Start at: multi\_hardware/Trackers/Ximea\_imperx\_demo\_threaded.py

* Program runs Ximea 6 DOF in its own thread driven by class CLM [line 42], which is launched by the main program. Thread calls \_\_init\_\_, which does:
  + Reads the JSON file for the cad model and vectorizes the elements
* The main program calls CLM::start(), which in turn calls CLM::calc() - Ximea\_imperx\_demo\_threaded.py [line 85]
  + Get the latest image from the Ximea
  + Estimate\_Pose\_Charucoboard\_Ximea() - Chaurco\_Specific/CharucoHelpers.py [line 208]
    - Cv::aruco::detectMarkers()
    - Cv::aruco::interpolateCornersCharuco()
    - Get\_pose() - Chaurco\_Specific/CharucoHelpers.py [line 277]
      * solvePNP\_Charuco\_Board() - CharucoHelpers.py [line 388]
      * cv::Rodrigues()
      * decompose\_extrinsics() - misc/GenericHelpers.py [line 17]
      * charuco\_reprojection\_error() - CharucoHelpers.py [line 380]
  + Get the angles from the PTU
  + ccm.dynamic\_extrinsics\_correct\_order() - Main/Deepaks\_Model/ccm.py [line 98]
    - Matrices::rotation() - Main/Deepaks\_model/matrices.py [line 40]
    - Matrices::translation() - Main/Deepaks\_model/matrics.py [line 47]
  + Ximea\_to\_imperx\_frame() - CharucoHelpers.py [line 312]
    - Just uses np.matmul
  + decompose\_Extrinsics() - GenericHelpers.py [line 17]
  + mat2euler()
    - imported from transforms3d.euler - need C++ implementation
  + The thread makes the 6DOF available to the main thread through member functions
* In the main thread:
  + In a loop, Fetch image, fetch the 6 DOF, do some work to put up the display on the image
  + Finally, display the image

Going to need matrix class that supports operations. Validate against

Need matrix to euler code from someplace (<https://www.learnopencv.com/rotation-matrix-to-euler-angles/>)