Report

Yusuf Can Anar

Python 3.7 is utilized.

Purpose :
We are asked to find the given steps below in Figure 1

1 – Detect where the reds are the brightest.
2 – Find the bars from the detected locations.
3 – Detect only the one we are interested from the bars found.
4 – Find corner coordinates of the bar we detected in the previous step and calculate euclidean distance between the corners.

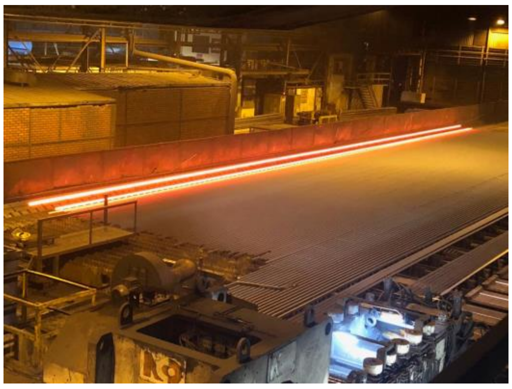


Figure 1

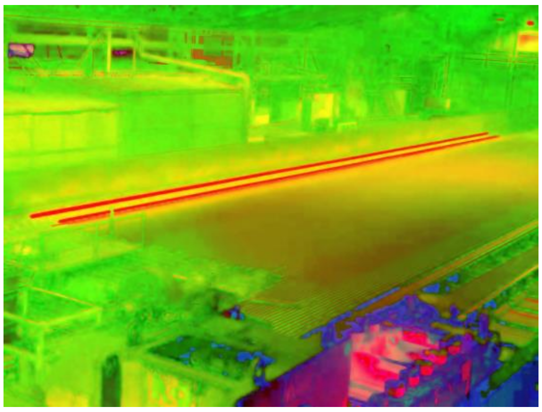
Program Explanation :

1) Every "viewImage" function call is going to be a station to display the images, so we can see every change in the image. Every "blue_mask" function is to have a clear sight of what we found.

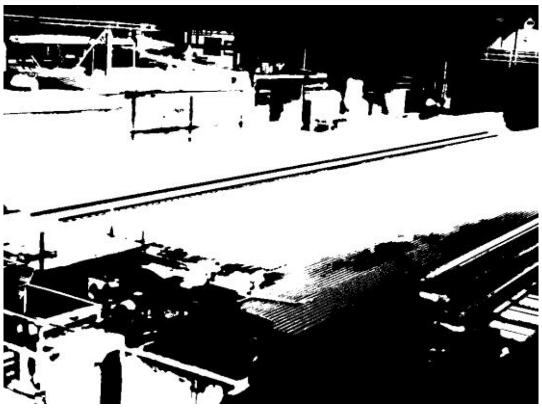
2) Input image(Figure 1) is assigned to "image" variable using openCV function "cv2.imread()".

3) Denoised "image" is assigned to the "img"(It is not necessary but can be implemented to get rid of small regions).

4) “img” is converted BGR to HSV to be able to create a mask and HSV image is assigned to “hsv_img”.

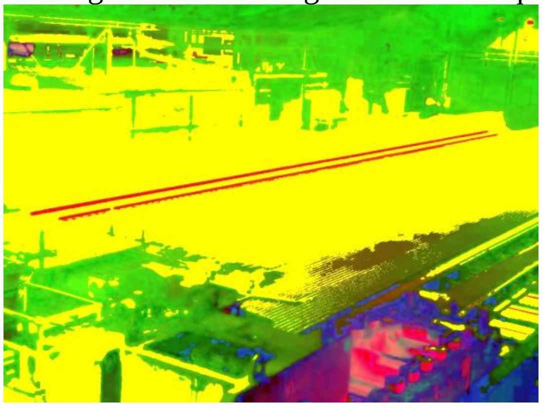


5) Current mask has red low and red high values that I have set to detect the red bars in the “hsv_img”.

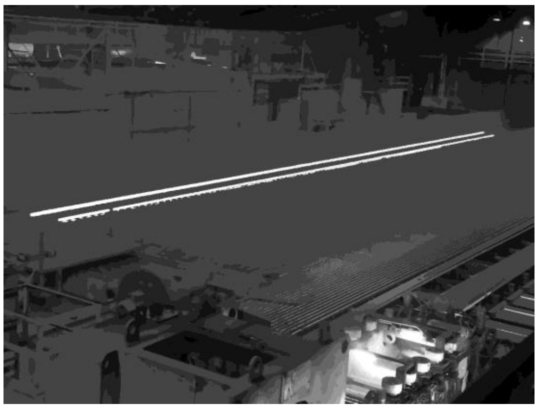


6) Then I have applied the current mask to the “hsv_img” as:

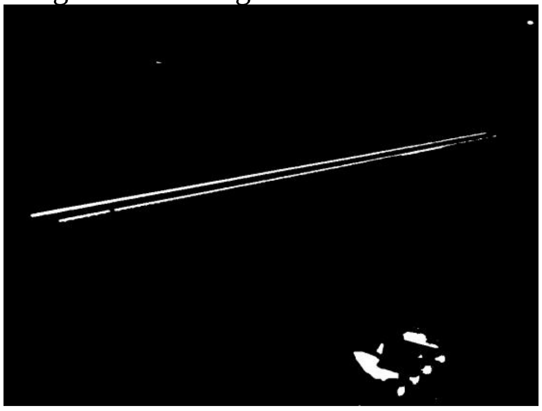
hsv_img[curr_mask > 0] = ([0,255,255]) by this step “curr_mask” ‘s elements that is True for the condition for pixel value outputs the True value which means in the “hsv_img” ‘s same elements are assigned to the given value on right side of the equation.



7) Converting the image HSV to RGB so we can turn the image RGB to gray form using “cv2.cvtColor()” function. By this step after detecting red locations we need to lesser the variables in the image which means we need gray image to specify background and foreground or objects of RGB image which is input image that’s why we can’t use hsv to gray directly.

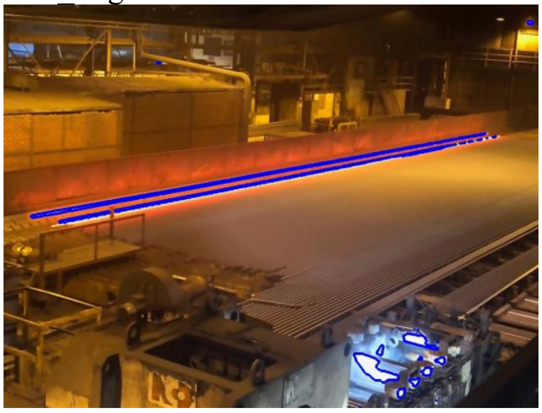


8) Threshold the gray image so the background now all black and objects are white.

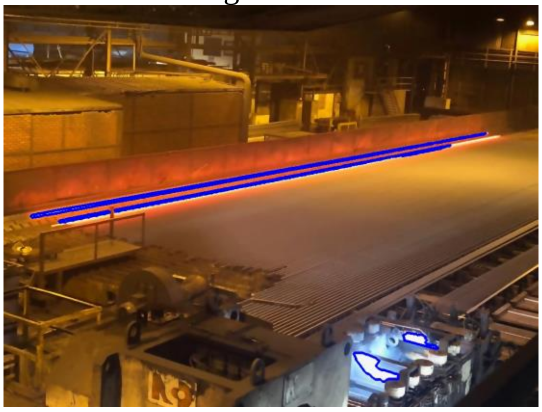


9) Using “cv2.findContours()” function we are getting a list of contours in the thresholded image which means white object contours.

10) Using “cv2.drawContours()” function, we can draw all the contours found in the thresholded image in the “_img” variable which is same as the input image.

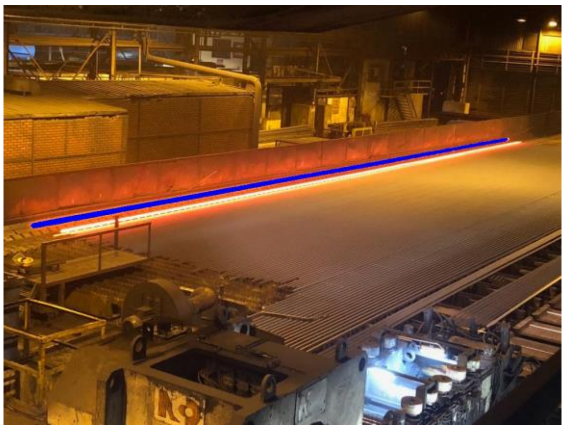


11) “remove_small_regions” , this function calculates all contour areas, finds their average area to eliminate the unwanted small regions.



12) “find_first_bar()” function detects only the first bar on the image by using the row coordinates. Since image coordinates starts at (0,0) which is the top left corner on the screen we can search for minimum row coordinates of the contours after removing small contours. The contours that has the minimum row coordinate is the interested bar. And we can draw only the insterested contour/bar in blue.

13) Using “cv2.fillConvexPoly()” by this function, we can fill in the contour with blue color to cover all the insterested bar.



14) We used “blue_mask()” function and we inputted “b_mask” which has only interested contour in white in the “finding_corners_and_euclidean_distance()” function. This function uses the “cv2.goodFeaturesToTrack()” function to detect the corners in the bar we found and assigned it “corners_img” array which has the all corner coordinates of the masked image.

15) Finally “cv2.boundingRect()” using openCV function, as we input the “corners_img” coordinates in to the function, we can get x,y, width and height as output of the function.

Function detects the corners in the object and fits a rectangle to the object. By this coordinates and lengths we can easily calculate the euclidean distance.