Programming Assignment 1

Mustafa Sercan AMAÇ, Hacettepe University 16/03/2020

Edge Detection

I used Canny's Method for Edge Detection using openCV's implementation. The method assumes that the step edges are corrupted by a gaussian noise. The first step is to clear the noise with a gaussian filter. After extracting the gradient magnitude of the image, non-max suppression is applied to the image. Then we apply hysterisis thresholding to the gradient image.

Listing 1: Canny's Edge Detection

```python
from __future__ import print_function
import cv2 as cv
class CannyMethod(object):
    def __init__(self,max_lowThreshold=100, ratio=3, kernel_size=3):
        self.max_lowThreshold = max_lowThreshold # hysterisis thresholding
        self.ratio = ratio
        self.kernel_size = kernel_size # gaussian filter size
    def getEdgeMap(self,src,src_gray,threshold):
        low_threshold = threshold
        img_blur = cv.blur(src_gray, (3, 3)) # clear noise
        detected_edges = cv.Canny(img_blur, low_threshold, low_threshold *↩
             self.ratio, self.kernel_size) # get edge map
        mask = detected_edges != 0
        dst = src * (mask[:, :, None].astype(src.dtype))
        return detected_edges, dst # return edge_map along with original ↩
            image
```

Hough Transformation

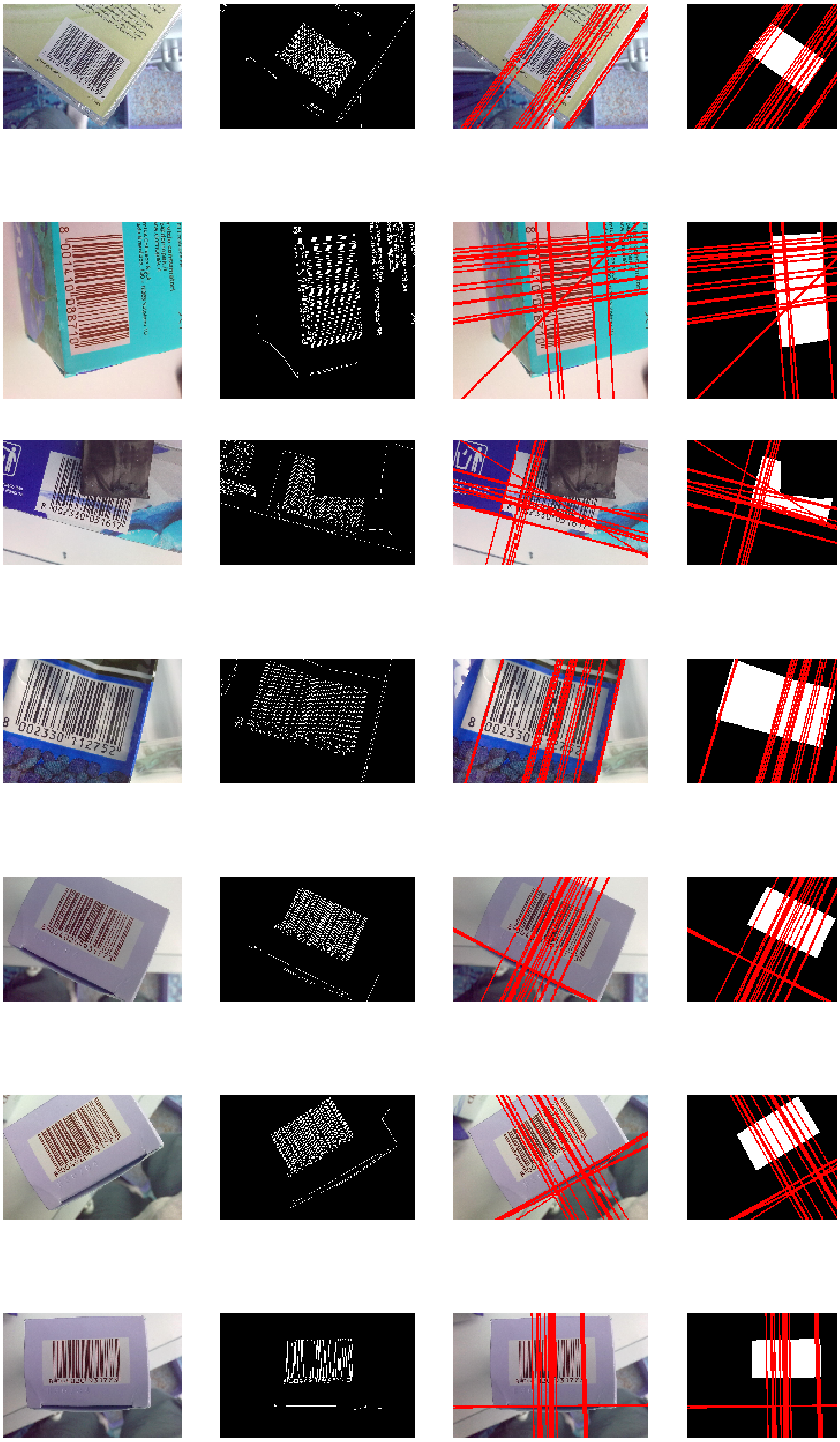
Hough transformation is a method to detect lines in an image.When this method is applied on an edge map it basically rotates a line on a pixel point and votes for each angle and rho value in polar hough space.A simple observation is when 2 points are on the same line, they will vote on the same polar coordinates.That is why hough transformation can be very effective for line detection. In my implementation it has 4 params, number of edges to draw, minimum and maximum values of angles to rotate and a pixel threshold for the pixel value of an edge.

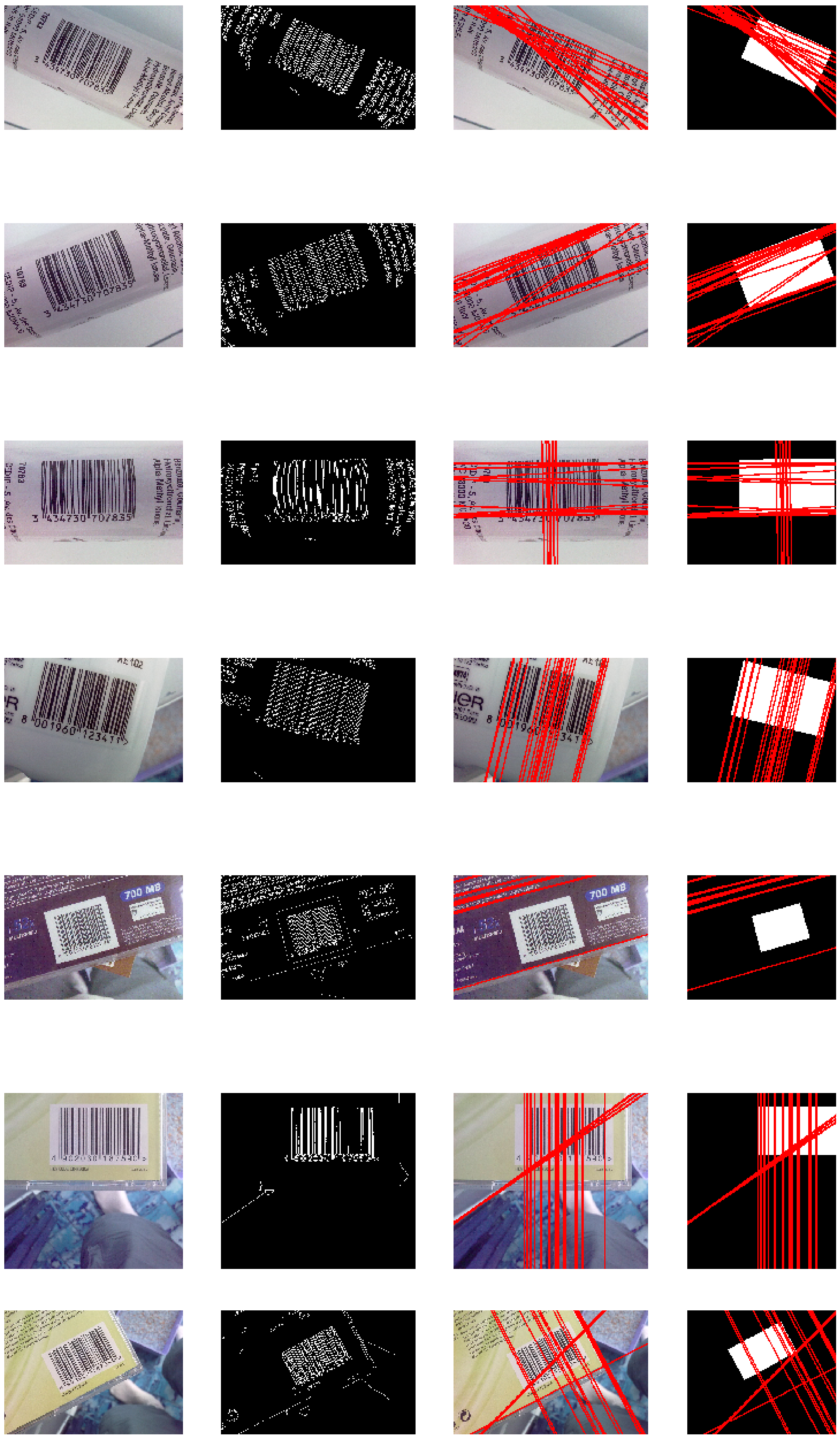
Listing 2: Hough Transformation Implementation

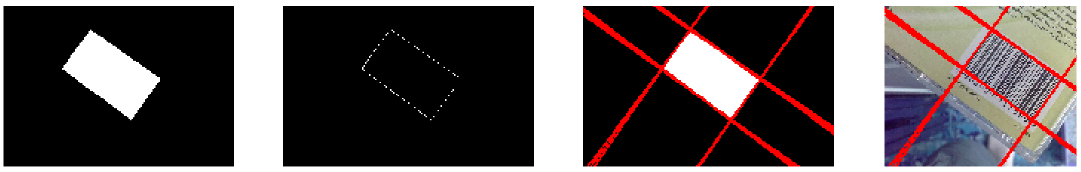
```python
import numpy as np
import math
class Hough(object):
    def __init__(self,n_edges,theta_min = -90,theta_max = 90,pix_threshold=10):
        self.thetas = [i for i in range(theta_min, theta_max+1)] # angles
        self.n_edges = n_edges # max n_edges to draw
        self.pix_threshold = pix_threshold # consider pixels whose value is over this
    def vote(self,img):
        self.rmax = int(math.hypot(img.shape[0],img.shape[1])) # the maximum value rho can get.
        self.hough_space = np.zeros((len(self.thetas), 2 * self.rmax + 1)) # This is the hough space that we will vote.

        for x in range(img.shape[1]): # the X and Y coordinates in an image is different thats why x == img.shape[1]
            for y in range(img.shape[0]):
                if img[y, x] > self.pix_threshold:
                    for i, theta in enumerate(self.thetas): # rotate the line
                        th = math.radians(theta)
                        ro = round(x * math.cos(th) + y * math.sin(th)) + self.rmax # we add r_max to get rid of negative values for indexing.
                        if ro <= 2 * self.rmax:
                            self.hough_space[i, ro] += 1 # vote
    def get_lines(self): # This method simply returns top n_edges in the hough space.
        return self.topk(self.hough_space,k=self.n_edges)
    def topk(self,a, k):
        idx = np.argpartition(a.ravel(), a.size - k)[-k:]
        return np.column_stack(np.unravel_index(idx, a.shape))
    def reset(self): # reset if needed
        self.hough_space = np.zeros((len(self.thetas), 2 * self.rmax + 1))
```

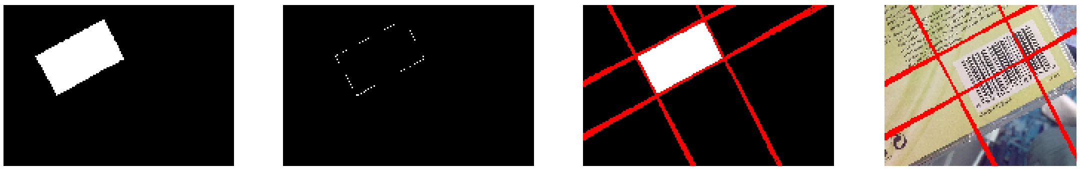
Results

Here are my algorithm's results;



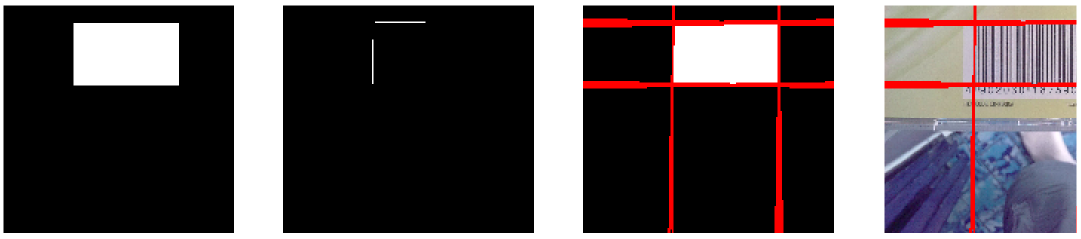


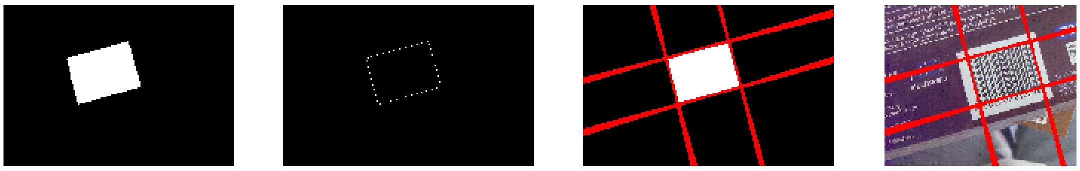


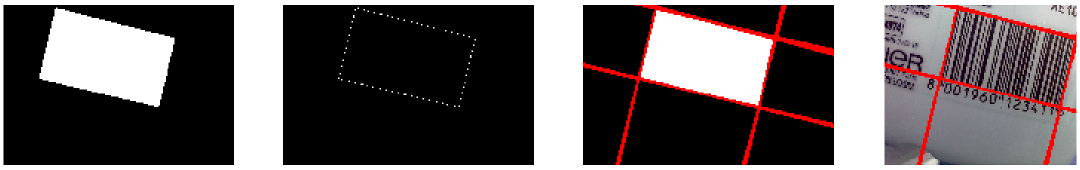


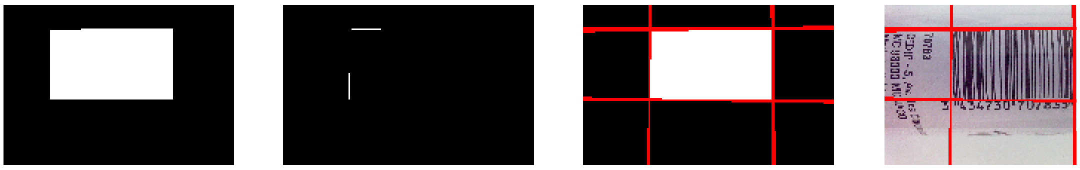
Another Perspective

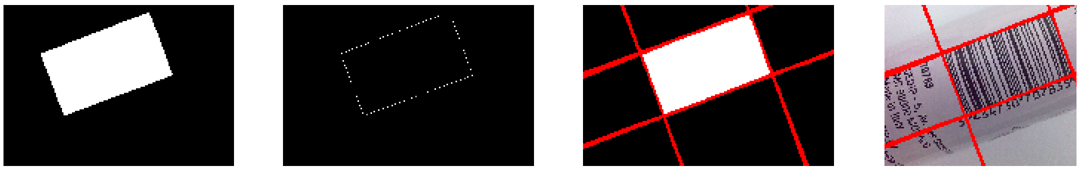
When I research about barcode detection I did come up with a paper that does barcode detection. It was simply detecting barcode boundaries with a neural network.(Corresponding to detections in our dataset.). So I thought there is a better application for our dataset. And that is drawing bounding boxes on barcodes. These are the results when i applied Canny's method and hough transformation on detection images and draw the lines on the original image; Here are my algorithm's results;

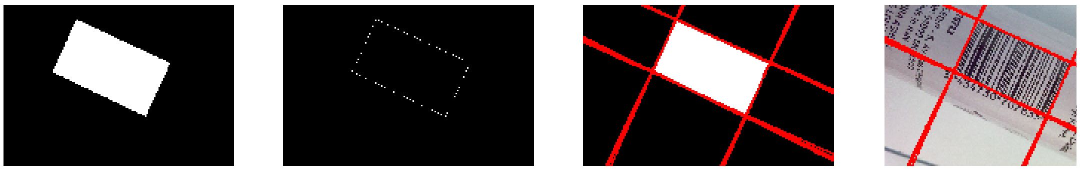


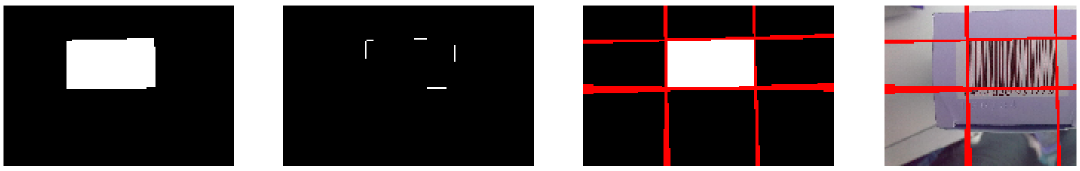


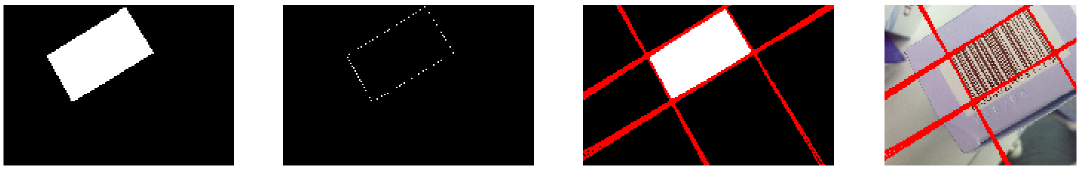


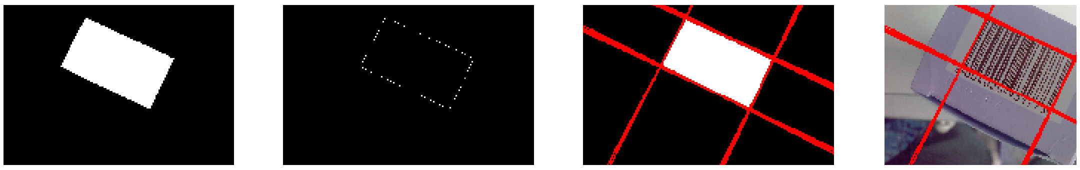


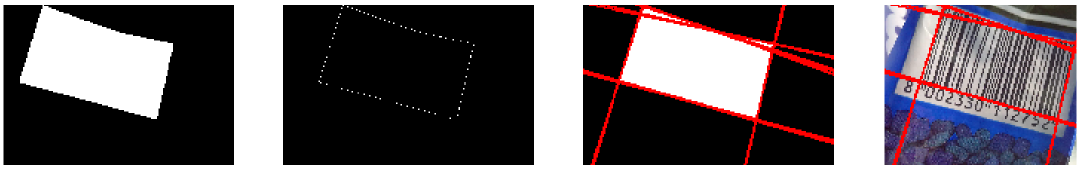


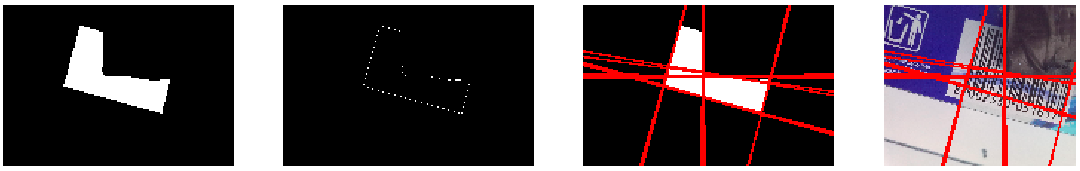


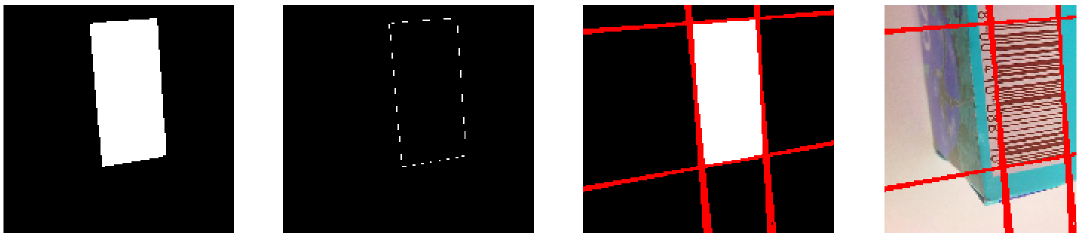












Comments

In the results section we can observe that some images are affected by the other edges in the image and doesn't draw good lines. A simple solution would be to set a higher hysterisis thresholding value and get rid of edges other than that barcode has. In another perspective section i did apply hough transformation on segment images and obtained really good results for drawing bounding boxes on barcodes. In both sections we can observe that this simple algorithm can be quite useful for detecting lines on a edge map.