MATLAB to Python: Initial Code Translation

By Joshua Champagne, Dow Draper, Matthew Lemoine, Aaron Scott, Robert Smith, Paul Yeon

Introduction

The project that we worked on this semester was to translate code from MATLAB to Python. The code that we looked at is currently in use by Pennington Biomedical Research Center. Its purpose is to look at a given scan of an individual and get biomarkers from the scan. While the MATLAB code is in use now by Pennington, they need to have a MATLAB license to operate it. Python presents a better option as it is easier to access by future researchers and implement into future projects. Writing the code in Python would also give us access to libraries in Python thus allowing us to modernize and condense the code.

Methods

We started by parsing and comprehending the .m file (MATLAB code extension) that Pennington is currently using. The *Avatar.m* file that we received had all the functions used to get the biomarkers and to fix the scan if it needed fixing. Originally after looking at the Avatar file, we thought that we could translate line-by-line, but issues arose when we realized that we could not run the code without original input data provided by Pennington. This was due to several assumptions that were made in the original MATLAB code regarding the schema of the mesh file which have since been lost to time. We eventually reverse-engineered the original code to determine that the input file should be an .obj file, which is a 3D mesh file typically used in digital graphical design. However, once we found which file the code used, we ran into the issue of not having a scan from Pennington to plug into the code. So, we found an ideal .obj file of a man to start using.

The biomarkers that we are interested in are the major body parts from the feet to the top of the head. These include the feet, knees, thighs, hips, crotch, forearms, biceps, shoulders, chest, neck, and top of the head. These biomarkers can help us learn more about an individual and can be used in biomedical research. The math clinic intends to use these measurements in further machine learning pursuits to estimate more complex biomedical information such as body fat percentage and appendicular lean mass.

What We’ve Done

We created python methods to initialize and partly clean .obj and .ply mesh files of body scans. We created functions to center and locate certain body parts (currently the feet). Our techniques were inspired by the MATLAB code but weren’t directly translatable, so we introduced our own methods for dealing with various problems not addressed by the original software.

We also compiled a list of original documentation over the course of the semester that gave us some insight into how the program works. As the MATLAB code is largely uncommented, and thus the original intention of the code would have been lost if not for these documents.

Methodology and Difficulties

Centering the mesh was difficult because the input parameters were vague. Consider the situation where the coordinates of a mesh are defined with respect to a nonstandard axis (i.e. the basis vector is not e_1, e_2, or e_3). Defining the axes based on the longest directions may create directions with only relative meaning; the x-axis for mesh A may differ from the x axis for mesh B. Not having access to data means that we must accommodate any input format, which greatly increases the number of variables we must account for. Difficulties like these must be addressed with a mathematically sound approach to prove their efficacy.

Where We Started

To start, we were given four files with MATLAB code from the original project. Two of the files consisted of the main body of the code. The main difference between these two files was the updated version; one of the files seemed to be a more recent and complete version. Three of the files appeared to be supplemental functions for the main code, which have since become defunct.

Problems

While working on this project, we encountered several issues that hindered our progress. One of the first difficulties that our group faced was the lack of coding experience from the members of our group. The members of our group consisted of students from various STEM majors, but none of us were highly proficient in both programming languages. This made it difficult to effectively understand and translate the code. Another issue that limited our work was the limited documentation provided when we started the project. When initially received the code, the comments explaining the purpose of the various functions were either limited or nonexistent. This in conjunction with a delayed connection with a contact that originally made the MATLAB code made it difficult to set goals for the project.

Conclusions and Moving Forward

By the end of the semester, our group was able to start the groundwork for future groups to pick up and continue this project. Although we were not able to completely translate the code, we were able to set the project up in a way that other groups could easily look over our materials and begin work. We have begun work and created documentation that can easily be understood for the next group to work on the project. The work that we have done can be used as a starting point for the next group.

For the next group that is taking this project over, you should strongly consider reading through the whole Avatar code and trying to understand how the original coders decided to write this code. We think that moving forward the next steps should be to attempt to replicate the output of each function instead of translating the MATLAB code line by line as python has many libraries that can streamline a lot of the manual work implemented by the older codebase.

We recommend writing clear and concise documentation in the code in the form of comments, docstrings, type annotations, and sound variable names. Our predecessors did not follow these best practices in their code, which resulted in a large amount of team resources being dedicated towards aggregating information this semester. Additionally, we suggest using visualization tools to ease the process of introducing new features; our team used the trimesh python library, but the choice of library is not a hard requirement.

References

Sobhiyeh, Sima et al. “Hole Filling in 3D Scans for Digital Anthropometric Applications.” *Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual International Conference* vol. 2019 (2019): 2752-2757.

Sobhiyeh, S., Kennedy, S., Dunkel, A., Dechenaud, M.E., Weston, J.A., Shepherd, J., Wolenski, P. and Heymsfield, S.B. (2021), “Digital anthropometry for body circumference measurements: Toward the development of universal three-dimensional optical system analysis software.” Obes Sci Pract, 7: 35-44.

Sobhiyeh, Sima et al. “Fully Automated Pipeline for Body Composition Estimation from 3D Optical Scans using Principal Component Analysis: A Shape Up Study.” *Annual International Conference of the IEEE Engineering in Medicine and Biology Society. IEEE Engineering in Medicine and Biology Society. Annual International Conference* vol. 2020 (2020): 1853-1858. doi:10.1109/EMBC44109.2020.9175211

Sobhiyeh, Sima & Dunkel, Alexander & Dechenaud, Marcelline & Kennedy, Samantha & Shepherd, John & Heymsfield, Steven & Wolenski, Peter. (2019). “Crotch detection on 3D optical scans of human subjects.” Electronic Imaging. 2019. 10-1. 10.2352/ISSN.2470-1173.2019.16.3DMP-010.