Decomposition of Muscle Activity for Sensorimotor Neuroscience Final Report

Daniel King, Jasmine Ortega, Rada Rudyak, Rowan Sivanandam

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

2022-06-21

Contents

Executive Summary

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

`EMGdecomPy` satisfies the needs of the Sensorimotor Physiology Lab's research. The decomposition algorithm was adapted from the pseudocode as described by Negro et al. (2016). The phases of decomposition (pre-processing, separation, and refinement, visualization) were broken up into individual functions. These functions were individually tested and extensively documented for the sake of transparency. Finally, per the partner's request, the 100 second time limit was removed from the software. While the sequential steps to decompose raw EMG signals were given, there was room for interpretation and improvement in the implementation of each step. As `EMGdecomPy`'s open-source licensing encourages other parties to expand on the package, future recommendations for improvement include improving code speed through optimization, improving decomposition accuracy through domain knowledge, as well as implementing the aforementioned algorithm relearning feature.

Introduction

The sensorimotor system, or the bodily process that links physical movement to outside sensory information, is a key component of human motion (Forbes, Little, and Candow 2012). A healthy sensorimotor system is critical, as it is required to communicate, interact, and maneuver throughout the world. Muscle movement, a key component of motion and the motor system's purpose, is the carefully coordinated result between the nervous system and muscular system.

The Sensorimotor Physiology Lab at the UBC School of Kinesiology studies the nervous system's role in muscle movement. Under Dr. Jean Sebastien Blouin, specific research objectives include understanding the sensorimotor system's role in human balance, as well as expanding on the known neurological-muscular mechanisms involved in human motion. 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, as well as create more effective therapeutic 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, the schema responsible for initiating and propagating electrical movement signals to the correct muscles (Purves 2018). These electrical bursts, known as action potential are initially fired off from the brain (Purves 2018). Action potential propagates throughout the body via neurons, which are cells specialized in transmitting electrical signals across long distances (Purves 2018).

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

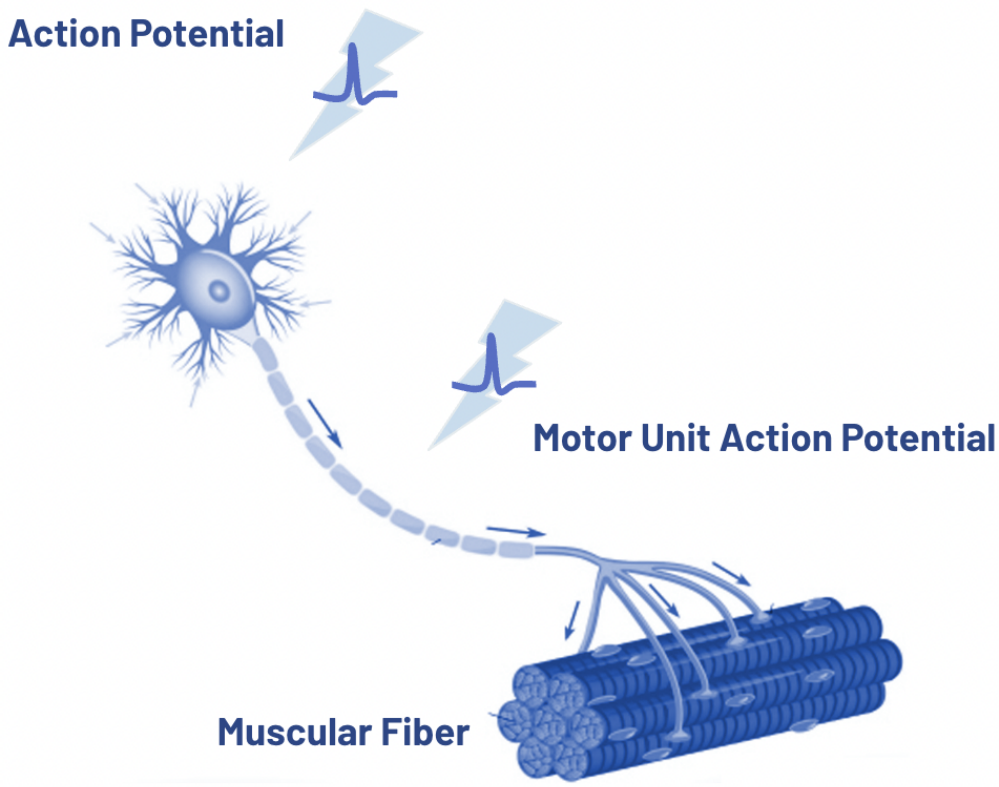

Figure 1: Diagram of a single motor unit.

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

Non-invasive electromyography (EMG), a technique employed by the Sensorimotor Physiology Laboratory measures the net firing of motor unit action potentials across a single muscle. Surface electrodes are placed

on the skin and voltage is measured as participants flex and relax a muscle ("EMG Decomposition Tutorial" 2006). The capstone partner uses a grid of 64 surface electrodes, as seen in **figure 2**, allowing the collection of 64 streams of MUAP data across a single muscle. The raw signal collected by EMG is the result of many motor unit action potential peaks constructively and destructively interfering with each other.

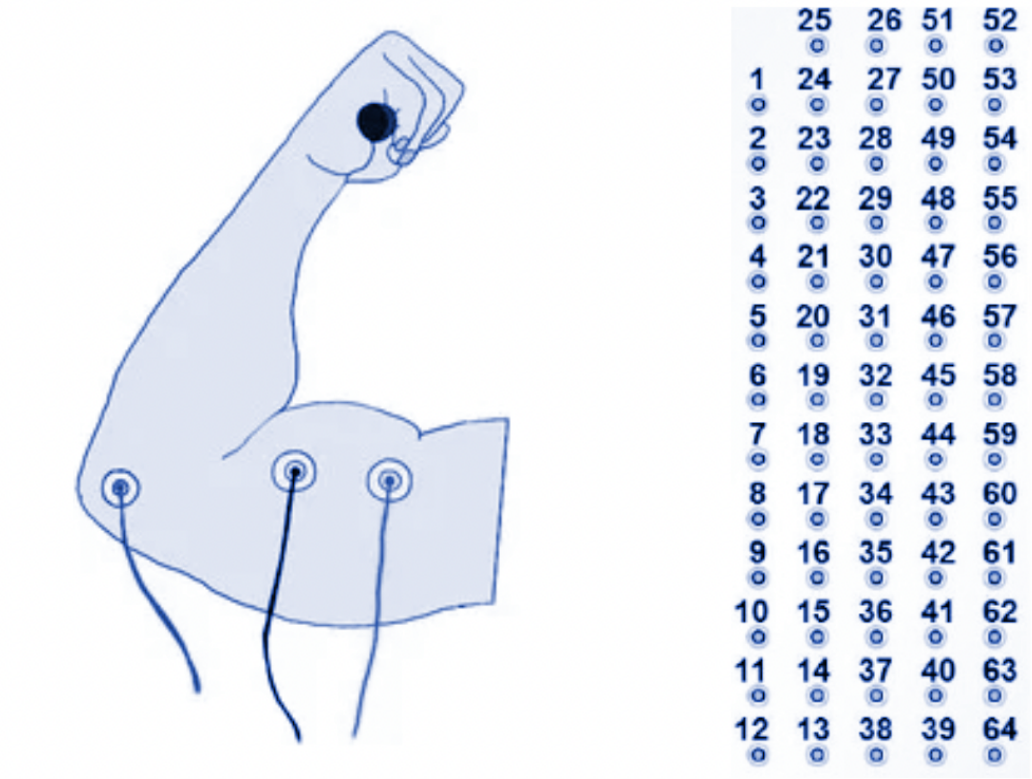

Figure 2: Diagram of surface EMG with 3 electrodes, and the 64 channel template used by the Sensorimotor Physiology Lab.

A blind source separation algorithm can decompose raw EMG signals into several individual electrical signals that can be ascribed to singular muscle units. The Sensorimotor Physiology Laboratory currently decomposes EMG signals using a free software from OT Bioelettronica, the graphical user interface (GUI) of which can be seen in **figure 3**. This software determines the individual motor unit action potential spike trains using a closed-source algorithm based off a paper published by Negro et al. (2016).

The decomposition of EMG signals is a critical technique for the Sensorimotor Physiology Laboratory's research. Decomposition allows researchers to gain a deeper understanding of which motor units are responsible for which parts of movement in a muscle. As is, the OT Bioelettronica product is a flawed solution for decomposing EMG signals for the Sensorimotor Physiology Laboratory.

`EMGdecomPy` is a Python package that addresses two issues with the OT Bioelettronica software, as well as builds some custom functionalities for the Sensorimotor Physiology Laboratory's use.

1. `EMGdecomPy` is open-source, meaning that the code is publicly hosted on GitHub and accessible to the general public. This is a desirable attribute for the capstone partner because it allows the algorithms' code to be easily inspected, which is ideal for troubleshooting results and publishing research. The GitHub repo contains thoroughly documented and tested functions that make up the decomposition steps described in the Negro et al. (2016) pseudocode.

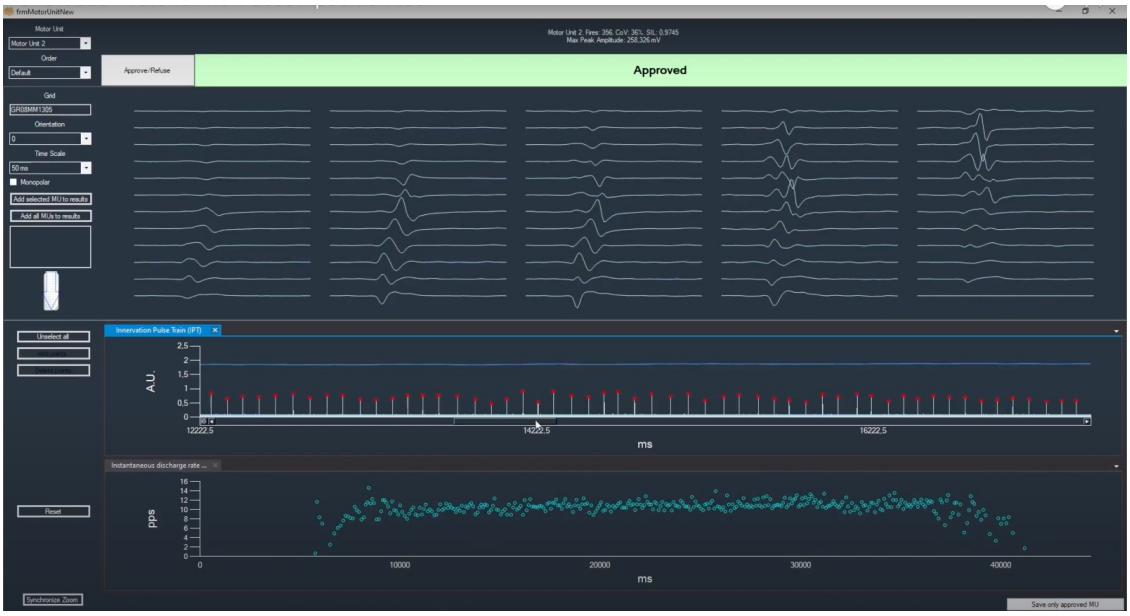

Figure 3: Graphical user interface of the OT Biolettronica software.

2. As the decomposition functions were coded from scratch, the hard-coded experimental duration of 100 seconds present in the OT Bioelettronica software was omitted The Sensorimotor Physiology Laboratory runs experiments up to 5 minutes in length, so this was a crucial characteristic to include in `EMGdecomPy`.
3. A visualization tool was an additional component wanted by the Sensorimotor Physiology Lab. Thus, `EMGdecomPy` contains visualization functions to view and interact with decomposed data. This is a useful feature for visually inspecting the identified motor units and identifying false positive MUAPs.
4. Finally, a Jupyter notebook demonstrating how to use `EMGdecomPy` is included in the GitHub repo. This notebook outlines how to decompose and visualize data using the package's functions. This notebook is a great starting point for first-time users.

The transparency of `EMGdecomPy`'s code and the open-source licensing supports other parties expanding on and improving the existing functionalities. While made for the Sensorimotor Physiology Laboratory, it is our hope that `EMGdecomPy` will be a useful package for other researchers to explore the mechanisms underlying the brain's interactions with our muscles.

Data Science Methods

The partner wants us to replicate the blind source separation algorithm used in Negro et al. (2016) to decompose raw EMG signals into their constituent motor unit spike trains. This algorithm provides advantages to other blind source separation algorithms in that it is an experimentally validated algorithm 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 motor unit spike train from unwanted noise and other motor unit spike trains. A motor unit spike train contains the firing times of a single motor unit, 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.

The first step of the algorithm is data pre-processing. Pre-processing occurs in a four step process. First the data is band-pass filtered to remove noise and increase the amount of motor units identified. Then each channel is centred by subtracting the mean of each channel, is necessary for LCA to extract all of the separation vectors. Then each channel is extended by a certain number of time-delayed versions of that channel, to convert the mixture of signals from a convolutive mixture to a linear instantaneous mixutre, which is necessary for LCA to work (Negro et al. 2016; Weenink 2012). Finally, the data is whitened, so that the covariance matrix of the channels and their extensions is equal to the identity matrix, which computationally simplifies and speeds up the convergence of the LCA step (Hyvärinen, Karhunen, and Oja 2001).

The pre-processed data then goes through the LCA step. The LCA step is based off of independent component analysis, where the separation vectors would be obtained by maximizing their statistical independence from each other. However, since the motor unit spike trains are extended along with the observations, they cannot be fully independent from each other. This is why LCA extracts separation vectors by maximizing sparsity instead of independence. Intuitively, this is done because a singular motor unit spike train will always be more sparse than the combination of multiple motor unit spike trains (Negro et al. 2016). 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 motor unit firings. Using a contrast function that measures sparsity of the motor unit spike train, 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 unique motor unit spike trains that are extracted. The LCA step converges when the separation vector no longer changes, within a tolerance.

After the separation vector is extracted, it goes through the refinement step. This is done because the latent component analysis 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 motor unit spike train. This process is carried out under the assumption that singular motor units will fire off action potentials at a much more regular rate than combinations of motor units (Negro et al. 2016). First, the motor unit spike train is estimated by applying the separation vector to the pre-processed data. 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 motor unit (Negro et al. 2016). The information from the accepted firing times are used to update the separation vector. The iterative refinement process converges once the coefficient of variation of the time between firings increases, which in other words means the regularity decreases.

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 accepted separation vectors correspond to the motor units 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 based on that sample identify firing times that are inaccurate. Then they would run the algorithm on the rest of the data and based on the firing times identified as inaccurate, the algorithm would 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 done. One idea was that we somehow change the initialization of the separation vectors so that they no longer identify the false firing times when applied to the pre-processed data. However, since the separation vector changes throughout the LCA and refinement processes, it would be hard to control the effect that it actually has on the estimated firing times. Another approach would be to influence the KMeans algorithm so that the threshold for the small peaks cluster includes the peaks that were as false firings, in the hopes

that future peaks of similar size are also false firings. The downside to this approach would be that we may increase the amount of incorrect identifications of large peaks as small peaks, which are discarded.

The stakeholders affected by our blind source separation algorithm are researchers and those that would be affected by their research. This is why it is important our algorithm works 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, and these results are used to inform a neuromuscular diagnosis down the line, 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. We have not had the chance to thoroughly validate our algorithm, 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 the `DEMUSE` tool. There are concerns with this approach as the `DEMUSE` software uses a similar but different algorithm than Negro et al. (2016). The `DEMUSE` software is a highly validated software in comparison to `OT Bioelettronica`, and therefore the partner wishes to compare our results to theirs.

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 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 which means they will understand how their experimental results are derived.

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 in the data, running the algorithm, and visualizing the data, will be laid out in a step-by-step process. Essentially, the Jupyter notebook will eliminate all confusion in how to use the package, and can be re-used to analyze different data, or to change hyper-parameters.

Conclusions and Recommendations

Muscle movement is the result of coordination between the body's nervous and muscular system. To move a muscle, the brain releases action potential, a short burst of electrical activity. Action potential propagates via the nervous system, triggering a motor unit. The motor unit generates a specific type of action potential, known as motor unit action potential (MUAP), which causes the attached muscle fibers to contract.

A technique known as electromyography (EMG), uses electrodes to measure the electrical activity across a single muscle. The raw signal is the result of many motor unit action potential peaks constructively and destructively interfering with each other. A blind source separation algorithm, as described by Negro et al. (2016), can decompose raw EMG signals into several individual electrical signals that can be ascribed to singular muscle units.

Our final data product presented is an open-source Python package, `EMGdecomPy`, containing two main elements. The first component is the reconstructed decomposition algorithm as described in the pseudocode of Negro et al. (2016). The algorithm has been slightly adjusted and validated 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.

References

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

Forbes, Scott C., Jonathan P. Little, and Darren G. Candow. 2012. "Exercise and Nutritional Interventions for Improving Aging Muscle Health." *Endocrine* 42 (1): 29–38. https://doi.org/10.1007/s12020-012-9676-1.

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.

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.

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.