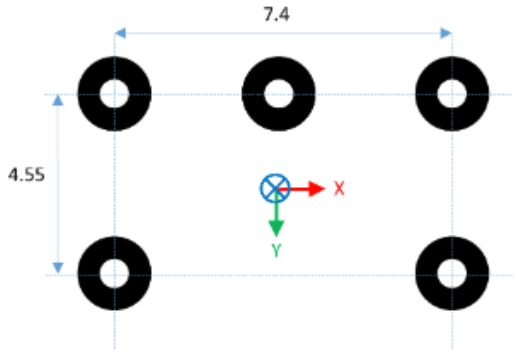
Lab 6: Find Pose from CCC Targets

Due Oct 9, 2020 by 11:59pm **Points** 10 **Submitting** a file upload

The task for today is to use OpenCV to detect a target composed of five contrasting concentric circles (CCCs) in an image, and compute the pose of the target. The dimensions (in inches) of the target pattern are shown in the figure.



The input image is “CCCtarget.jpg”, and is shown below. Assume that the camera that took this image has focal length equal to 531.0 pixels (in both x and y), image center at (x=320.0, y=240.0), and no lens distortion.

Finding correspondence

In a previous homework, you used binary image operations to detect CCC targets in an image. Now, you must order the detected targets, meaning that you have to figure out which CCC is which, and put them into the correct order, namely

```
0 1 2
3   4
```

The Python function “orderTargets()” (**order_targets.py** (https://elearning.mines.edu/courses/25410/files/2220631/download?download_frd=1)) performs this operation. It takes a list of target centroids and tries to determine the correct order. The strategy is to first find a collinear triple of targets, and assume that these are targets 0, 1, and 2. Then, it finds the closest remaining one to target 0, and assumes that this is target 3. Then, it finds the closest remaining one to target 2, and assumes that this is target 4. Finally, it checks to make sure that targets 3 and 4 are on the right hand side of the line going from 0 to 2; and if not it renumbers all the targets. If “orderTargets()” finds five targets in the correct configuration, it returns a list of five target centroids in the correct order. Otherwise, it returns an empty list.

Finding pose

The next step is to compute the pose of the target, using OpenCV’s “solvePnP()” function. You will need to pass in two sets of corresponding points:

- The five 3D points on the model, in a numpy array of size (5,3). Each row of the array is contains the X,Y,Z coordinates of the point (in inches).
- The corresponding five 2D points in the image, in a numpy array of size (5,2). Each row of the array contains the x,y coordinates of the point (in pixels).

If successful, the function will return the pose as a rotation vector and translation vector.

To do

- Read in the input image.
- Detect CCC targets, using your methods from HW2.
- Call “orderTargets()” to find a set of five CCCs in the correct order.
- Draw the correct label number (i.e., 0,1,2,3,4) on each CCC target in the image.
- Compute the pose by calling OpenCV’s “solvePnP()” function.
- Draw the coordinate axes on the image, by calling OpenCV’s “projectPoints()”.
- Finally, write the pose onto the image using OpenCV’s “putText()” function.

When writing the pose onto the image, don’t write the numbers with too many digits, as it will clutter up the image. For translation, just use one significant digit to the right of the decimal point. For the rotation vector, use two significant digits to the right of the decimal point.

Turn in

- The image with CCC targets labeled, coordinate axes drawn, and the pose written onto the image.
- Your Python program.

Hint: The correct pose for this image is somewhere in the following ranges.

- For translation, X is between 5 and 10, Y is between -20 to -5, and Z is between 70 to 80.
- For the rotation vector, the X component is between -0.4 to -0.1, the Y component is between 0 and 0.05, and the Z component is between -0.9 and -0.7. (UPDATE TO THIS: IGNORE THE RANGES FOR THE X,Y COMPONENTS)