Introduction to Computer Vision-Assignment 1

Cumhuriye Tulug Kucukogut

S025791

This project evaulates the rectangles from image and determines the image consists of sudoku or not. For reaching of the result many algorithms,transformations and detections are used.

INTRODUCTION

In this project, https://github.com/wichtounet/sudoku_dataset/tree/master/images ‘s repo is used as a dataset. Many images are used for sampling. Python’s numpy and opencv libraries are used for implementation. Project is created from 3 main steps that are finding biggest area, determining lines and finding the rectangles. After all these steps checking the image that consists of sudoku or not.

To find the biggest area and rectangles; countor, for detecting lines; Hough transform are used. All these implementation parts are explained in the Background.

For starting the project:

The path should be given by the user then you can compile it. Images coming one by one, when you want to see another image, you should press any key. The final parts are shown which are rectangles and determination of sudoku.

BACKGROUND

For implementation of this project opencv and numpy libraries are installed. Numpy is powerful for arrays and opencv is used for computer vision.

As explained in the introduction, the project consists of 3 main steps.

1)Finding Biggest Area

Images can contain many different objects, according to the dataset the main part of the images is sudoku so sudoku’s biggest area is found then it is subtracted from other parts of the image. For finding the main part, different techniques are used.

Segmentation

Thresholding is used for segmentation. Segmentation provides to divide an image to regions according to the different features like color. Thresholding converts image to binary And it is implemented on grayscale images. For this reason after reading the image by imread method, image is converted to the grayscale.

```
image = cv2.imread(images[i])
img = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
```

Before usage of the Canny, Gaussian Smoothing method is used. Gaussian smoothing helps to remove noises. It is used with kernel and according to the documentation kernel should be odd and positive.Bigger values give better results.

```
img = cv2.GaussianBlur(img, (5, 5), 0)
```

Threshold is divided into 3 cathegories[1].
1)Simple Thresholding
2)Adaptive Thresholding
3)Otsu's Thresholding

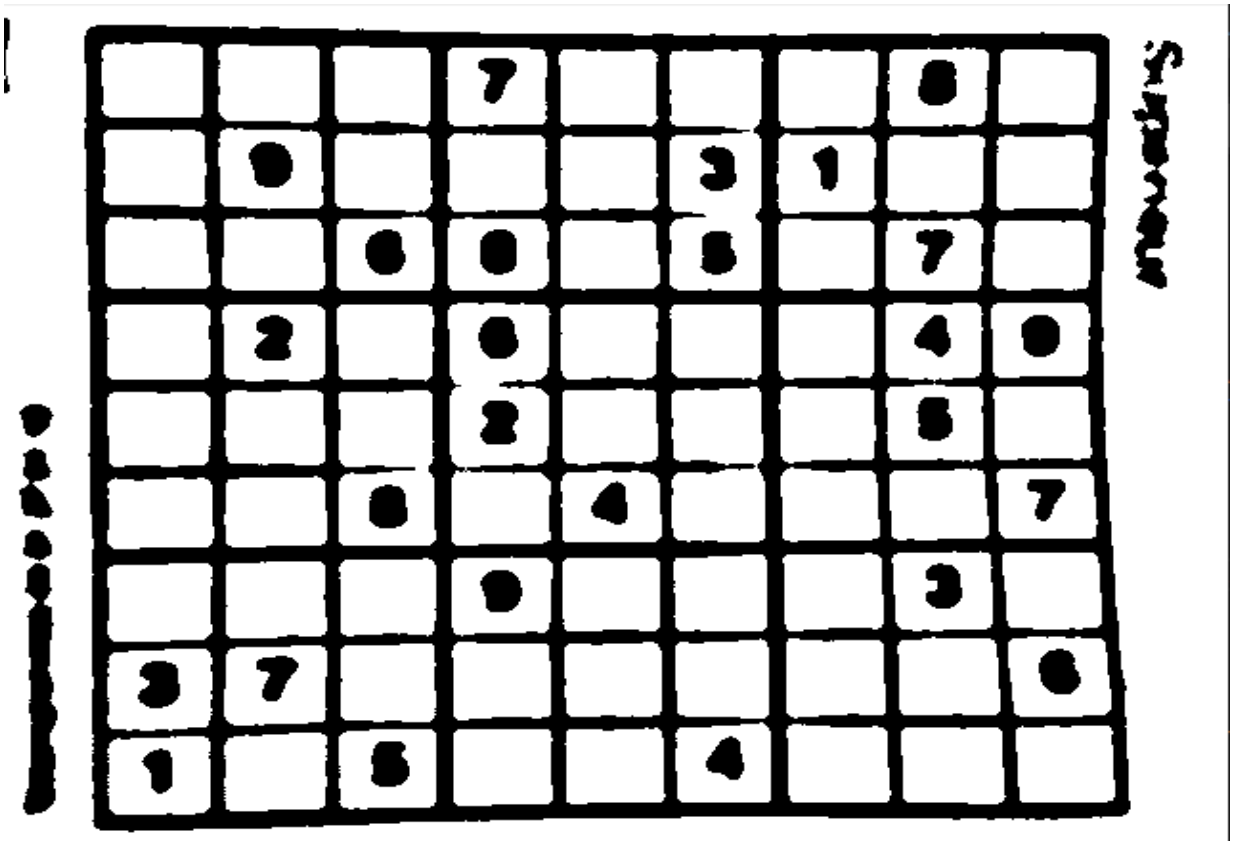
In this project, different images are used and all of them have different lighting conditions so decided to use adaptive thresholding that notrs the light factor[2]

adaptiveThreshold(src, dst, maxValue, adaptiveMethod, thresholdType, blockSize, C)

Adaptive method can be gaussian or mean;
cv2.ADAPTIVE_THRESH_GAUSSIAN_C gives better results according to the cv2.ADAPTIVE_THRESH_MEAN_C so Gaussian is used for images

thresh = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 51, 5)

Gaussian provides gentler smoothing and preserves edges better than a similarly sized mean filter. This two adaptive methods are tried in the images ADAPTIVE_THRESH_GAUSSIAN_C gives better results.[3]

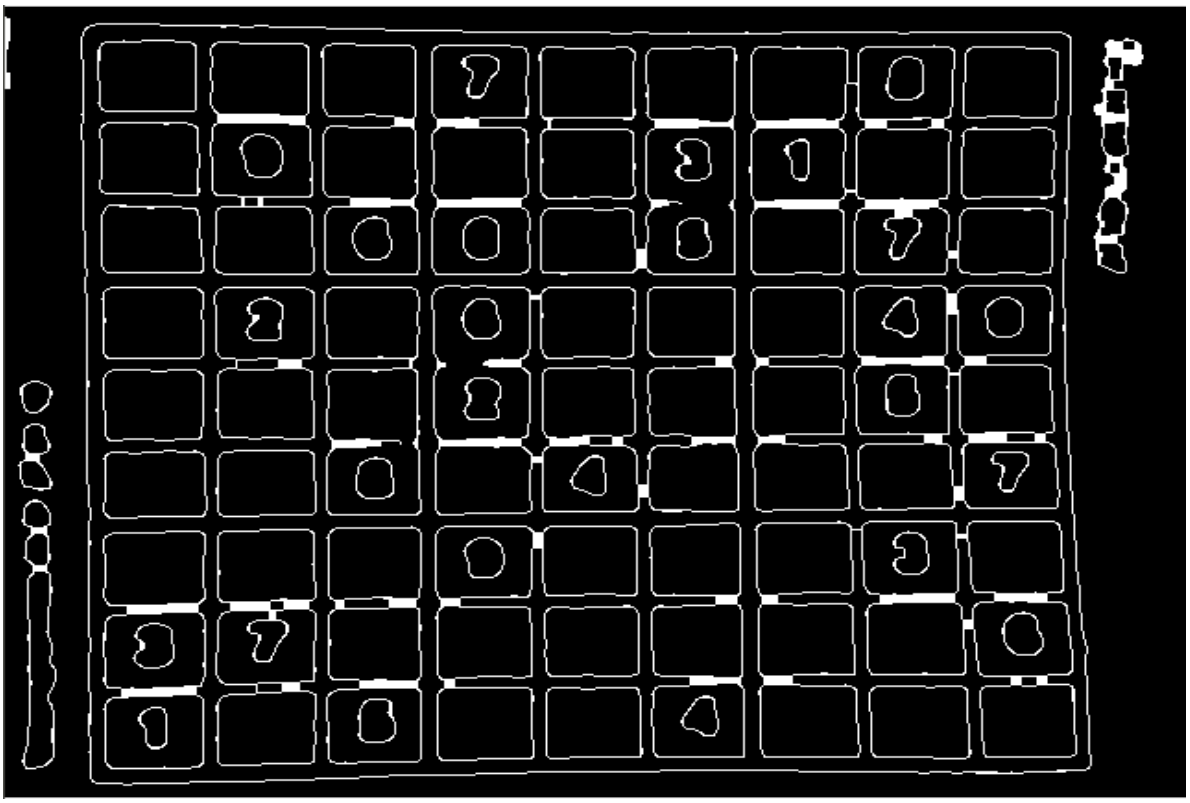


Finding edges

For finding the biggest area lines should be clear The Canny filter helps to find the strongest edges and get cleaner contours[4].Canny is used generally used for contour estimation

```
edges = cv2.Canny(thresh, 10, 30, apertureSize=3)
edges = cv2.dilate(edges,kernel,iterations = 1)
edges = cv2.erode(edges,kernel,iterations = 1)
```

After the detection, lines look noisy and some of lines are broken so equalizing for all images some morphological transformations are used. Dilation increases the white region in the image.In this project it is usedfor filling the broken lines. Erosion is the synonym of dilation.It removes small anomolies[5]



Contour

Contours are used for shape and object detection. A curve joining all the continous points[4]
In this project finding biggest area,all areas are compared and big one is selected.Firstly mask is created and biggest area is drawn on this via drawcontours method.
Mask and image is conjucted to each other via bitwise operator.

Bitwise operators are useful for extracting any part of image. And operation provides conjuction on images. It calculates the per-element bit-wise conjunction of two arrays or an array and a scalar.[5]

After the conjuction, founded greatest area is substracted via mask from background

```
contours,hier = cv2.findContours(edges,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
if contours is not None:
cnt = None
max_area = 0
for c in contours:
area = cv2.contourArea(c)
if area > max_area :
cnt = c
max_area = area

cv2.drawContours(mask, [cnt], -1, 0, -1)
mask = 255 - mask
result = cv2.bitwise_and(image, mask)
result[mask==0] = 255
```

2)Determining Lines

Determining Edges

For finding lines, vertical and horizontal edges are found via morphological operations which are erosion and dilation. Dilation adds pixels to the boundaries of the object in an image, while erosion does exactly the opposite. The amount of pixels added or removed, respectively depends on the size

and shape of the structuring element used to process the image. In general the rules followed from these two operations have as follows[6]
For vertical lines;
Kernel is created and height size should be greater than the width

```
verticalKernel = np.ones((25,1),np.uint8)
horizontalKernel = np.ones((1,25),np.uint8)
```

Hough Transform For Lines

In this project finding the lines, firstly mask is created and lines are observed. In hough transform two values are created which are rho and theta start points and finish points are dependable on them. Implementation this algorithm many lines are detected. X1s are starting point for vertical lines and Y1s are starting point for horizontal lines
For these operations

```
lines = cv2.HoughLines(edges,1,np.pi/180,200)
for rho,theta in lines[0]:
    a = np.cos(theta)
    b = np.sin(theta)
    x0 = a*rho
    y0 = b*rho
    x1 = int(x0 + 1000*(-b))
    y1 = int(y0 + 1000*(a))
    x2 = int(x0 - 1000*(-b))
    y2 = int(y0 - 1000*(a))

    cv2.line(img,(x1,y1),(x2,y2),(0,0,255),2)
```

methods are used

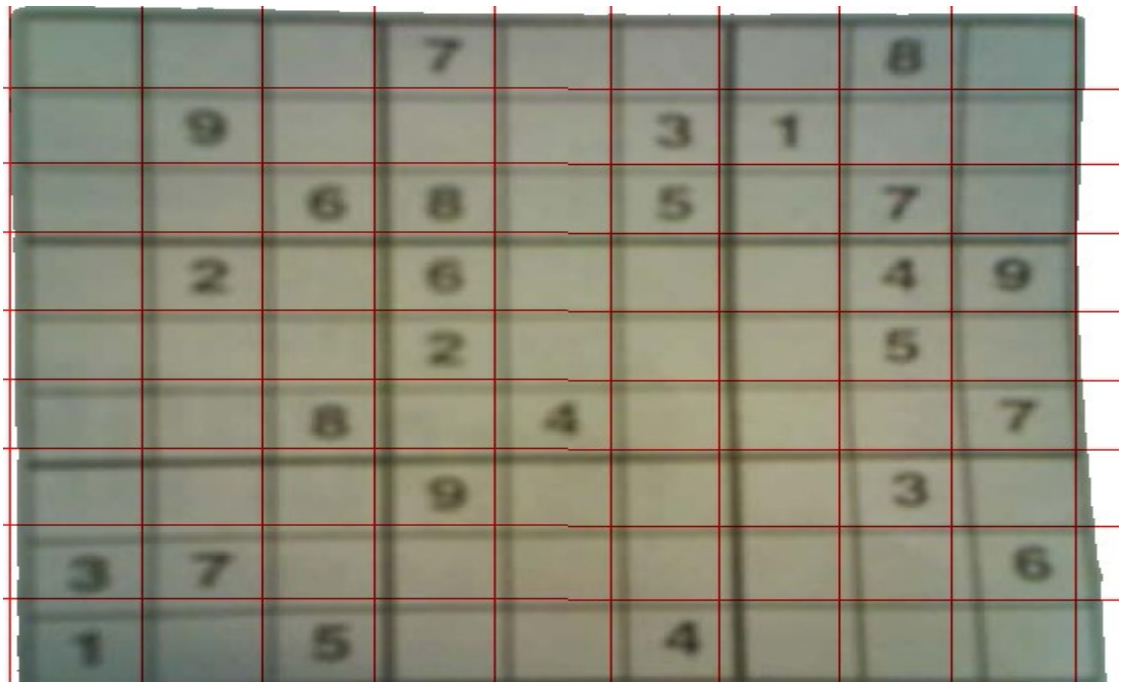
Groups of the Lines

After finding the lines, many lines are created more then excepted in some images. The reason of the thickness of the lines. Thick line provides minimum two lines so starting points are grouped according to the closeness.

Closeness are calculated according to the distance between starting points. Distance list is created and average of differences are calculated. Then compared all starting points according to the average. All these operations are implemented in calculateCloseNumbers method. Returned value is generated according to the closeness of the starting points.

Conjuction of Lines

Bitwise_and operator is used twice firstly used for concatenation of vertical and horizontal lines then created lines concatenated with main image with lines.



Creating rectangles

After specifying the lines of the image, contour is used for finding rectangles.approxPolyDP bounding rects methods are used.

ApproxPolyDP approximates the contour shape, if the result of the shape is 4 it gives the rectangle.Also it gives an approximation about the shape via epsilon value. In this project different epsilon values are tried and 0.1 gives the better solution

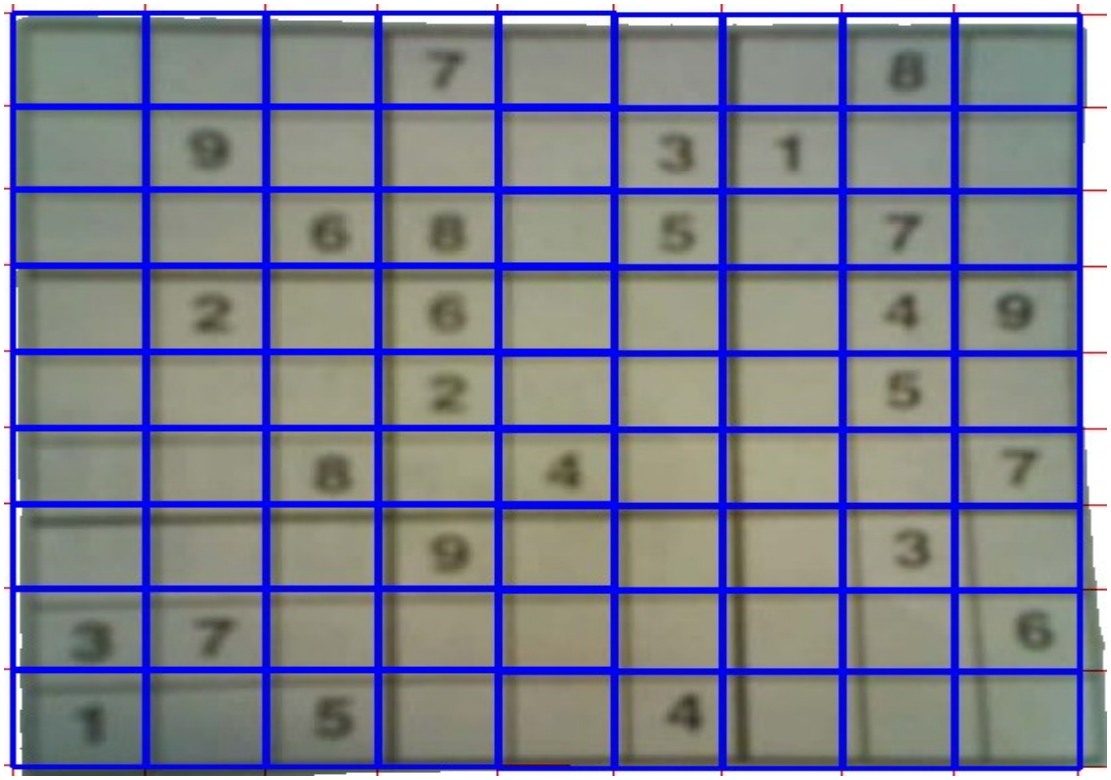
Calculation of approximation,boundingRect method is used for finding rectangles.

It is divided into two groups but rotation is not important for that project so straight bounding rectangle is more useful

x,y,w,h = cv.boundingRect(cnt)

cv.rectangle(img,(x,y),(x+w,y+h),(0,255,0),2)

Then implementation according to the sudoku;sudoku consists 10 vertical and 10 horizontal lines also 81 rectangles all of them are calculated in the implementation via contours and decided it sudoku or not



References

1)https://www.geeksforgeeks.org/python-thresholding-techniques-using-opencv-set-1-simple-thresholding/#:~:text=Thresholding%20is%20a%20technique%20in,maximum%20value%20(generally%20255).
2)https://docs.opencv.org/master/d7/d4d/tutorial_py_thresholding.html
3)https://homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm
4)https://medium.com/sicara/opencv-edge-detection-tutorial-7c3303f10788
5)https://www.geeksforgeeks.org/difference-between-dilation-and-erosion/
6)https://docs.opencv.org/3.4/dd/dd7/tutorial_morph_lines_detection.html