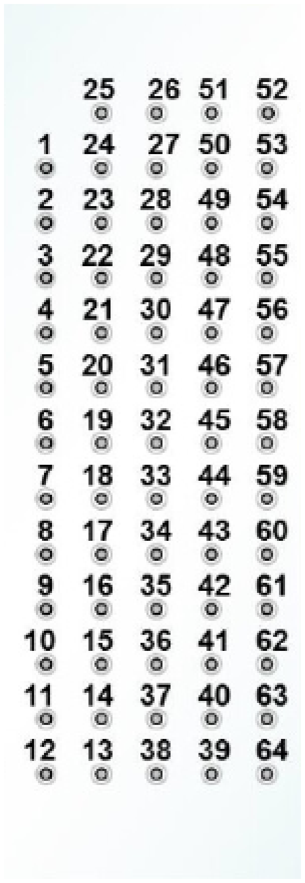
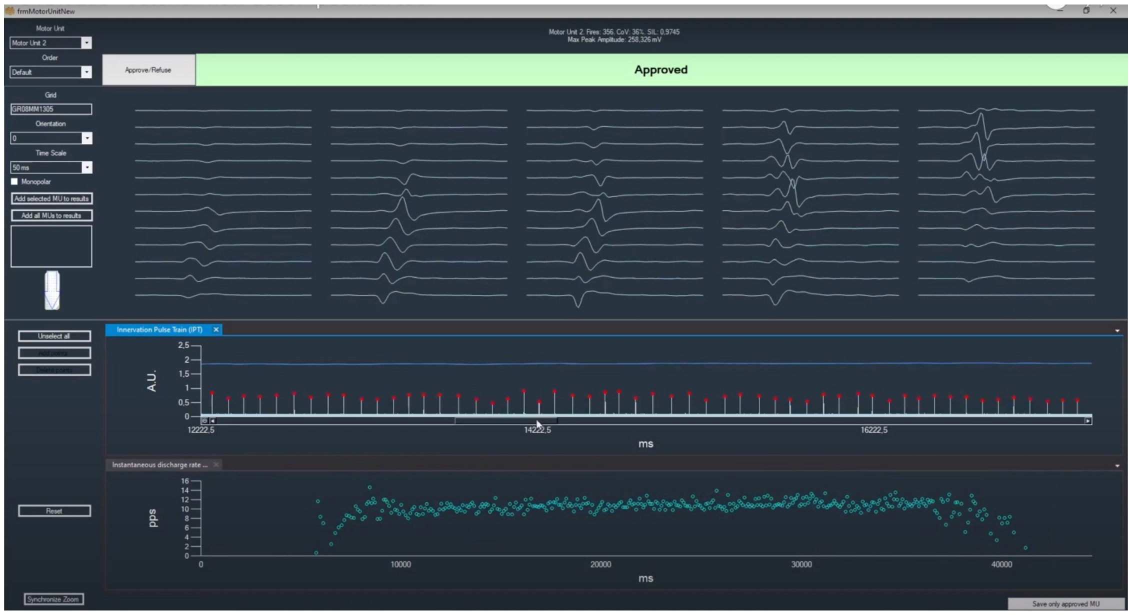


Figure 3: The 64 channel template used by the Sensorimotor Physiology Lab.



Motor Unit 2. Fires: 356. CoV: 36%. SIL: 0,9745
Max Peak Amplitude: 258,326 mV

Motor Unit: Motor Unit 2
Order: Default
Grid: GR08MM1305
Orientation: 0
Time Scale: 50 ms
Monopolar

Approved

Figure 4: Graphical user interface of the `OTBioLab+` software (OT Bioelettronica SRL 2022).

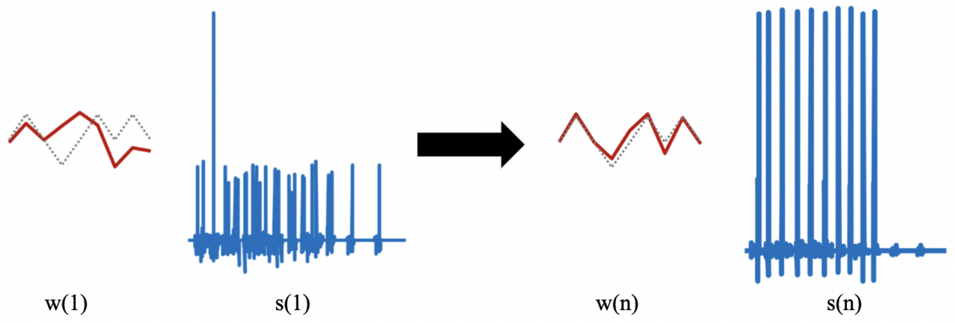


Figure 5: Separation vector extraction process performed by the blind source separation algorithm. w(1) is the separation vector at the first iteration of the latent component analysis step, where the red line is the current estimation and the gray line is the theoretical ideal separation vector. s(1) is the separation vector's corresponding MUST. w(n) is the estimated separation vector at the final iteration of the latent component analysis step, and s(n) is its corresponding MUST. Figure modified from Negro et al. (2016).

MUST contains the firing times of a single MU, and it is the mixture of these that compose the raw EMG signal.

Separation vectors are extracted through broadly three steps:

1. The data is pre-processed.
2. Latent component analysis (LCA) is performed to estimate a separation vector.
3. The refinement process is used to increase the quality of the estimated separation vector.

Steps two and three are repeated for a predetermined amount of iterations of the overall blind source separation algorithm.

Step 1: Pre-Processing.

The first step of the algorithm is data pre-processing, which consists of four sub-steps (**figure 6**):

a. First, the data is *band-pass filtered.* This removes noise from the data and increases the number of identified separation vectors.

b. Then, the data is *centred* by subtracting the mean of each channel per channel. This process is essential for LCA to extract all of the separation vectors.

c. Then each channel is *extended* by a certain number of time-delayed versions of that channel. This process converts the mixture of signals from a convolutive mixture to a linear instantaneous mixture, which is mathematically necessary for LCA to work (Negro et al. 2016; Weenink 2012).

d. Finally, the data is *whitened*, so that the covariance matrix of the extended channels is equal to the identity matrix. This improves performance by computationally simplifying the convergence of the LCA step (Hyvärinen, Karhunen, and Oja 2001).

Step 2: Latent Component Analysis.

The pre-processed data then goes through the LCA step. The LCA step is based on independent component analysis, where the separation vectors are obtained by maximizing their statistical independence from each other. However, since the MUSTs are extended along with the observations, they cannot be fully independent. This is why LCA extracts separation vectors by maximizing sparsity instead of independence. Intuitively, this is done because a singular MUST will be more sparse than the combination of multiple MUSTs (Negro et al. 2016). As seen in **figure 5**, the MUST is much more sparse after the final iteration of LCA in comparison to the first iteration.

a. First, the separation vector is *initialized* by using a time instance of high activity in the whitened data, as these time instances are likely to correspond to multiple MU firings.

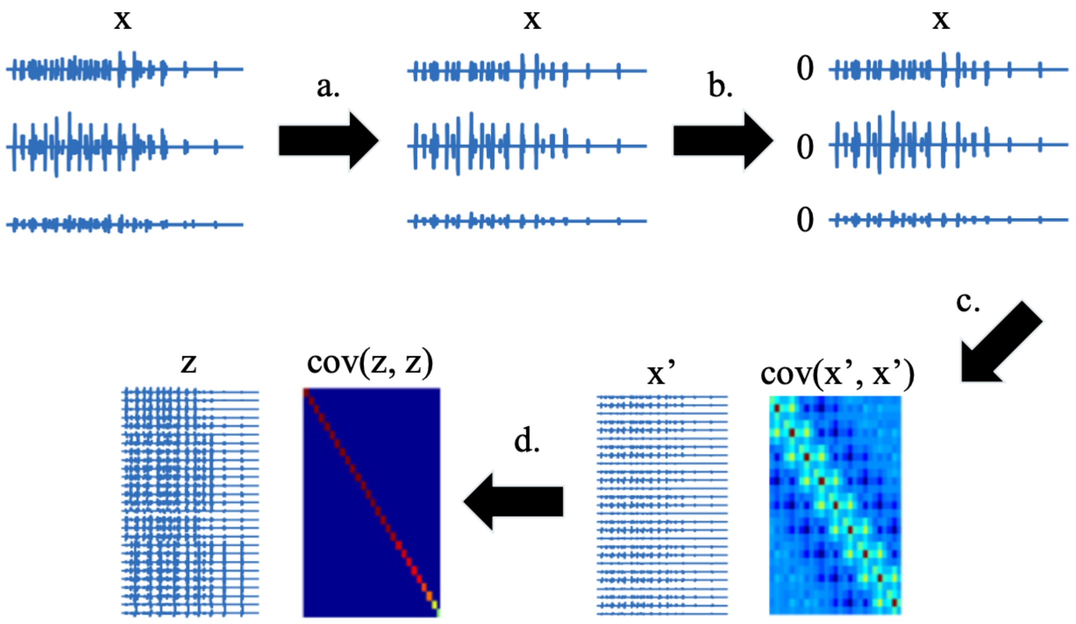


Figure 6: The data pre-processing pipeline. a. The original raw data, x, is band-pass filtered. b. The data of each channel is centred around 0. c. The data is extended to form x', with covariance matrix, cov(x', x'). d. The data is whitened to form z, with covariance matrix, cov(z, z). Figure modified from Negro et al. (2016).

b. Using a contrast function that measures the sparsity of the MUST, the estimated separation vector is *iteratively updated*. In each iteration, the estimated separation vectors are orthogonalized against previously accepted separation vectors and normalized, to increase the number of extracted unique MUSTs.

c. The LCA step converges. This happens when the separation vector approximately no longer changes between iterations.

Step 3: Refinement.

After the separation vector is extracted, it goes through the refinement step. This is done because LCA may converge to unreliable estimates and through refinement, the quality of the estimate is increased. The refinement step is an iterative algorithm that maximizes the regularity of the MUST under the assumption that singular MUs fire off action potentials at a much more regular rate than combinations of MUs (Negro et al. 2016).

a. First, the MUST is estimated by applying the separation vector to the pre-processed data.

b. Then, the firing times are determined by applying the peak-finding algorithm from Virtanen et al. (2020). Of these firing times, the instances that correspond to small peaks in the spike train are separated away from the large peaks using the KMeans algorithm from Pedregosa et al. (2011). The firing times corresponding to small peaks are discarded as they likely correspond to the firing occurrence of more than one MU (Negro et al. 2016). The information from the accepted firing times is used to update the separation vector.

c. The iterative refinement process converges once the coefficient of variation of the time between firings increases.

$$\text{silhouette score} = \frac{b-a}{max(a,b)} \tag{1}$$

$$a = \text{inter-cluster sum of point-to-centroid distances}$$
$$b = \text{intra-cluster sum of point-to-centroid distances}$$

$$\text{pulse-to-noise ratio} = 10 \log_{10} \left(\frac{P_{\text{Signal}}}{P_{\text{Noise}}} \right) \tag{2}$$

$$P_{\text{Signal}} = \text{Power of signal}$$
$$P_{\text{Noise}} = \text{Power of noise}$$

Once the refinement process is done, the refined separation vector is accepted based on a user-defined threshold of either the silhouette score between the signal and the noise or the pulse-to-noise ratio. The silhouette score is defined in (1) and is calculated using the signal and noise clusters in the MUSTs (Negro et al. 2016). The pulse-to-noise ratio is defined in (2). The accepted separation vectors correspond to the MUs that the blind source separation algorithm extracts from the raw signal.

A further improvement to the algorithm that we did not have time to implement would be a re-learning feature. The user would run the algorithm on a sample of the data, and then identify inaccurate firing times (false positives) based on physiological limits of MU firing rates. The algorithm would use this information to no longer make similar mistakes in the rest of the decomposition. Implementing this feature would be quite complex because it is unclear how this would be implemented programmatically.

The stakeholders affected by our blind source separation algorithm are researchers and those that would be affected by their research. This is why our algorithm must work properly so that researchers' results are accurate and do not affect the general public adversely down the line. For example, if someone uses `EMGdecomPy` and obtains inaccurate results which are used to inform a neuromuscular diagnosis, it could greatly affect someone's life. Periodically, the results of our algorithm should be compared to others to obtain a second opinion on the decomposition of the EMG signal.

The data provided by Hug et al. (2021) contains EMGs and their decomposition results from many different muscles at different voluntary muscle contraction intensities (100% being the most intensely the subject can contract their muscle). We have not had the chance to thoroughly validate our algorithm using this data, as the debugging process took a great deal of time. We have only received qualitative results, obtained by visually comparing MUAP shapes identified by `EMGdecomPy` and those Hug et al. (2021) identified using `DEMUSE`, a commercial software created by Holobar (2016). There are concerns with this approach as `DEMUSE` uses a similar but different algorithm than Negro et al. (2016). `DEMUSE` is a highly used software in comparison to `OTBioLab+`, and therefore the partner wishes to compare our results to theirs.

$$\text{rate of agreement} = \frac{c_j}{c_j + A_j + B_j} \tag{3}$$

$$c_j = \text{Number of discharges (MU firings) for the j-th MU, identified by both algorithms}$$
$$A_j = \text{Number of discharges for the j-th MU, identified by one of the algorithms}$$
$$B_j = \text{Number of discharges for the j-th MU, identified by the other algorithm}$$

In the future, we would like to quantitatively validate our results on more muscle groups, using the metric recommended by Negro et al. (2016), the rate of agreement (RoA), defined in (3). RoA theoretically ranges from 0 to 1. The closer the RoA is to 1 the more the decomposition provided by `EMGdecomPy` matches the one provided by Hug et al. (2021) for the j-th MU. Therefore getting a high RoA would mean that `EMGdecomPy` is approximately as accurate at decomposing raw EMG signals as the `DEMUSE` software for the Hug et al. (2021) datasets.

Data Product and Results

Our final data product consists of two elements. The first element is the Python package, `EMGdecomPy`, and its associated GitHub repository containing the replicated Negro et al. (2016) blind source separation algorithm. The second element is a Jupyter notebook containing a workflow leading up to an interactive dashboard visualizing the decomposition results.

Creating a Python package allows anyone who would like to decompose EMG signals into constituent motor units to simply download the package off of `pip` and run the `decomposition` function with the desired hyper-parameters on their data. Using a Python package to deliver our blind source separation algorithm not only increases accessibility but also the readability of the code by breaking the blind source separation algorithm into many different functions with informative docstrings. By giving the partner access to the GitHub repository for `EMGdecomPy`, we allow them to view the source code and even edit it if they desire to add more functionality in the future. These attributes of our data product satisfy the partner's desires to replicate the blind source separation algorithm from Negro et al. (2016) without experimental duration limits, while providing them with easy-to-understand open-source code, allowing them to understand how their experimental results are derived.

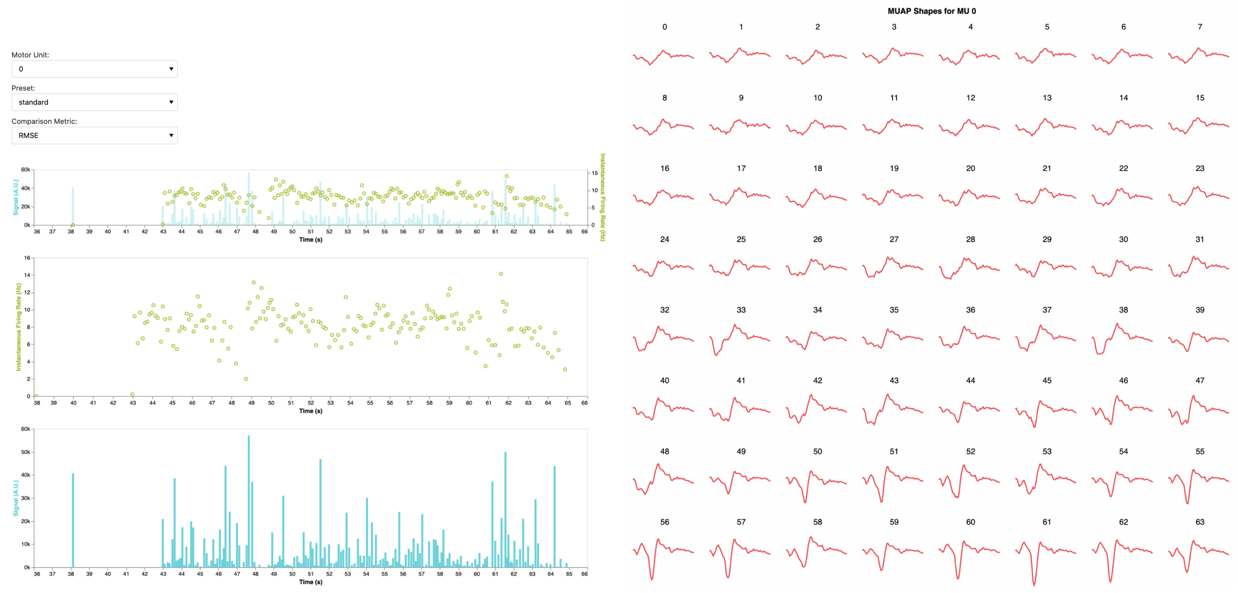

Motor Unit: 0
Preset: standard
Comparison Metric: RMSE

Figure 7: The graphical user interface of the EMGdecomPy interactive dashboard. On the top-left is the widgets controlling which motor unit is visualized, which channel arrangement preset is being used, and which comparison metric should be used to determine the peak contributions to each MUAP shape. The bottom-left contains an overlay of the firing rate and time plots, and the separated firing rate and time plots, respectively. On the right is the average MUAP shapes obtained from each channel.

The Jupyter notebook provides the partner with an interface in which to use `EMGdecomPy`. This notebook further facilitates our package's ease of use, as loading the data, running the algorithm, and visualizing the results, will be laid out in a reproducible step-by-step process. The GUI of the interactive dashboard provided in this notebook can be seen in **figure 7**.

As mentioned above, we have not had the chance to thoroughly validate our algorithm. However, preliminary results look promising, as 3 out of 5 of the MUAP shapes identified by `EMGdecomPy` were also identified by Hug et al. (2021) for the *Gastrocnemius lateralis* muscle with 10% contraction intensity.

Conclusions and Recommendations

Muscle movement is generated when the brain sends electrical activity known as action potentials to an MU. The MU fires a form of action potential called a MUAP, which causes the attached muscle fibres to contract. A technique known as EMG uses electrodes to measure the electrical activity across a single muscle. The raw EMG signal is the combination of many MUAPs and can be decomposed using a blind source separation algorithm into separate electrical signals that correspond to different MUs.

Our final data product presented is an open-source Python package, `EMGdecomPy`, containing two elements. The first element is an algorithm based on work by Negro et al. (2016). The algorithm was developed using data from Hug et al. (2021). The second component is a Jupyter notebook that provides a template for users to interactively view the decomposed output of the algorithm.

Although `EMGdecomPy` has not been thoroughly validated yet, preliminary qualitative results show that the package identifies 3 out of 5 of the same motor units as the commercial `DEMUSE` software when run on the same *Gastrocnemius lateralis* muscle with 10% contraction intensity data obtained from Hug et al. (2021). Recommendations for future work include expanding the qualitative validation to other muscle groups and contraction intensities provided by Hug et al. (2021) and performing quantitative analysis using the RoA metric, as well as increasing the software's accuracy through domain knowledge and implementing the re-learning feature.

References

"EMG Decomposition Tutorial." 2006. *EMG Decomposition Tutorial.* http://www.emglab.net/emglab/Tutorials/EMGDECOMP.html.

Farina, Dario, and Ales Holobar. 2015. "Human-Machine Interfacing by Decoding the Surface Electromyogram [Life Sciences]." *IEEE Signal Processing Magazine* 32 (1): 115–20. https://doi.org/10.1109/msp.2014.2359242.

Holobar, Ales. 2016. *DEMUSE Tool.* https://demuse.feri.um.si/.

Hug, Francois, Simon avrillon, Alessandro Del Vecchio, Andrea Casolo, Jaime Ibáñez, Stefano Nuccio, Julien Rossato, Aleš Holobar, and Dario Farina. 2021. "Analysis of motor unit spike trains estimated from high-density surface electromyography is highly reliable across operators," February. https://doi.org/10.6084/m9.figshare.13695937.v1.

Hyvärinen, Aapo, Juha Karhunen, and Erkki Oja. 2001. "What Is Independent Component Analysis?" In *Independent Component Analysis*, 158–61. Wiley.

McLaughlin, Katy. 2020. "Motor Neuron - the Definitive Guide." *Biology Dictionary.* https://biologydictionary.net/motor-neuron/.

Nam, Dahyun, Jae Min Cha, and Kiwon Park. 2021. "Next-Generation Wearable Biosensors Developed with Flexible Bio-Chips." *Micromachines* 12 (1): 64. https://doi.org/10.3390/mi12010064.

Negro, Francesco, Silvia Muceli, Anna Margherita Castronovo, Ales Holobar, and Dario Farina. 2016. "Multi-Channel Intramuscular and Surface EMG Decomposition by Convolutive Blind Source Separation." *Journal of Neural Engineering* 13 (2): 026027. https://doi.org/10.1088/1741-2560/13/2/026027.

OT Bioelettronica SRL. 2022. "Decomponi." *OT Bioelettronica.* https://otbioelettronica.it/component/sppagebuilder/?view=page&id=158.

Pedregosa, F., G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, et al. 2011. "Scikit-Learn: Machine Learning in Python." *Journal of Machine Learning Research* 12: 2825–30.

Purves, Dale. 2018. "Lower Motor Neuron Circuits and Motor Control." In *Neuroscience*, 357–80. Sinauer Associates, an imprint of Oxford University Press.

Virtanen, Pauli, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, et al. 2020. "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python." *Nature Methods* 17: 261–72. https://doi.org/10.1038/s41592-019-0686-2.

Weenink, David. 2012. "Blind Source Separation." *Blind Source Separation.* https://www.fon.hum.uva.nl/praat/manual/blind_source_separation.html.