**Folder: Mesh\_Optimization**

This folder contains scripts that optimize the PAM placement by using points contained in the mesh files that define the structure of the OpenSim bone models.

Note: Many of these scripts contain file paths specific to my (Connor’s) PC. These can be updated to be more general, as they are only using scripts and functions contained within the GitHub repository.

**Files and Descriptions:**

***Muscle\_Name*\_Mesh\_Opt.m** – These scripts use the skeletal mesh from OpenSim to determine the best placement for PAMs in order to replace one or more muscles. Each script follows the general structure:

1. Add file paths to the human muscle class structure, robot PAM class structure, functions folder, and OpenSim data folder
2. Creates transformation matrices for the joints that the muscles will cross over. These are made by creating a linearly spaced vector of angles between the joints minimum and maximum orientations. Those angles are then used to create a rotation matrix and then a transformation matrix by using the distance between the first point in a muscle’s reference frame and the following joint frames that the muscle crosses over into.
3. Runs the class structure for the human muscles.
4. Runs the class structure for the PAM based on an initial starting point. This initial starting point is typically the same as the human muscle’s locations, but may have one or more points removed to make a more linear path.
5. If the joint has more than one degree of freedom, the torque tensor is unstacked into multiple torque matrices that only capture information about one axis.
   1. If the PAM is replacing a group of muscles, those human muscles have their torques added together for comparison later against the PAM torque.
6. The cost function runs to get a starting value of the cost.
7. The mesh optimization begins, by importing location data from the skeletal locations that create the model mesh. For each run through the optimization loop, one location is updated to be one of the locations from the mesh. Every permutation between two bones that the muscle connects to is used, with the cost value being updated for ever run, looking for the lowest value.
8. Once the optimization is finished, the best location is updated. Torque profiles are generated for the human and optimal pam location. A plot of the angle between the human torque vector and the pam torque vector is generated. A plot of the human skeleton with the human muscle in red and the pam in green is created.

These scripts can be run without any changes on the users part. These are a starting point for optimization. The results from these scripts should always be the same, since the locations they are pulling from are constant, so the solution should be a global minimum for the data set.

The intention was for these scripts to serve as a starting point, for both the optimization code and for generating decent locations for physical PAM attachments. This code can be improved by adding onto it a more traditional machine learning algorithm, in which after the best location is found from the mesh data points, that would serve as a starting point to do a first order gradient descent search around that point. The implementation of this is in “Bifemsh\_Mesh\_Refine\_Opt.m”.

**Bifemsh\_Mesh\_Refine\_Opt.m** – This script is an example extension of the Mesh\_Opt code that was discussed previously. It contains all of the previous code, but adds a component that uses a machine learning algorithm to search the space around the best attachment location based on the skeletal data points. The algorithm looks at 8 points around the muscle locations before and after a point and calculates the cost function between all 64 connections using those points. Once a new minimum is found between all of those points, it uses the minimum cost locations to repeat the process. It outputs the same form of results as the previous code.

**costFunctionKnee.m** – This cost function is an adjustment from the generic one found in the “Functions” folder. Because the knee only actuates in one direction, this cost function more heavily weights the torque and torque angle around the z direction of the knee, and reduces the value of those parameters in the other directions.

Inputs: This function takes the human muscle torque and the PAM replacement torque as inputs.

Outputs: The output of this function is a C or cost value. The value is unitless and can be scaled to any arbitrary amount through the use of the gains Gt and Ga.