Automatic Real-Time Tracking of Soleus Muscle Fascicles for Ankle Exoskeleton Control

Connor Williams

24 April 2023

1 Introduction

Exoskeletons are wearable devices that provide external forces to limbs, with goals of restoring, supporting, or enhancing human motion [1]. Depending on the type of exoskeleton, the assistive torques they provide can help people lift heavier objects, walk longer distances, or recover from injuries.

Ankle exoskeletons in particular, such as the Dephy ExoBoot shown in Figure 1, plantarflex the ankle to propel the wearer forward. Some applications of these include walking [2], running [3], and jumping [4]. They aim to reduce the metabolic cost of locomotion, which is the amount of energy required to move.

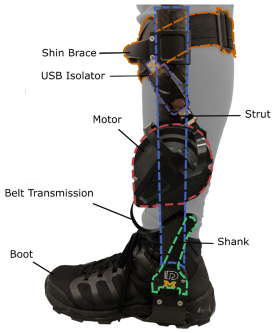

Figure 1: The Dephy ExoBoot, a powered ankle exoskeleton. The motor pulls on the heel of the boot to use the foot as a lever to propel the body forward during locomotion. Adapted from [5].

While these robots have great potential to serve society, how we control them is a difficult challenge. For ankle exoskeletons specifically, they are typically tuned for a specific task type, such as walking, running, or jumping, and switching between tasks can be difficult [1]. Additionally, the exact timing of the correct amount of torque is critical to optimally support the wearer. At best, mistimed torques can decrease the system's overall efficiency, but at worst

could cause the user to fall and injure themselves. Before we can see ankle exoskeletons used throughout our lives, we must research methods to safely control them.

One approach to this problem presented by Pridham and Stirling is an ankle exoskeleton control algorithm that uses real-time analysis of soleus muscle architecture [6]. As shown in Figure 2, the soleus is a muscle located in the back of the calf beneath the gastrocnemius. This muscle is responsible for plantar flexion, or toe pointing, which is the action taken to push the toe off of the ground to propel the body forward.

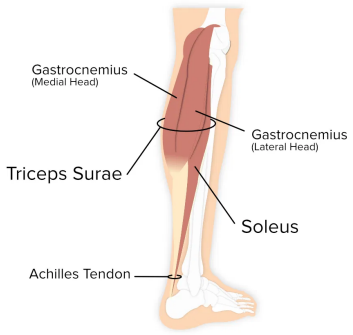

Figure 2: The location of the soleus in the calf. Adapted from [7].

The key soleus features of interest are fascicles and aponeuroses, shown in Figure 3. Fascicles are strands composed of muscle fibers and are, on average, 42mm long [8]. They contract according to signals coming from the nervous system. In [9], they describe a model that relates fascicle length and velocity to the energy used within the muscle. The exoskeleton control algorithm previously mentioned uses this model to actuate ankle exoskeletons alongside energy usage in the soleus. To implement this control algorithm, we must measure the lengths and velocities of these fascicles in real-time. This is an open area of research and is the focus of this project.

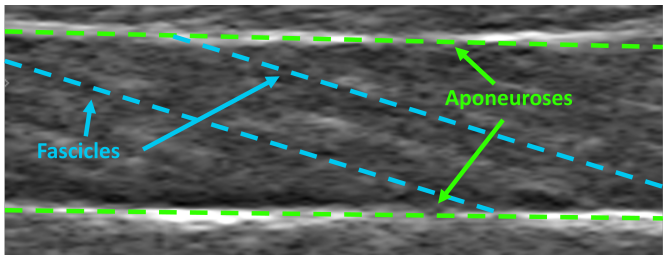

Figure 3: The soleus imaged using ultrasound technology. Aponeuroses are the bold horizontal lines, while fascicles are the dark spaces between the light diagonal lines. Adapted from [10].

1.1 Automated Fascicle Tracking

Fascicle tracking is done via analyzing ultrasound images. Ultrasound sensors are placed on the back of each calf muscle and directed towards the bone. This provides a cross-section of the soleus and enables the use of computer vision. The ultrasound sensors would be integrated into the human-exoskeleton system, and the high-level control algorithm would operate as follows:

1. The ultrasound sensor takes one scan of the calf muscle.
2. An on-board computer retrieves the scan and uses an algorithm to measure the fascicle lengths and velocities.
3. The computer inputs the fascicle information into the exoskeleton control algorithm, producing a torque.
4. The exoskeleton actuates with that torque until the next ultrasound scan.

After an extensive literature review to learn about how fascicle tracking has been performed in the past, we identified three common fascicle tracking algorithm archetypes: heuristics, affine flow, and deep learning.

1.1.1 Heuristic approaches

We classified heuristic algorithms as using a set of defined rules to enhance and interpret image features. These rules can involve blurring an image to reduce noise, increasing contrast via histogram equalization [11], highlighting edges using Canny edge detection [12], or detecting lines using the Hough Line Detector [13]. These rules are discrete steps in a computer vision pipeline, where the output of one step becomes the input to the next.

Seynnes et al. created a heuristic algorithm, Simple Muscle Architecture Analysis (SMA), that uses these steps to measure muscle fascicles in the gastrocnemius [14]. While the level of agreement between hand measured fascicles and their algorithm was within 95%, they reported their algorithm taking 8.4 seconds to process one frame. The researchers also advised users of SMA to carefully tune several processing parameters for every ultrasound sample, indicating their algorithm is sensitive to changes in the input images.

The measurements produced by heuristic fascicle tracking algorithms can "jitter". During walking, the ultrasound sensor may move slightly on the surface of skin, which can change the fascicles visible in the image. Additionally, changes in the muscle while it contracts causes certain areas to become brighter or darker. This means the heuristic could measure one particular fascicle in one frame, then jump to a completely different fascicle in the next frame, producing very inconsistent length measurements.

As demonstrated by the SMA algorithm, heuristics can be susceptible to failure when there are changes in image contrast or brightness. Each step in a heuristic has a set of adjustable parameters, such as how much to blur, how high a brightness threshold cutoff must be, or how many points must fall along a line

to be considered a fascicle. If these parameters are tuned too finely to a certain set of images, it may incorrectly weed out fascicles, causing false negatives. If the parameters are not strict enough, they may misclassify unwanted image features as fascicles, leading to false positives. Developing a universal heuristic is a difficult balance between sensitivity and specificity.

1.1.2 Affine flow approaches

Affine flow, short for affine optical flow, uses information about how the image changes to estimate where key points, such as fascicle endpoints, have likely moved [15]. It is based on optical flow, which produces motion estimates for each pixel in the frame. Since the endpoints of fascicles frequently leave the image frame, developers fit an affine transformation matrix to generalize the overall motion estimates of the image, allowing for extrapolation beyond the image frame.

The affine flow approach was explored in [16], and was later implemented into a software package named UltraTrack [17]. It features a graphical user interface where users load a prerecorded ultrasound video, select an initial fascicle by plotting its endpoints in the first video frame, and let the software estimate the motion of the endpoints for the rest of the video. Researchers in [18] demonstrated that UltraTrack produces measurements with very similar reliability and accuracy to professional examiners. In their study, they used the coefficient of multiple determination (CMD) between three examiners and UltraTrack to assess reliability, which was greater than 0.98 in all test cases after correcting for differing initial fascicle selection. To assess accuracy, they used the coefficient of multiple correlation (CMC), which was 0.94 after the same correction.

This algorithm has its own major fault: drift. Because the algorithm assesses overall motion and gradually changes its estimate of where the fascicle endpoints are, any error causes the endpoints to diverge from their true positions over time. UltraTrack mitigates drift by allowing users to select key-frames, which are frames in which the fascicles should be of the same length, such as each time the heel of the foot strikes the ground. Unfortunately, this exact procedure relies on manual detection of key-frames and assumes cyclic motion, like walking or running. Thus, it is infeasible for real-time and general use, so alternative drift-reduction strategies must be employed.

1.1.3 Deep learning approaches

Deep learning approaches involve collecting large amounts of hand-measured ultrasound videos, training a carefully crafted machine learning architecture, and using the resulting trained model to measure fascicle lengths in live ultrasound feed.

Rosa et al. created a proof-of-concept model that demonstrated its potential and published the recorded ultrasound data for public use [10]. When testing their model on ultrasound videos of people walking, their model correlated with measurements from UltraTrack with a Pearson's correlation of $r = 0.47$, which

they classified as a "moderate" correlation. They stated that while these initial results are promising for real-time use in exoskeleton control, more work must be done to determine how accurate a model must be to support real-time use. Deep learning methods are not typically robust to changes in the input data, such as using ultrasound videos from a person who was not involved in the model training phase, which can limit their generalizability.

2 Methods

Based on the literature, we explored combining a heuristic with affine flow to balance the strengths and weaknesses of both approaches. Because affine flow has been validated against hand labeled measurements [18], we used it as the backbone of our algorithm. To mitigate vertical drift, we used a heuristic to constrain the fascicle endpoint locations to the aponeuroses.

Our algorithm is represented graphically in Figure 4. It begins with initializing a fascicle, which we did manually. The fascicle was defined using two endpoints terminating at the aponeuroses and its length was measured by calculating the straight-line distance between the endpoints. From there, the fascicle's endpoints are moved using affine flow. The estimated endpoint positions may drift from their true positions, so the next steps are to detect the aponeuroses with a heuristic and to shift the endpoint positions back to the aponeuroses. The video progresses to the next frame and the algorithm repeats starting at the affine flow update step.

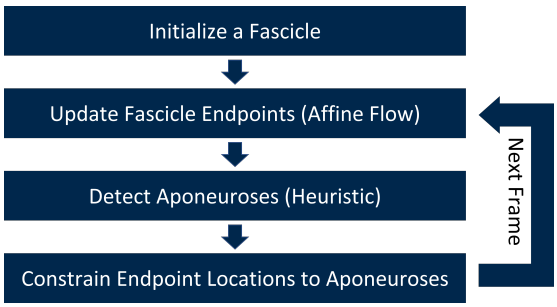

Figure 4: The high-level structure of our algorithm that aims to eliminate drift from affine flow with a constraint provided by a heuristic algorithm.

2.1 Endpoint constraint algorithms

For this project, we developed two endpoint constraint types: the line constraint, and the line region constraint. Both begin with using a heuristic to detect both of the aponeuroses in an ultrasound image. The heuristic creates several aponeurosis line proposals using the Hough Line Detector and selects the median lines as the best representations. From there, after the fascicle endpoint position

is updated using affine flow, the algorithm calculates how far the endpoints have moved from their respective aponeuroses. How each one handles that distance differs:

None: As a control, the algorithm makes no adjustments to the fascicle endpoint locations, allowing affine flow to drift without corrections.

Line constraint: Each endpoint is moved to the closest point on the lines representing their respective aponeuroses, shown in Figure 5

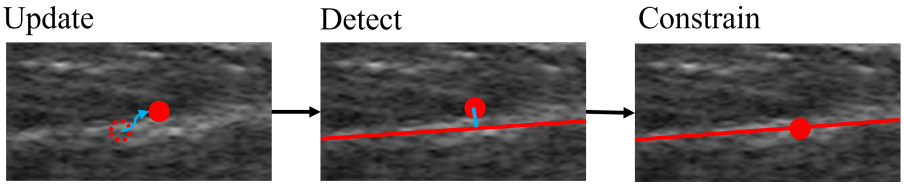

Figure 5: Visualization of the line constraint algorithm. Adapted from [10].

Line region constraint: Each endpoint is allowed to drift within a 10 pixel region about their respective aponeurosis line. If the endpoint has moved outside of that region, it is moved back into the region.

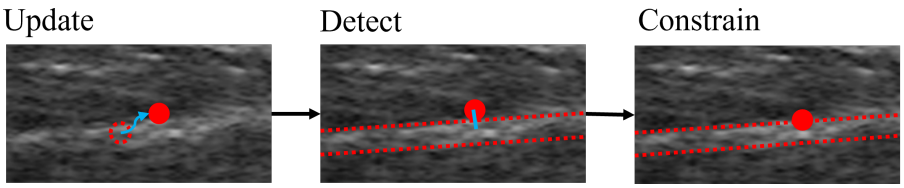

Figure 6: Visualization of the line region constraint algorithm. Adapted from [10].

2.2 Data

To test our algorithm, we used the data gathered in [10]. We selected two 60FPS ultrasound videos of subjects walking for approximately 24 seconds. The two videos were selected to contrast performance; one video already had little drift in affine flow, while the other was difficult to assess even with preexisting fascicle tracking tools. These were named Video 1 and Video 2, respectively.

2.3 Comparison metrics

We selected processing speed and net fascicle length drift as our performance metrics. The processing speed, measured in cycles per second (Hz), serves to demonstrate real-time algorithm efficacy. If the processing speed is too slow, the exoskeleton user may be negatively impacted, leading to the issues previously mentioned. We defined net drift as the difference between the minimum fascicle lengths in the first and last gait cycles. We used this to evaluate the effectiveness of our drift mitigation strategies.

3 Results

Table 1 contains the numerical results of running our three algorithms on the first selected video, Video 1. Figure 7 displays the fascicle length over time for each algorithm graphically. As mentioned before, Video 1 was selected because the affine flow algorithm already has a small drift of 2.2mm. This is useful because we would hope our constraints can perform similarly or better. The *affine flow + line constraint* algorithm successfully reduced drift to -0.5mm. We also observe a high divergence in the *affine flow + line region constraint* algorithm, as shown by its large decrease in length over time. When adding the constraints, the FPS dropped by roughly 110Hz for Video 1.

Algorithm	Cycles per Second (Hz)	Drift (mm)
Affine flow	**216**	2.2
Affine flow + line constraint	104	**-0.5**
Affine flow + line region constraint	101	-35.3

Table 1: Performance and drift metrics for Video 1. The best performing algorithm for each metric is in bold.

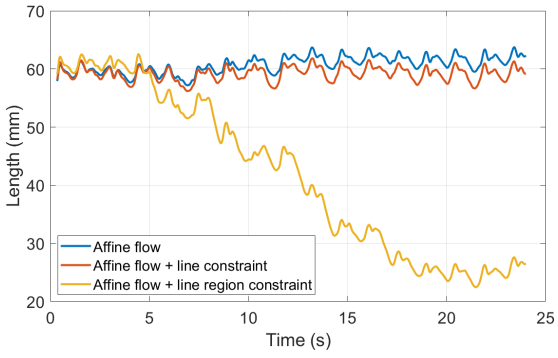

Figure 7: Comparison of lengths over time by various algorithms on Video 1.

Table 2 contains the numerical results of running our three algorithms on Video 2. Figure 8 displays the fascicle length over time for each algorithm graphically. In this video, *affine flow + line region constraint* reduced drift the best, followed closely behind by *affine flow + line constraint*. Both of these algorithms reduced the original drift by half. Adding constraints to affine flow dropped the FPS by about 140Hz. No algorithm diverged in this video, but no algorithm completely reduced drift.

Algorithm	Cycles per Second (Hz)	Drift (mm)
Affine flow	**217**	17.0
Affine flow + line constraint	77	8.0
Affine flow + line region constraint	73	**7.2**

Table 2: Performance and drift metrics for Video 2. The best performing algorithm for each metric is in bold.

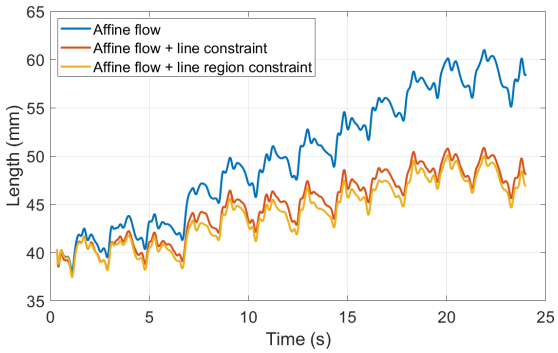

Figure 8: Comparison of lengths over time by various algorithms on Video 2

Between videos, the processing speed without constraints was similar at 216Hz and 217Hz. When we added the constraints, the processing speed was 30Hz slower in Video 2 than in Video 1. The *affine flow + line constraint* algorithm performed without diverging in both videos, but was not the best at reducing drift in Video 2.

4 Discussion

We expected the processing speed to drop when adding constraints. Without constraints, the algorithm must only execute the affine flow procedure without any of the heuristic steps. However, we did not expect to see a drop in frame rate in constraint algorithms between the two videos. This may be explained by the brightness of the aponeuroses in each video. In Video 2, the aponeuroses

are much brighter and bolder than in Video 1. The heuristic detects many lines using the Hough Line Detector, which finds more lines if there are more bright points in an image. This leads to lower processing speeds because of the extra stress placed on the line detector. This indicates that the algorithm's performance depends on what a human's soleus looks like under ultrasound, so it may work very well for people whose aponeuroses look thin, and poorly for people with bold aponeuroses.

As shown by the *affine flow + line region constraint* algorithm's drift in Video 1, this approach to drift mitigation may not be stable in all situations. The reason for instability is while this approach prevents endpoint drift perpendicular to the aponeuroses, there is no limit to how far the endpoints can move along the aponeuroses. In the case of Video 1, the *affine flow + line region constraint* algorithm caused the endpoints to diverge. While the *affine flow + line constraint* algorithm was stable in both Videos, its output may also diverge under certain conditions because of the same limitation.

The most promising outcome is that every algorithm executed faster than the ultrasound sensor could record (60Hz). This means if we ran similar algorithms to detect fascicles and use that to control an exoskeleton, this procedure could run without affecting human perception. As mentioned previously, even small delays between a human's action and an exoskeleton's reaction can have negative impacts on the user experience.

5 Limitations

While these results are promising, it is important to understand the shortcomings of our algorithm and our method we used to assess it. The following section describes these limitations of our experiment.

5.1 Accuracy Metrics

Our analysis did not include any accuracy metrics. Accuracy metrics would assess the impact that the constraints had on the measurement output. While the shape of the length curves over time may appear correct in Figures 7 and 8, it is impossible to tell without comparing them to ground-truth measurements. We had tried measuring the fascicles by hand, which took several days, but the result would need more time and verification before it could be used. We also looked into using UltraTrack to produce ground-truth measurements because it has been demonstrated to produce fascicle length measurements very similar to those of professional human examiners [18]. However, we decided against it because UltraTrack's underlying algorithm is affine flow. Because our algorithm is also based on affine flow, we did not use it for comparison because we would have compared an algorithm to itself.

5.2 Input Video Specifications

Video 1 and Video 2 were selected from the data set collected in [10] because our algorithm could analyze both of them without making any alterations, which is not the case for the remaining 4 videos in that data set. The aponeurosis detection heuristic expects to find aponeuroses within specific regions because there are several horizontal lines in the ultrasound images that the algorithm may misclassify as aponeuroses. Limiting the aponeurosis detection to only consider lines in specific regions prohibits people with different calf sizes from using this algorithm. To make this algorithm usable for everyone, we must figure out a different way of detecting aponeuroses that accommodates for the true range of calf architectures.

5.3 Fascicle Initialization

Affine flow requires initial fascicle endpoints before it can begin estimating where they have moved throughout the video. It is critical to accurately initialize the fascicle, otherwise all measurements that follow will be inaccurate. This could be done using a fascicle detecting heuristic. While the literature review and preliminary experimentation revealed heuristics do not follow specific fascicles, leading to jitter, they could be useful when selecting just one fascicle in the beginning of the algorithm. If fine-tuning a heuristic proves to be a challenge, we could instead use a pre-trained machine learning model, such as the one publicly available in [19].

5.4 Hardware

We developed and executed this fascicle tracking algorithm on a laptop with an Intel 8^{th} Gen. i5 CPU and an NVIDIA GTX 1050 GPU. Unfortunately exoskeletons must be lightweight in order to successfully reduce the metabolic cost of using them, so a large, powerful computer would not be suitable. Power capacity is also a limiting factor because ankle exoskeletons rely on batteries. With these sizing considerations in mind, our algorithm will likely run on a miniature computer, like the Intel NUC or a Raspberry Pi. When we move our algorithm to one of these platforms, we may see a drop in the processing speed, so we must further optimize our algorithm.

6 Lessons Learned

Throughout this project, I learned a great deal about the state of automatic fascicle tracking. When I first was presented with this project, I thought I would spend only about a month developing the fascicle tracking algorithm and would move onto testing it on an exoskeleton by the end of the semester. I had actually developed an entire heuristic algorithm over the course of one day - the first weekend of the semester that I started this project. I thought the rest of the project could be devoted to just improving that heuristic to reduce

the jitter. After a few weeks of making significantly less progress than that first day, I delved into the literature to see how this problem has been solved before. I slowly realized that this really is an open area of research and that no one had perfected real-time fascicle tracking. We had to re-scope this project to focus solely on algorithm development. This was slightly disheartening to realize at first, but ultimately it was worth taking the time to solve this problem methodically and I'm happy with the progress that we made.

I also learned what it's like to manage my own research project. Last summer, I worked on a research project that was led by a PhD student. I was given clear objectives, and my work always felt like it was progressing the project. In this capstone project, it was up to me to create my own objectives, with the help of my advisors.

I sometimes struggled during update meetings with my advisors, where I was asked questions that I did not completely know the answers to and I wouldn't know how to respond. I didn't want to give the impression that I hadn't learned anything since the previous meeting, so I'd try to answer all of their questions anyway. This created meetings that may not have been as productive as they could have been. Eventually, I realized that "I don't know" is an acceptable and *necessary* response sometimes. Without admitting that I didn't have complete understanding of the topic, I couldn't work with my advisors as effectively to learn. Once I started keeping closer track of my questions throughout each week, I was well-prepared for our update meetings and would learn much more.

7 Conclusions & Future Work

In this project, we combined previous approaches of tracking soleus muscle fascicles to control ankle exoskeletons in real-time. Our combination of affine flow and heuristics had positive results for measuring the lengths of fascicles in real-time from ultrasound images. In most cases with constraining affine flow output to aponeuroses, we saw a decrease in the amount of drift from the underlying affine flow algorithm without detrimentally decreasing the processing speed. While these results are limited to our small sample size, they demonstrate the potential for combining heuristic and affine flow approaches.

An improvement in future iterations could focus on the pennation angle of the fascicles rather than their absolute positions. Pennation angle is defined as the angle between a fascicle and the lower aponeurosis. Because fascicles lay in roughly the same direction, the pennation angle detected in one portion of the image should match the pennation angle in another, which could reduce the issues caused by jitter.

Our algorithms had acceptable processing speeds on both of the ultrasound videos that were used for testing. There was a significant difference in processing speed across videos, so we must evaluate these algorithms on more data sets to estimate the worst-case scenario. Moving forward to using these fascicle measurements to control an exoskeleton, the processing speed will be critical to ensure smooth control.

Once we find an algorithm that can reliably and accurately measure fascicles, we can optimize the code to boost the performance. Currently, for the sake of modularity and testing, the algorithm repeats some procedures like blurring the image to reduce noise. Identifying redundant tasks would be a first step in enhancing performance. Another approach would be to switch from Python, a scripting language, to C++, a language that is optimized via compiling.

Finally, after we refine our fascicle tracking algorithm, we can begin testing the control algorithm described in [6] on people. We can then begin to determine the efficacy this new and emerging method of control. When we combine this fascicle tracking algorithm with algorithms to read data from the ultrasound sensor and send control data to the exoskeleton, we may find another drop in performance due to the complexity of each of these tasks. To resolve this, we will consider multi-threading, which allows a computer to execute multiple tasks simultaneously. This means the computer could read ultrasound images, process ultrasound images, and control the exoskeleton simultaneously without each process affecting the efficiency of the others.

References

[1] R. Baud, A. R. Manzoori, A. Ijspeert, and M. Bouri, "Review of control strategies for lower-limb exoskeletons to assist gait," *Journal of NeuroEngineering and Rehabilitation*, vol. 18, p. 119, Jul 2021.

[2] L. M. Mooney, E. J. Rouse, and H. M. Herr, "Autonomous exoskeleton reduces metabolic cost of human walking," *Journal of NeuroEngineering and Rehabilitation*, vol. 11, p. 151, Nov 2014.

[3] D. E. Miller, G. R. Tan, E. M. Farina, A. L. Sheets-Singer, and S. H. Collins, "Characterizing the relationship between peak assistance torque and metabolic cost reduction during running with ankle exoskeletons," *Journal of NeuroEngineering and Rehabilitation*, vol. 19, p. 46, May 2022.

[4] B. Marinov, "Can you jump higher with a powered exoskeleton?," Apr 2023.

[5] R. L. Medrano, G. C. Thomas, and E. Rouse, "Methods for measuring the just noticeable difference for variable stimuli: Implications for perception of metabolic rate with exoskeleton assistance," in *2020 8th IEEE RAS/EMBS International Conference for Biomedical Robotics and Biomechatronics (BioRob)*, pp. 483–490, 2020.

[6] P. S. Pridham and L. Stirling, "Ankle exoskeleton torque controllers based on soleus muscle models," *Plos one*, vol. 18, no. 2, p. e0281944, 2023.

[7] Westcoastsci, "Soleus, the forgotten muscle for runners," Apr 2023.

[8] B. Bolsterlee, T. Finni, A. D'Souza, J. Eguchi, E. C. Clarke, and R. D. Herbert, "Three-dimensional architecture of the whole human soleus muscle in vivo," *PeerJ*, vol. 6, p. e4610, Apr. 2018.

[9] A. V. Hill, "The heat of shortening and the dynamic constants of muscle," *Proceedings of the Royal Society of London. Series B, Biological Sciences*, vol. 126, no. 843, pp. 136–195, 1938.

[10] L. G. Rosa, J. S. Zia, O. T. Inan, and G. S. Sawicki, "Machine learning to extract muscle fascicle length changes from dynamic ultrasound images in real-time," *Plos one*, vol. 16, no. 5, p. e0246611, 2021.

[11] S. M. Pizer, J. D. Austin, J. R. Perry., H. D. Safrit, and J. B. Zimmerman, "Adaptive Histogram Equalization For Automatic Contrast Enhancement Of Medical Images," in *Application of Optical Instrumentation in Medicine XIV and Picture Archiving and Communication Systems* (S. J. D. III and R. H. Schneider, eds.), vol. 0626, pp. 242 – 250, International Society for Optics and Photonics, SPIE, 1986.

[12] J. Canny, "A computational approach to edge detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-8, no. 6, pp. 679–698, 1986.

[13] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Commun. ACM*, vol. 15, p. 11–15, jan 1972.

[14] O. R. Seynnes and N. J. Cronin, "Simple muscle architecture analysis (sma): An imagej macro tool to automate measurements in b-mode ultrasound scans," *Plos one*, vol. 15, no. 2, p. e0229034, 2020.

[15] S. Baker and I. Matthews, "Lucas-kanade 20 years on: A unifying framework," *International Journal of Computer Vision*, vol. 56, pp. 221–255, Feb 2004.

[16] N. J. Cronin, C. P. Carty, R. S. Barrett, and G. Lichtwark, "Automatic tracking of medial gastrocnemius fascicle length during human locomotion," *Journal of applied physiology*, vol. 111, no. 5, pp. 1491–1496, 2011.

[17] D. J. Farris and G. A. Lichtwark, "Ultratrack: Software for semi-automated tracking of muscle fascicles in sequences of b-mode ultrasound images," *Computer methods and programs in biomedicine*, vol. 128, pp. 111–118, 2016.

[18] J. G. Gillett, R. S. Barrett, and G. A. Lichtwark, "Reliability and accuracy of an automated tracking algorithm to measure controlled passive and active muscle fascicle length changes from ultrasound," *Computer methods in biomechanics and biomedical engineering*, vol. 16, no. 6, pp. 678–687, 2013.

[19] N. J. Cronin, T. Finni, and O. Seynnes, "Fully automated analysis of muscle architecture from b-mode ultrasound images with deep learning," *arXiv preprint arXiv:2009.04790*, 2020.