Translating MatLab to Python

Berend Grandt, Kim Nguyen, Shelby Primeaux, & Grishma Shrestha

December 4, 2025

1 Introduction

The Pennington Biomedical Research Center currently uses MATLAB code introduced by Dr. Steven B. Heymsfield to substitute for DEXA body scans. DEXA scans are medical scans that assess body composition and density, usually to monitor bone health and provide biometrics like muscle mass and body fat percentage to provide valuable information about overall health. While MATLAB is a powerful tool, it requires expensive licensing and version updates which cause obsolescence of functions within the code. This makes it inconvenient to use over extended periods, especially when considering other factors such as employee turnover which result in information loss. Pennington has suggested using Python as an alternative programming language. Python has versatile and free open-source libraries as well as elegant syntax which make it accessible and durable. Translating the MATLAB code to Python would reduce costs and limit necessary changes which would streamline future updates.

2 What was done

Prior groups set a necessary foundation for our group to continue progressing on the project. Previous work included compiling original documentation to better understand the MATLAB code and centering and locating certain body parts. Because the code had little commenting, previous groups dedicated substantial time and effort to parsing the *Avatar.m* file and determining the structure of the code base. Since the majority of the schema of the code had been lost to time, the group had to reverse-engineer the code to discover multiple key components of using the code. One of the most pertinent was their discovery of the necessary file type, which is an *.obj* file. This finding was essential to subsequent translation work and gave us a head start on locating necessary data and testing files. They also compiled valuable documentation and clearly noted challenges and suggestions to give us guidance on our approach. They noted that we should not attempt to translate directly, but rather take a holistic view, which allowed us to make significant progress on the project and save lead time.

3 What Did We Do

We were able to make significant headway in developing the MATLAB functionalities in Python. Accurate landmarks located include hips, shoulders, collar, and legs, as well as measure trunk length using the previous crotch-detection code. While most other landmarks have been located, some are lacking precision including chest circumference, surface area, and volume. Improved accuracy should develop over time with additional testing. We also spent substantial time documenting and reorganizing the code and our repository for efficient transfer to future groups. With improved organization, it is now easier to locate necessary files, and future contributors will be able to navigate the project more efficiently. Our repository structure includes dedicated folders for the main file, the mesh orienting and cleaning processes, the convexity search functions, and the individual anatomical regions. Organizing the repository in this way maintains a logical flow that closely reflects the structure of our code, which enhances clarity and overall comprehensibility.

3.1 Initial Approach

With guidance from the previous groups to avoid translating line-by-line, our initial approach was not far off from our ultimate approach which allowed for significant progress on the project. We initially thought working from the bottom of the code upward would be best, and for us that seemed logical due to the dependencies created by prior functions. Because definitions are sequential in MATLAB, we figured that determining dependencies from the outermost functions (ie. those with the longest dependency chains) would give us a wide view of the code and allow us to minimize conflicts and errors from the start. Though this method was detailed and thorough, with our time constraint, the amount of hours that would have been spent on this approach was unreasonable considering the length of our project. We were also making little progress using this approach and at the suggestion of previous contributors, switched to a top down approach that proved to be more efficient. This approach locates the topmost functions first which allowed us to develop our own chain and eliminate unnecessary functions.

3.2 Problems

Throughout the project, we encountered several challenges, including understanding the structure and functionality of the original MATLAB code, determining how to organize our Python implementation, and identifying the resources needed to keep the project progressing. Although extensive documentation was provided at the beginning, the inability to run the MATLAB code on our machines made it difficult to interpret its structure and determine how to begin translating it. As previously mentioned, we eventually had to pivot to a new approach mid-project, which became more manageable once we received the crotch-detection code and the overview file. These resources clarified the key functionalities that needed to be replicated in Python, with the crotch-detection code in particular helping us locate multiple body parts, since most body parts are derived from the crotch-detection code. Even with this guidance, the MATLAB code remained intimidating due to its extensive dependencies, making it challenging to decide where to start and which functions should be translated first.

3.3 Output

We utilized multiple key libraries including **Trimesh**, **NumPy**, and **SciPy**. **Trimesh** was suggested by previous groups as visualization tool and has convexity and concavity applications which we did not explore in depth, but could be a future consideration. Particularly useful in **Trimesh** was a function called *slice plane* that allowed us to slice the given mesh at a point using a plane defined by a normal vector. This returned the positive normal side of the plane created from the mesh. With a few additional steps to isolate the desired body part, an entire portion of the body, such as the leg, could be identified by the code. **NumPy** performs repetitive mathematical operations and geometric calculations. **SciPy** is built in **NumPy** and performs high-level algorithms. These tools were sufficient for the scope of our project, but future groups could explore additional libraries as needed.

Two of the main structural components of the code used to get the desired output include the *cache* and *static* decorators in Python. *Cache* speeds up the run-time of the code by storing values to prevent repeated calculations. That is, once a body part is located, when that body part function is called by a subsequent function, the stored value is used rather than running the function again. The *static* decorator allowed us to modularize the code to prevent dependency loops and errors. We were able to organize the code simply into the main anatomical regions including the head, trunk, arms, and legs. *Static* prevents conflicts among these different regions. Using these decorators laid the foundation for the structure of the code which allowed us to efficiently locate the required landmarks.

4 Conclusions and Moving Forward

Although significant progress has been made in locating many body parts, there is still work to be done. Because we received very few test files, we ultimately focused on a single file to ensure the code functioned correctly. This presents a potential issue, because running the code on additional files may produce inconsistent results and require further adjustments. The extent of these adjustments is uncertain, making this an important consideration for future groups. If future groups are able to obtain additional testing files, it may resolve this issue. Our group was unable to obtain additional files due to permission restrictions from Pennington; however, if this limitation can be addressed in the future, it would help mitigate the issue as the code continues to evolve. Another potential next step is converting the measurements into the appropriate units. The code currently returns values in the mesh's native units, or none if no units are specified. However, medical staff require measurements in centimeters. Finally, additional work could be done in the overview file to document the Python functions and their behavior, which would further improve both documentation quality and overall efficiency.

References

[1] S. Sobhiyeh, M. Dechenaud, A. Dunkel, M. LaBorde, S. Kennedy, J. Shepherd, S. Heymsfield, and P. Wolenski, "Hole filling in 3d scans for digital anthropometric applications.," in Annual International Conference of the IEEE Engineering in Medicine and Biology Society, 2019.

[2] S. Sobhiyeh, A. Dunkel, S. Heymsfield, S. Kennedy, D. Marcelline, J. Weston, J. Sheperd, and P. Wolenski, "Digital anthropometry for body circumference measurements: Toward the development of universal three-dimensional optical system analysis software.," in Obes Sci Pract, 2021.

[3] S. Sobhiyeh, N. Borel, M. Dechenaud, C. Graham, J. Sheperd, M. Wong, P. Wolenski, and S. Heymsfield, "Fully automated pipeline for body composition estimation from 3d optical scans using principal component analysis: A shape up study.," in Annual International Conference of the IEEE Engineering in Medicine and Biology Society., 2020.

[4] S. Sobhiyeh, A. Dunkel, M. Dechenaud, S. Kennedy, J. Shepherd, S. Heymsfield, and P. Wolenski, "Crotch detection on 3d optical scans of human subjects.," 2019.