1. Camera Robot Calibration: The approach taken for recognizing the edges of the petri dish is through converting the image into a black& white image using *imbinarize()* and using the *imfindcircles()* function to generate a list of circles. *imbinarize(src,' adaptive')* was used as this would account for the general threshold levels that would have to be defined and calculated for *iim2bw()* to be utilized. Utilizing the approximate pixel width of the Petri dish, it is used as an input range for the *imfindcircles()* function. We assume that the largest circle in the image would be that of the Petri dish. (Due to the design of the workspace). Hence, it is outputting the center of the Petri dish.

2. Find and Segment the Robot with Color Detection:
    a. Segmenting the colors: Converting the images from RGB to HSV provides the largest contrast and easily finds the color ranges of the robot.
    b. Finding the HSV values range: To find the red and blue regions of the robot, we would have to know the 'Hue', 'Saturation', and 'Contrast Values' of the regions. This was accomplished by writing a custom script *HSV_Value_Extractor.m.* The script allows the user to select the area and crop it by pressing enter. Next, select all the points of interest, and it will output the max and min values of HSV.
    c. Connecting the regions: Next would be to create a mask that would select the regions and *bwconncomp()* would isolate the regions and find the connected areas. Then *regionprops()* would ensure the area, centroid, and bounding box of the various regions are found. There could be some regions of discontinuity due to filter values not being completely precise, hence it would be viable to find the weighted centroid.
    d. Finding the weighted centroids of the color region: The weighted centroid is calculated using:

$$\bar{x} = \frac{\bar{x}_t A_t + \bar{x}_{s_1} A_{s_1} + \bar{x}_{s_2} A_{s_2} + \bar{x}_{s_3} A_{s_3}}{A_t + A_{s_1} + A_{s_2} + A_{s_3}}$$

Repeating the same for both blue and red areas. To find the centers of the red and blue region. After this, the two points are joined, and the midpoint is treated as the center of the robot, and the angle of the line is known as the theta of the robot.

3. Finding the location of the robot: Finding the x and y coordinates with respect to the origin and multiplying with the scalar multiple found between petri dish dimensions and pixels would allow us to find the actual coordinates of the robot.

4. We found the standard deviation of the processing time to be around 0.021s, and the average computational time is 0.11 seconds.
5. Found the error between the calculated and manually keyframed using *imtool()* coordinates to be an average of 4.12 %. The error is caused by the following reasons:
    a. Due to the lighting causing effects in particular frames, not allowing for clean HSV conversion and ranges to be accounted for.
    b. Acceleration of the robot is causing some changes with respects to regions.