BBM 418 - Computer Vision Laboratory
Programming Assignment-1 Report

Spring 2020
Assoc. Prof. Dr. Nazlı İkizler Cinbiş

Arda Hüseyinoğlu
21627323

Department of Computer Engineering, Hacettepe University
April 3, 2020

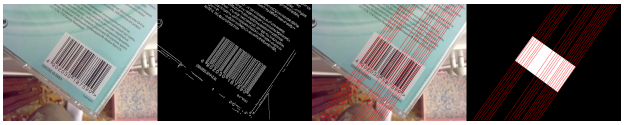
1 Explanation

First of all, file paths of all images in the original image directory and detection image directory which includes ground truth images are stored in two different arrays. Since the file name of original image and corresponding ground truth image are same, those images are in the same order in original image and detection directories. Thus, image addresses in arrays at same index are chosen and `detectLinesOnBarcode` function are called to detect barcode lines on original image by means of corresponding ground truth image. In `detectLinesOnBarcode`, these two addresses are read to get original and ground truth image. Original image is converted to gray before calling `cv2.Canny` which uses canny edge detector algorithm to find edges on gray scaled image. Edge detector returns a binary 2-d edge map. Since edge map does not only include edges on barcode area but other edge points in image, `determineEdgePointsInBarcode` is called to check each detected edge point on edge map if it takes place on barcode area. In other words, each pixel intensity on ground truth image corresponding to edge point location on edge map is checked if it is 255 (white). If so, these coordinates are stored and so all possible edge points on barcode area are obtained.

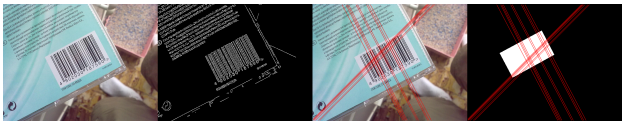
After getting these edge points, `HoughTransform` function is called. Firstly, an hough accumulator array is created to store votes for each (ρ, θ) pair by calling `createAccumulatorArray`. Range of θ values is chosen as $[-\pi/2, \pi/2]$ with 1 pixel resolution. Then, maximum rho value which is diagonal of the image is computed and range of ρ values is chosen as $[-\rho_{max}, \rho_{max}]$ with 1 pixel resolution.After getting accumulator array, `findNumberOfVotes` are called. Accumulator array keeps track for its every location how many time sinusoidal curves belonging to each edge point pass those locations.For rows in accumulator array(y-axis): ρ_{max} corresponds to first row, 0 corresponds to row in the middle and ρ_{max} corresponds to last row. So, half of the value of ρ_{max} is added to every derived ρ value from the equation of $\rho = x.\cos(\theta) + y.\sin(\theta)$ to have rho_{max} started 0^{th} index. For column in accumulator array(x-axis), $-\pi/2$ corresponds to first index and $\pi/2$ corresponds to last index. To get $\cos(\theta)$ and $\sin(\theta)$ values that corresponds to each θ value easily, `getCosSinThetaValues` function is used. After getting rho values for each theta, each location in accumulator array corresponding (ρ, θ) pair is increased by 1. When ρ values are computed, I used numpy array feature to decrease compute time around 15 seconds to 3-5 seconds multiplying edge point coordinates by $\cos(\theta)$ and $\sin(\theta)$ at once without iterating every edge point for same θ value.

After voting is done, `findMaxVotedLocations` function is called to find maximum voted (ρ, θ) pair in accumulator array. My procedure to find threshold value is to get 30 most voted bins first. Then, mean value of them is computed. This mean value is determined as threshold. Lastly, each location is checked if it is greater than the threshold value. If so, that (ρ, θ) pair is stored. If the mean value is greater than 120, then 30 most voted bins; if it is greater than 130, then 40 most voted bins are selected automatically. It is an improvement for the images whose lines on barcode can be detected well. Lastly, lines are drawn to the original image and ground truth image in according to their rho and theta values by calling `drawLinesToImage` function for each image. Then, `showOutput` function is called to concatenate 4 image horizontally and show them as output.

2 Result



(a) 05102009081.png



(b) 05102009082.png

(c) 05102009216.png

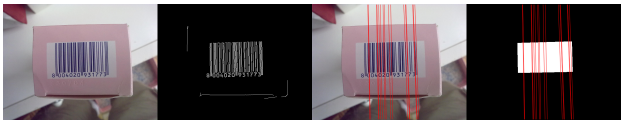
(d) 05102009120.png

(e) 05102009133.png

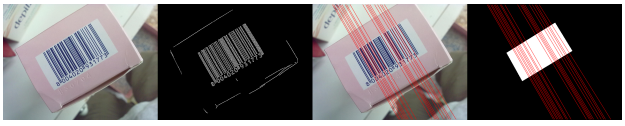
(f) 05102009138.png

(g) 05102009139.png

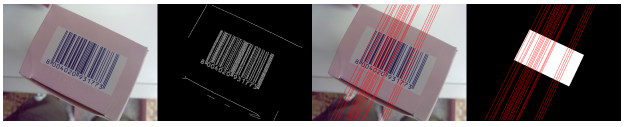
(h) 05102009140.png



(i) 05102009153.png



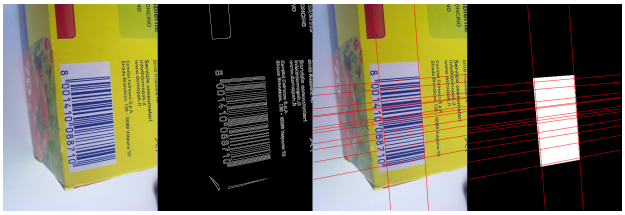
(j) 05102009154.png



(k) 05102009155.png

(l) 05102009196.png

(m) 05102009083.png



(n) 05102009229.png