MATLAB to Python: Code Translation

Fall 2025: Berend Grandt, Kim Nguyen,
Shelby Primeaux, & Grishma Shrestha

Department of Mathematics, Louisiana State University,
Baton Rouge Advisors: Dr. Nadejda Drenska & Dr. Peter Wolenski

Pennington Biomedical Research Center
Louisiana State University

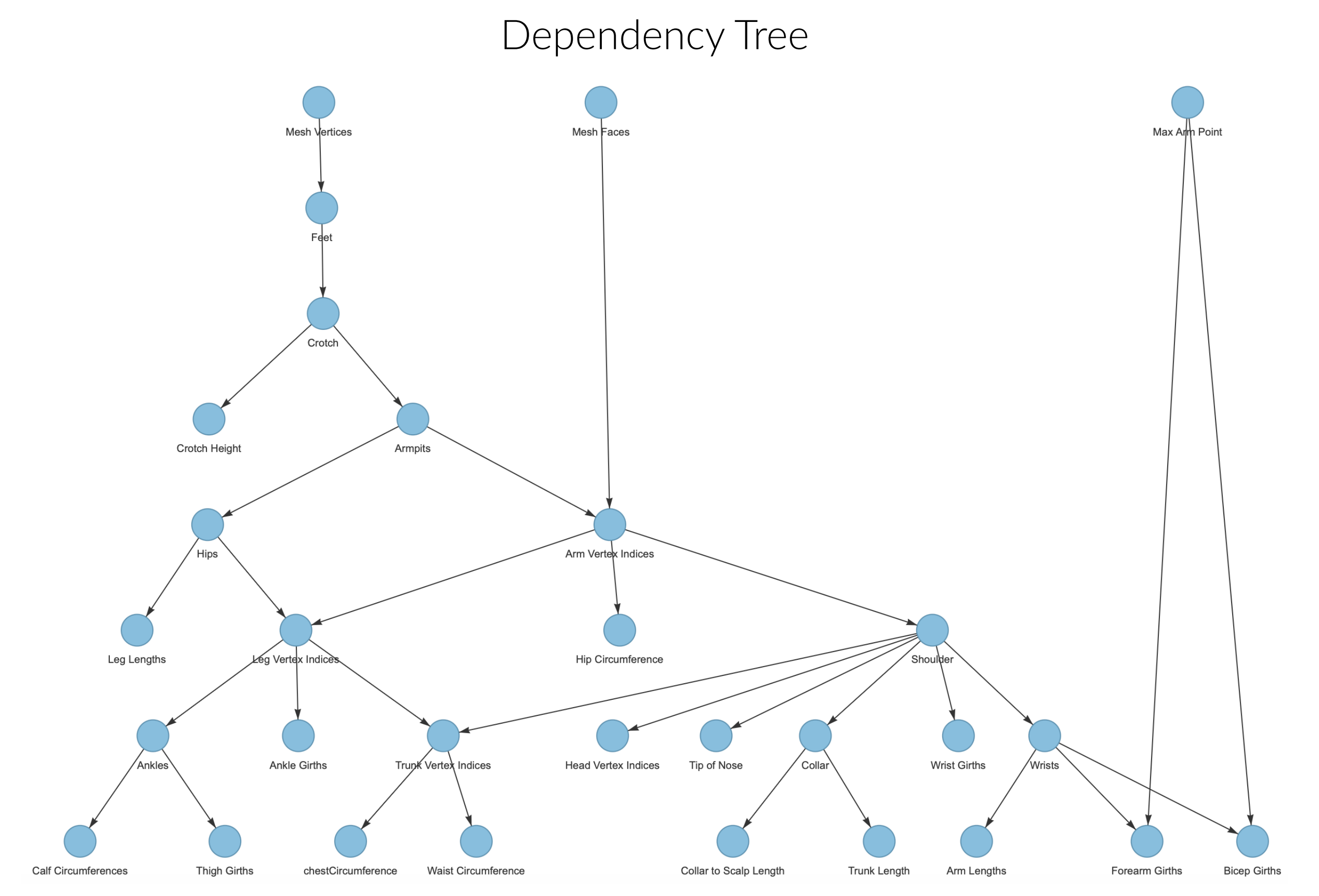
Introduction

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. While MATLAB is a powerful tool, it is costly to maintain and requires regular updates. Python, on the other hand, provides a free and flexible alternative with extensive open-source libraries for scientific computing. Converting the existing MATLAB code to Python would reduce costs, improve accessibility, and streamline future updates. The premise of the project is to make additional progress on the translation and provide clear documentation for future groups.

Objectives

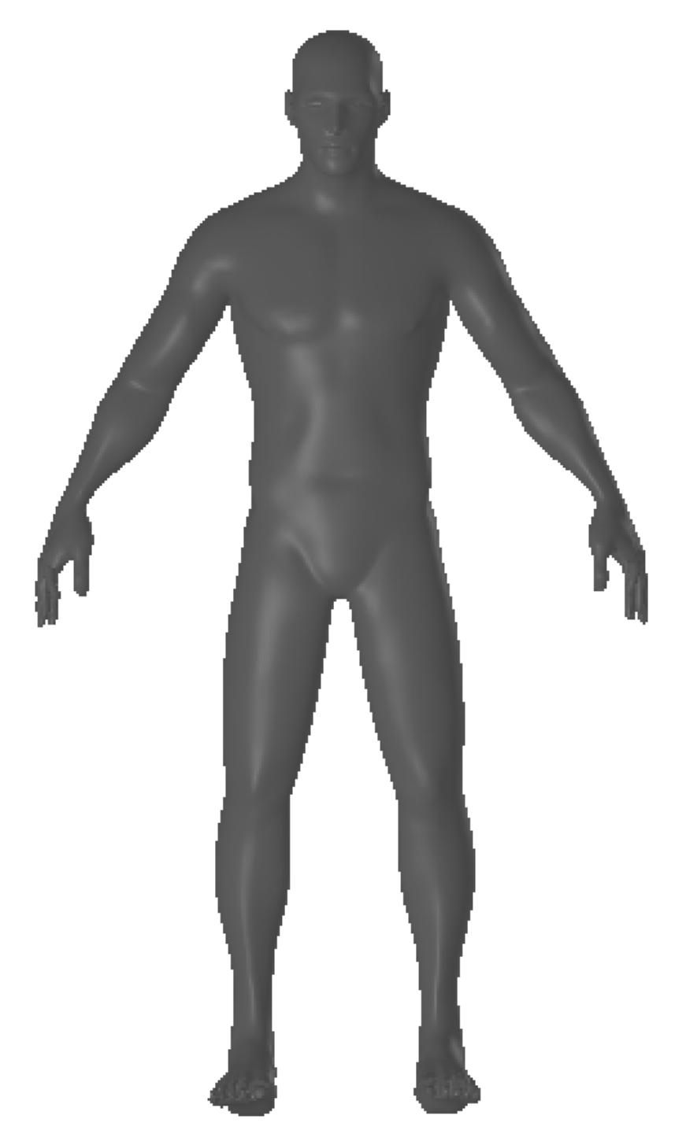
At the beginning of the project, our primary goal was to orient and clean the 3D mesh file used to obtain body measurements. Initially, we adopted a *bottom-up* approach, starting with the lowest-level functions in the dependency structure and working backward to identify how they were called. However, as the project progressed and new information became available, we determined that a *top-down* approach starting with the less dependent functions would be more effective. After noticing that the mesh was clean and oriented, we started to shift our focus towards identifying body parts on the model. Our objectives then evolved to refining body part detection using the existing crotch-detection code from previous cohorts, improving documentation, and organizing our work to ensure future research teams can smoothly continue the project.

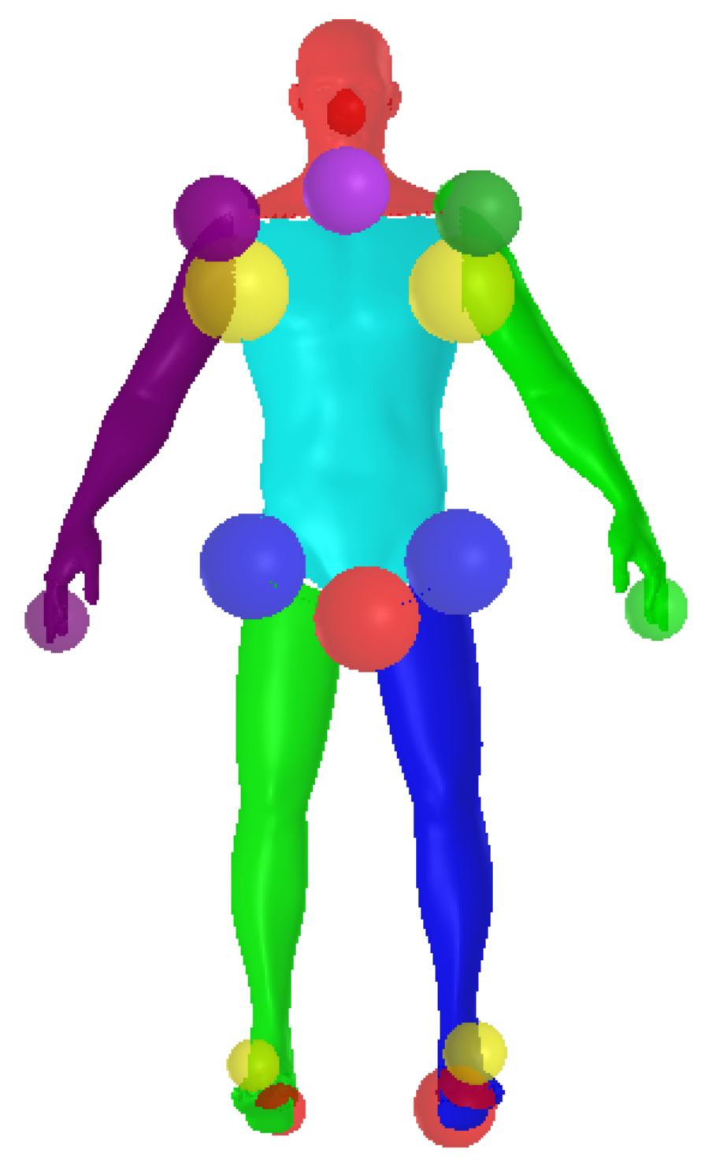
Code Structure and Approach



We were able to replicate the MATLAB functionality in Python and accurately locate the hips, shoulders, collar, and legs, as well as measure trunk length using the previous crotch-detection code. While all measurements have been obtained, the accuracy of some metrics, including chest circumference, surface area, and volume, varies. By utilizing cache in Python, we developed the code structure with the necessary capabilities while improving efficiency and reducing time requirements. Because caching allows us to store values, we can work on functions concurrently with fewer conflicts, since earlier functions do not need to be repeatedly called.

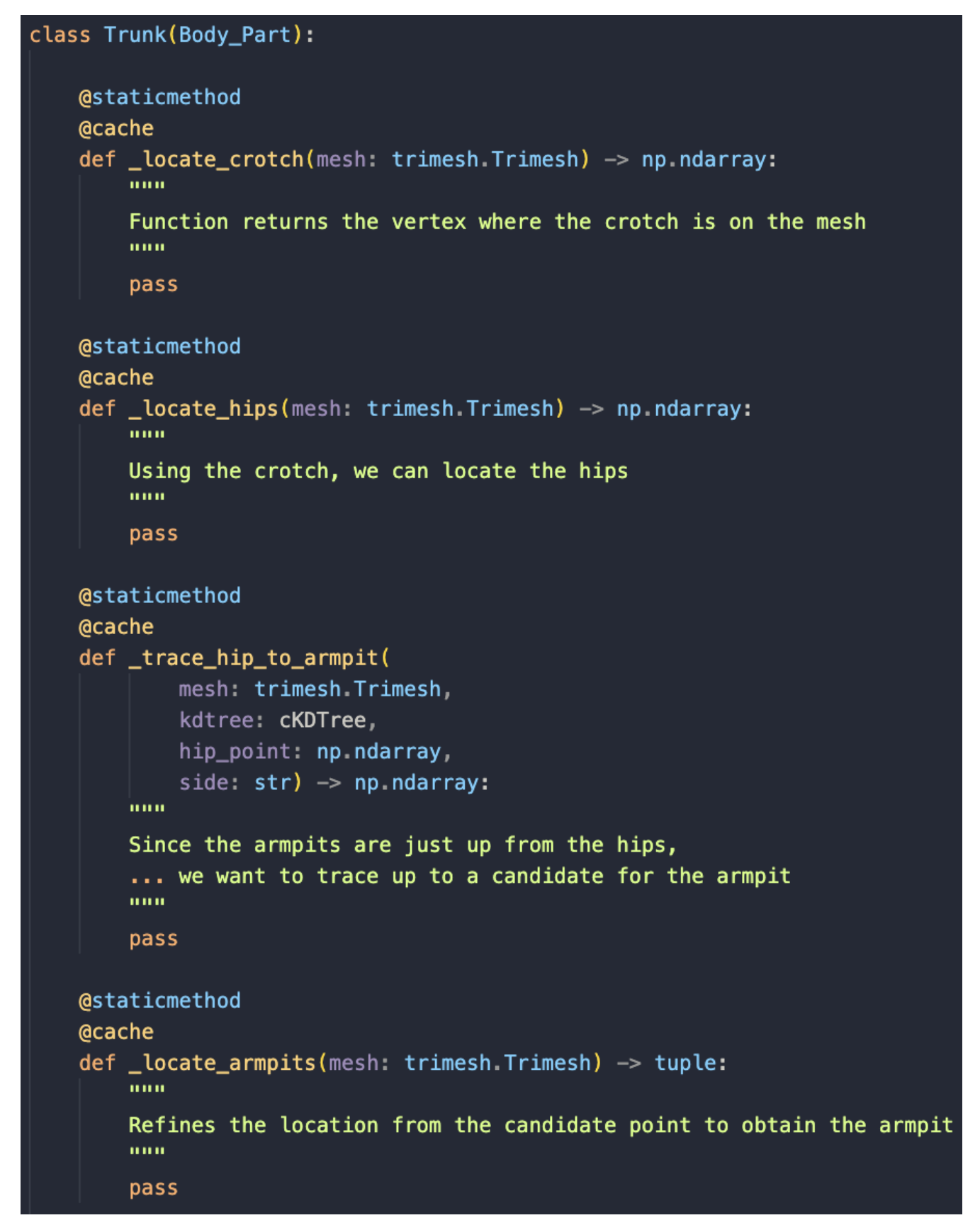
Output


Original input mesh without measurements


Output w/ marker coverage and body part detection

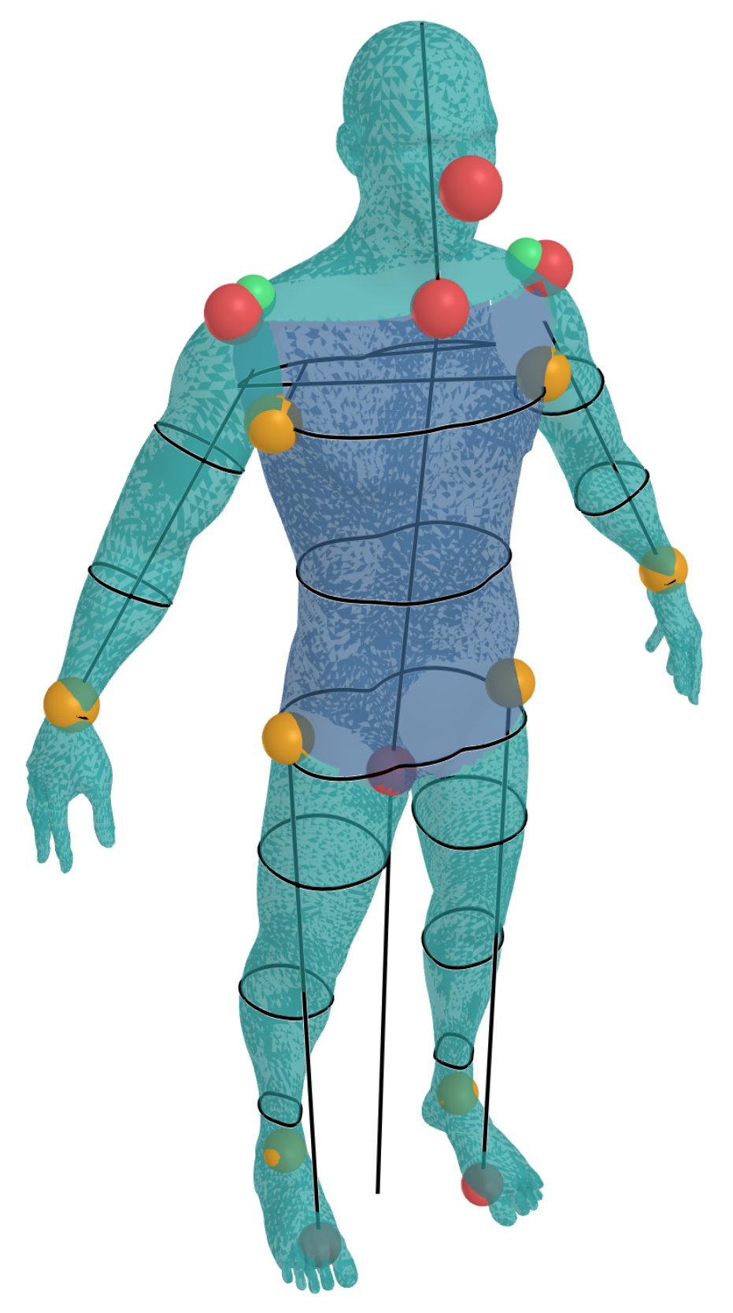
Python Capabilities

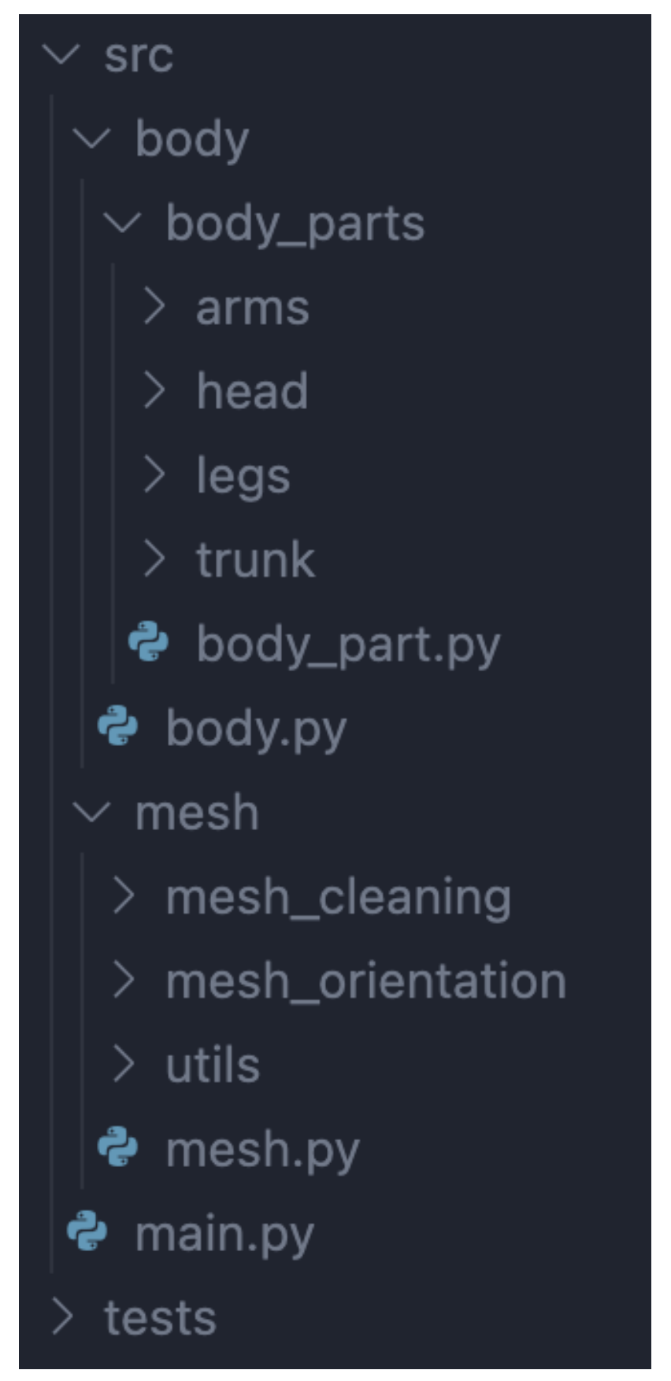
Compared to MATLAB, Python is elegantly-written, well-supported, and material updates are scarce. It's extensive standard library, including tools such as cache, allowed us to efficiently reproduce the MATLAB code in Python for free. Throughout the project we drew on many key libraries including **Trimesh**, **NumPy**, and **SciPy**. **Trimesh** is a 3D modeling library that cleans and orients the meshes and is useful for computing surface areas and volumes. **NumPy** stores points of the triangular mesh and performs vector and matrix operations in addition to some geometrical calculations. **SciPy** is an open source library built on **NumPy** which can perform high-level, efficient algorithms. While Python offers a wide range of advanced features, we focused on these tools to form the core of the computational framework needed for the scope of our project.



```python
class Trunk(Body_Part):

    @staticmethod
    @cache
    def _locate_crotch(mesh: trimesh.Trimesh) -> np.ndarray:
        """
        Function returns the vertex where the crotch is on the mesh
        """
        pass

    @staticmethod
    @cache
    def _locate_hips(mesh: trimesh.Trimesh) -> np.ndarray:
        """
        Using the crotch, we can locate the hips
        """
        pass

    @staticmethod
    @cache
    def _trace_hip_to_armpit(
            mesh: trimesh.Trimesh,
            kdtree: cKDTree,
            hip_point: np.ndarray,
            side: str) -> np.ndarray:
        """
        Since the armpits are just up from the hips,
        ... we want to trace up to a candidate for the armpit
        """
        pass

    @staticmethod
    @cache
    def _locate_armpits(mesh: trimesh.Trimesh) -> tuple:
        """
        Refines the location from the candidate point to obtain the armpit
        """
        pass
```

Class with static and cache


Circumference estimation



```
⌄ src
  ⌄ body
    ⌄ body_parts
      > arms
      > head
      > legs
      > trunk
      body_part.py
    body.py
  ⌄ mesh
    > mesh_cleaning
    > mesh_orientation
    > utils
    mesh.py
  main.py
> tests
```

Code reorganization

Challenges

The project presented a steep learning curve from the beginning. The challenges included understanding the MATLAB code and how it works, structuring our Python code, developing an approach, and determining what resources were needed to keep the project moving forward. Initially, the MATLAB code was overwhelming and the number of functions to account for was unmanageable, making it hard to determine how to structure our code. This required some additional research to determine which libraries were needed and which functionalities may be helpful, since we were not familiar at first with many of the Python tools we employed. At the outset, we had access to only a single data file, which also presented an early challenge. As our work progressed, we identified the need for further required materials and coordinated with other groups to obtain them, improving the overall quality of our workflow. Although permission restrictions prevented us from testing with a larger dataset, the accessible data was sufficient to validate the central functionality of our code thus far.

Moving Forward

While there was significant progress made by locating multiple body parts and documenting the project, much work remains. The scope of our tests was limited as we focused mostly on getting the code to work on one file. Running the code on different files could show unexpected results and require adjustments to the code for more accurate measurements. In addition to locating other body parts, a later goal is to adjust the units. The desired units are centimeters, however the code currently returns either the unit system of the mesh or none if units are not specified. In the future, more documentation could be done of the functions used in our code into the excel sheet because the majority are incomplete.

Acknowledgements

- We would like to thank Prof. Peter Wolenski, Dr. Nadejda Drenska, and Matthew Lemoine for guiding and supporting us.
- We would like to thank Clinten Graham, Muhammed Habibovic, Steven Heymsfield, Sophia Ramirez, and Sima Sobhiyeh for their help with learning the MatLab code.

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