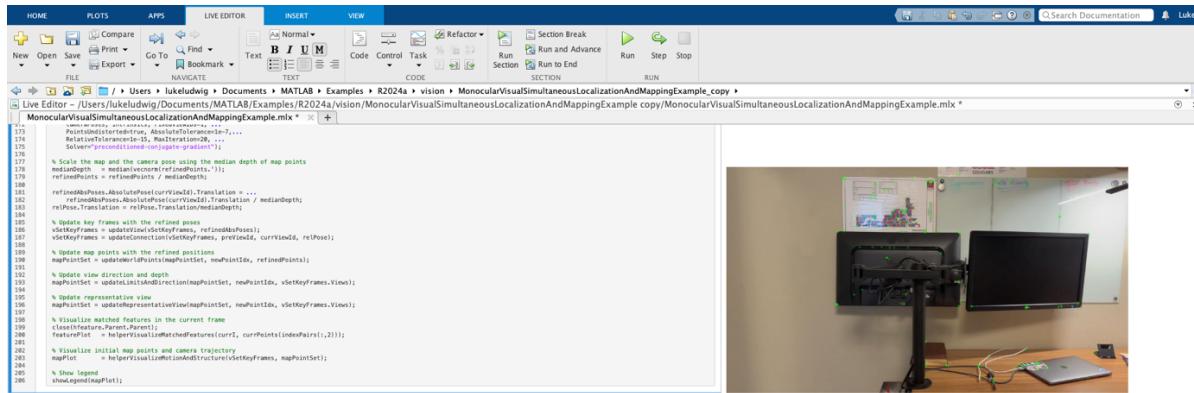


Step 2 Summary

The second step took a lot of time to figure out since the creation of a dataset was not outlined and there was little to no information when I searched google. I came up with a solution in MATLAB which took my mp4 video file and saved a frame after every 15 frames. This was inspired by MATLAB since they acquired key frames after every 20 frames for a 30-fps camera. I intended to include the rawVideo.mp4 file in the GitHub repository but the file size was too large. I saved the key frames into a file folder and saved them in an accessible location for MATLAB to access to perform analysis. The first section of code and its output is shown below.



```

% Initialize variables
map = monocularSLAM();
map.cameraPoses = cameraPoses;
map.intrinsic = intrinsic;
map.curFeatures = curFeatures;
map.curFrameIdx = curFrameIdx;
map.curLocations = curLocations;
map.curMatchedPoints = curMatchedPoints;
map.curPoints = curPoints;
map.curScales = curScales;
map.curTimestamp = curTimestamp;
map.featureIdx = featureIdx;
map.first = first;
map.focalLength = focalLength;
map.feature = feature;
map.himage = image;
map.imageSize = imageSize;
map.intrinsic = intrinsic;
map.indexPairs = indexPairs;
map.inlierCurPoints = inlierCurPoints;
map.inlierPrePoints = inlierPrePoints;
map.inliersdxF = inliersdxF;
map.inliersdxFH = inliersdxFH;

% Scale the map and the camera pose using the median depth of map points
medianDepth = median(map.refinedPoints(:,2));
refinedPoints = refinedPoints / medianDepth;

% Set initial camera pose
refIntrinsic = cameraIntrinsic;
refIntrinsic.PoseTranslation = refPose.Translation / medianDepth;
refPose.Translation = refPose.Translation / medianDepth;

% Update key frames with the refined pose
vSetKeyFrames = updateConnection(vSetKeyFrames, refinedPose);
vSetKeyFrames = updateConnection(vSetKeyFrames, previousIdx, curPose, refPose);

% Update map points with the refined positions
mapPointSet = updateMapPointSet(mapPointSet, refinedPoints);
mapPointSet = updateInitialAndDirection(mapPointSet, newPointIdx, refinedPoint);

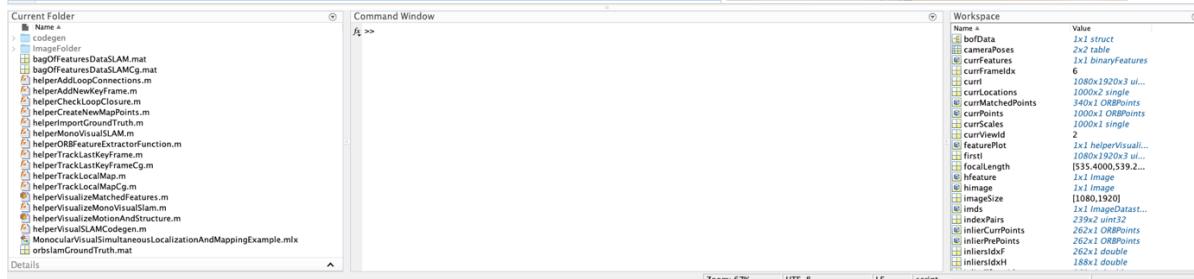
% Update view direction and depth
mapPointSet = updateInitialAndDirection(mapPointSet, newPointIdx, vSetKeyFrames.View);
mapPointSet = updateRepresentativeView(mapPointSet, newPointIdx, vSetKeyFrames.View);

% Visualize initial map points in the current frame
featureIdx = helperVisualizeMonocularFeatures(curIdx, currPoints(indexPairs(:,2)));
featureIdx = helperVisualizeMonocularFeatures(curIdx, currPoints(indexPairs(:,1)));
featureIdx = helperVisualizeMonocularFeatures(curIdx, mapPointSet);

% Visualize initial map points and camera trajectory
helperVisualizeMonocularTrajectory(featureIdx, mapPointSet);
helperVisualizeMonocularStructure(featureIdx, mapPointSet);

% Show legend
showLegend(map);

```



```

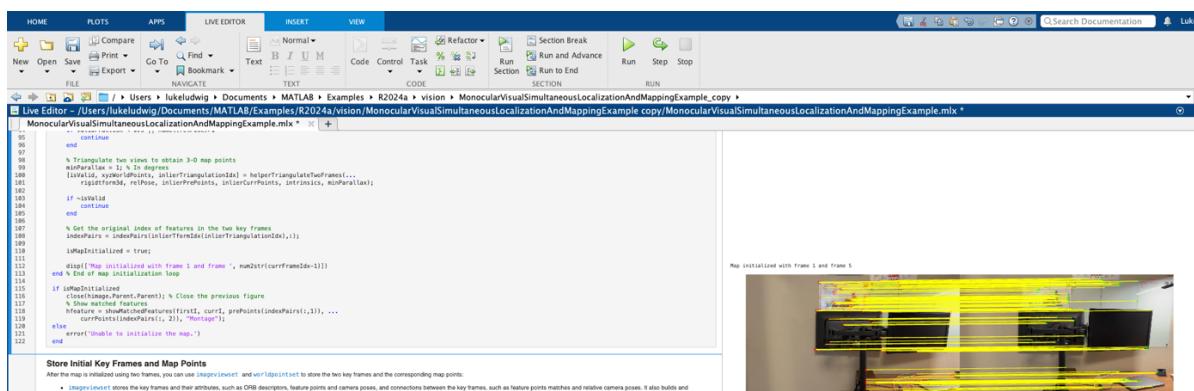
% Get the original index of features in the two key frames
indexPairs = indexPairs(:,curFrameIdx:(curFrameIdx+1));
imMapInitialized = true;

% Diagnose "Map initialized with frame 1 and frame 1"
diagnose("Map initialized with frame 1 and frame 1");

end % End of map initialization loop

if ~isvalid(map)
    % Close the previous figure
    close(hFigure.Parent.Parent); % Close the previous figure
    % Show matched features
    % Add the current matched features if first, cur1, prePoints(indexPairs(:,1)), ...
    % currPoints(indexPairs(:,2)), "Montage");
    % else "Unable to initialize the map."
    error("Unable to initialize the map.");
end

```



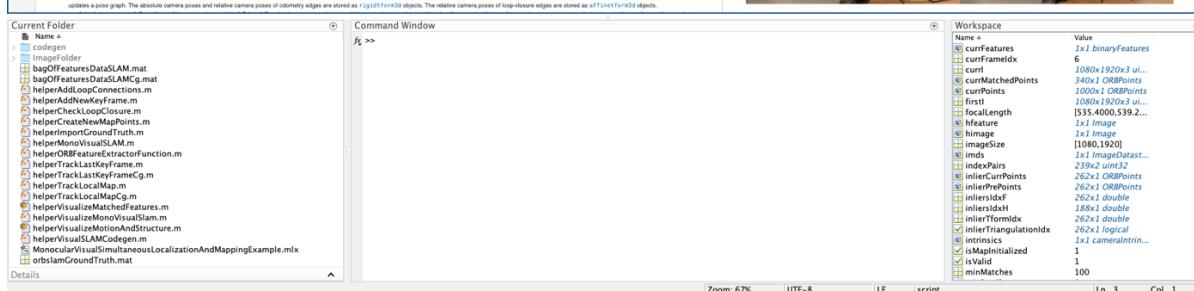
```

Store Initial Key Frames and Map Points
After the map is initialized using two frames, you can use imageviewet, and world2d to store the key frames and the corresponding map points.
• imageviewet stores the key frames and their attributes, such as ORB descriptors, feature points and camera poses, and connections between the key frames, such as feature points matches and relative camera poses. If it also builds and updates a graph, the absolute camera poses and relative camera poses of boundary edges are stored as affineTransform objects. The relative camera poses of non-boundary edges are stored as affineTransform3D objects.

Current Folder
Name
bagOfFeaturesDataSLAM.mat
bagOfFeaturesDataSLAMCg.mat
helperAddLoopConnections.m
helperAddNewKeyFrame.m
helperCheckLoopClosure.m
helperImportGroundTruth.m
helperMonoVisualSLAM.m
helperORBFeatureExtractorFunction.m
helperTrackLastKeyFrameCg.m
helperTrackLocalMap.m
helperTrackLocalMapCg.m
helperVisualizeMonocularFeatures.m
helperVisualizeMotionAndStructure.m
helperVisualSLAMCgcodegen.m
MonocularVisualSimultaneousLocalizationAndMappingExample.mlx
orbSLAMGroundTruth.mat

Command Window
f1 >>

```



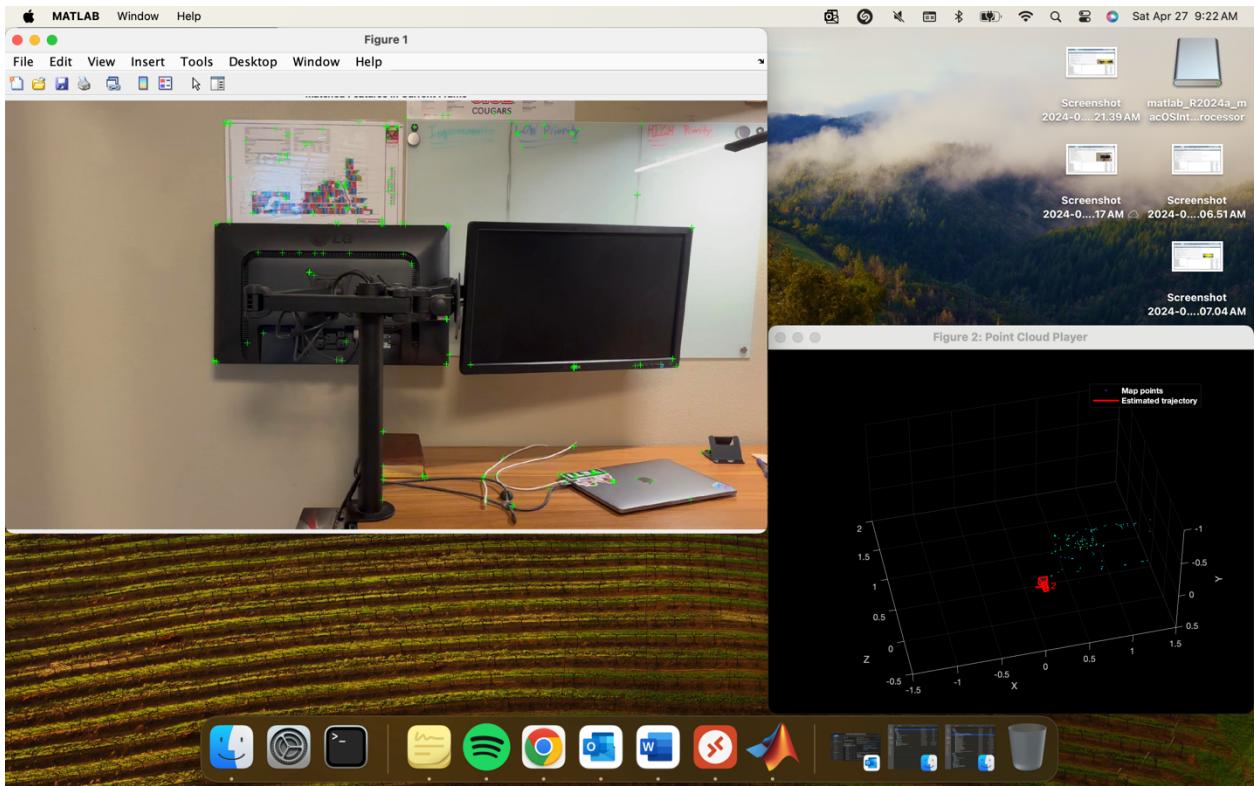
```

Current Folder
Name
bagOfFeaturesDataSLAM.mat
bagOfFeaturesDataSLAMCg.mat
helperAddLoopConnections.m
helperAddNewKeyFrame.m
helperCheckLoopClosure.m
helperImportGroundTruth.m
helperMonoVisualSLAM.m
helperORBFeatureExtractorFunction.m
helperTrackLastKeyFrameCg.m
helperTrackLocalMap.m
helperTrackLocalMapCg.m
helperVisualizeMonocularFeatures.m
helperVisualizeMotionAndStructure.m
helperVisualSLAMCgcodegen.m
MonocularVisualSimultaneousLocalizationAndMappingExample.mlx
orbSLAMGroundTruth.mat

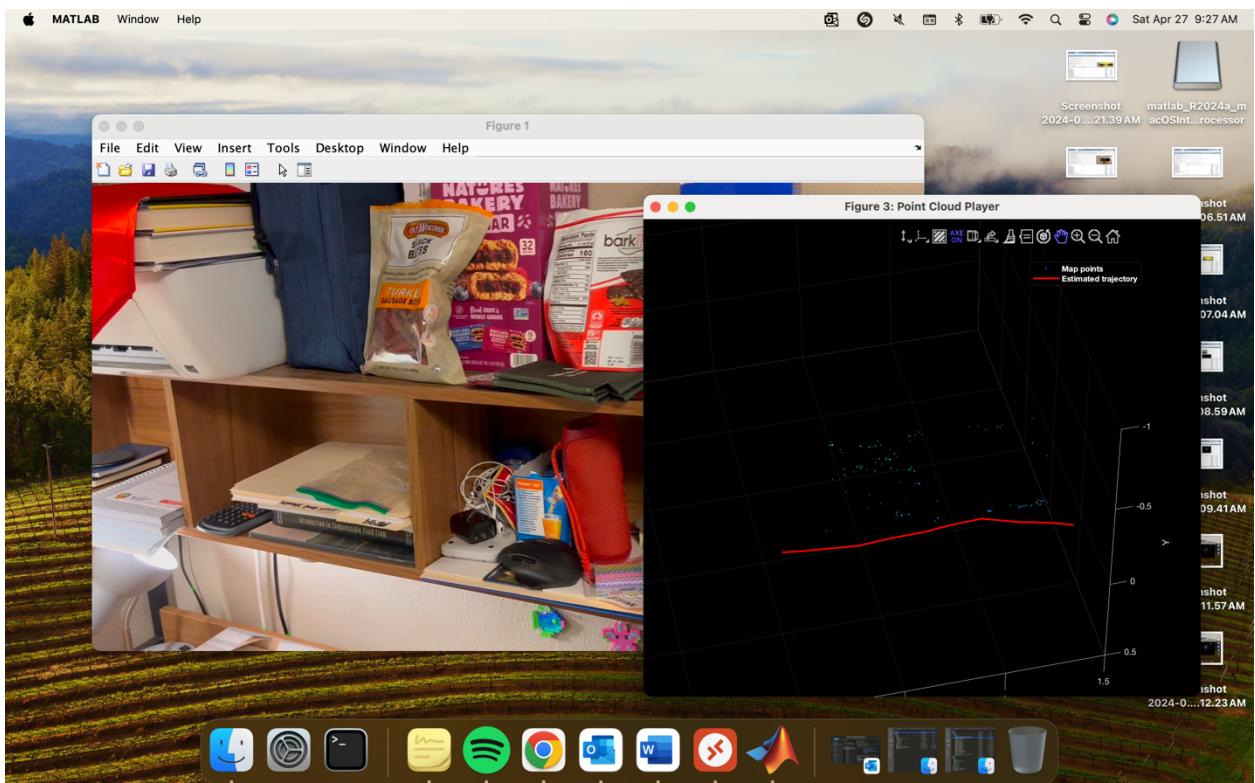
Command Window
f1 >>

```

Continuing onto the next section and its output,



Finally, we continue onto the next section, which was the tracking section and its output,



Unfortunately, MATLAB stopped its analysis after 20 or so key frames. I tried to resolve this issue, but with limited time it would have been too complex and time consuming. The issue probably lies within the image data itself. The raw video was over a minute and a half long with more than 2600 frames which seemed to be an issue for MATLAB's computational capacity. It could have also been my computer as well. My MacBook Pro is starting to get worn down. The texture of the photos is also relatively complex and there are a lot of objects within them as well. If I had time to redo it, I would create a much simpler scene and make the raw video much shorter.