Problem: The video *fiveCCC.mp4* shows a target composed of five concentric contrasting circle (CCC) features. The dimensions of the target (in inches) are 4.55×7.4. The origin of the target's coordinate system is in the middle of the rectangle, with its x-axis pointing to the right, its y-axis pointing down, and its z-axis pointing into the page.

1. Read and display each image of the video, and print the frame number on the displayed image.
2. Write a program that finds the five CCCs in each image, and also finds the correspondence between the image features and the model features.
3. Find a picture that you like and map it onto the target plane in each image, using a homography (projective transform). The picture should look like it is attached to the plane of the target.
4. Find the pose of the target with respect to the camera in each frame of the video. Draw the XYZ coordinate axes of the target as an overlay on the image, and also print the pose values on the image, in terms of translation (in inches) and the rotation vector.
5. Create an output video of your results, and post it to Google Drive.

Use the following values for the camera intrinsic parameters: focal length = 531 (in pixels), image center (x,y in pixels = $(320, 240)$. Assume no lens distortion.

I used the code I had for the second homework, "HW2-Q4.py", which detected the 5 CCCs in the video, as well as the Lab 6 code, "Group11-Lab6.py". For the homography addition, I downloaded a Mines logo from https://www.mines.edu/webcentral/logos/ and determined the points to map to the 5 CCCs based on the height h and width w of the Mines logo image. The logoPoints for the homography are $[[0,0];[w/2,0];[w,0];[0,h];[w,h]]$, which correspond to the (x, y) coordinates as appear in points 0-4 in the target. I could not get the "order-targets.py" function to work consistently, so I found an alternative. After the thresholding, connected components, and centroid matching occurs based on code done in "HW2-Q4.py", I matched the first five centroids to the 0-4 points. They did match exactly, so that defined the exact image coordinates (OrderedPoints) for the target in the frame. For subsequent frames, the centroids that deviated less than 10 pixels from the image coordinates in OrderedPoints were identified as the new OrderedPoints. The homography was done given the logoPoints and OrderedPoints. The same technique as used for "Group11-Lab6.py" was implemented where the nonzero pixels of the homography were overlayed onto the same pixels in the video frame. Next, the solvePnP command was called to determine the pose given the TargetPoints (true coordinates of 5 CCC image) and OrderedPoints (5 CCC coordinates in video frame). Next, the projectPoints command was called to draw the XYZ axes for the pose of the target in the frame. Finally, the strings for the rotation vector, translation vector, and frame number were added to the image. The resulting video can be found here: https://drive.google.com/file/d/1ZqxkkKDtXsUNF0oz_5kAZHU_l_fmvvig/view?usp=sharing.