Make It Aesthetic

Anna-Maria Hohmann
Konrad Kring
Meret Lindanis

4th presentation

03.12.2019

About Us

Meret

- computer science
- medical technologies

Anna

- business informatics
- project management

Konrad

- pedagogics
- music

Goals of Our Project

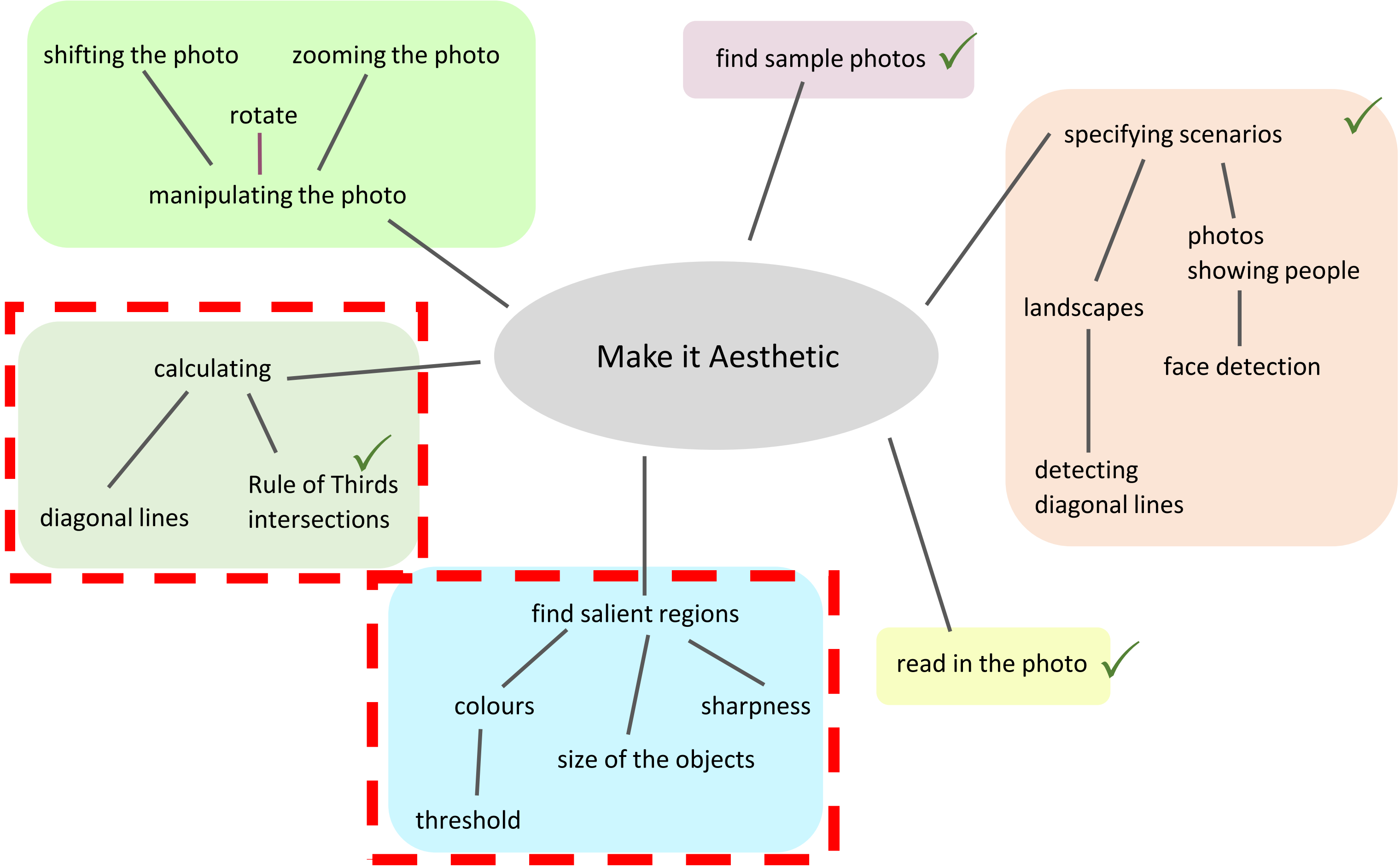
our motivation:

- interested in photography
- opening aesthetic photography to the public
- simplifying the aesthetic photography for the user
- being able to save every moment in beautiful photos
- bringing this knowledge into school

Goals of Our Project

- make given photos aesthetic
- by zooming, rotating or cropping the photo
- selecting the guideline the photo should follow

Milestones



Make it Aesthetic

find sample photos ✓

specifying scenarios ✓

landscapes

detecting diagonal lines

photos showing people

face detection

read in the photo ✓

find salient regions

colours

threshold

size of the objects

sharpness

calculating

diagonal lines

Rule of Thirds intersections ✓

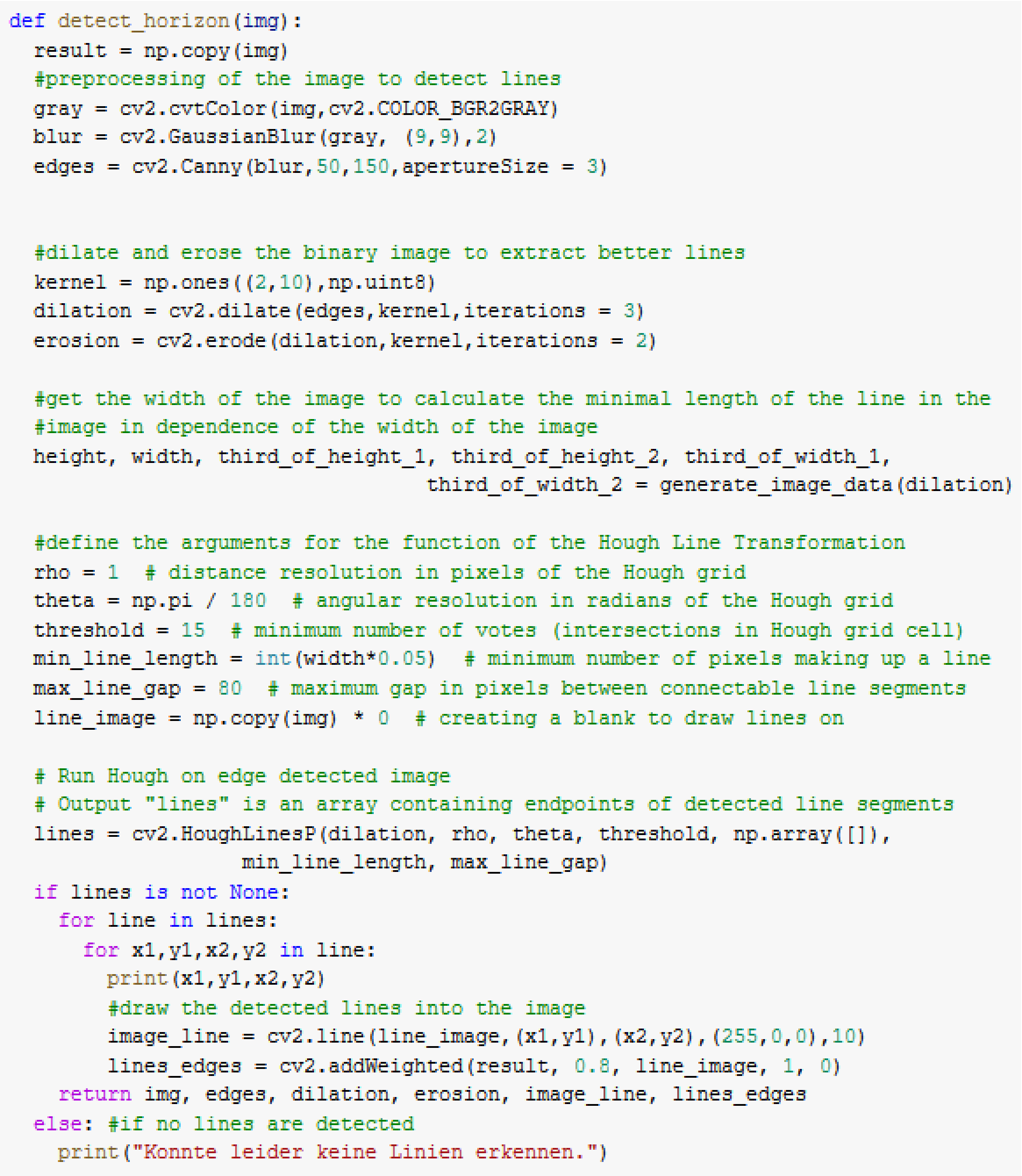
manipulating the photo

shifting the photo

rotate

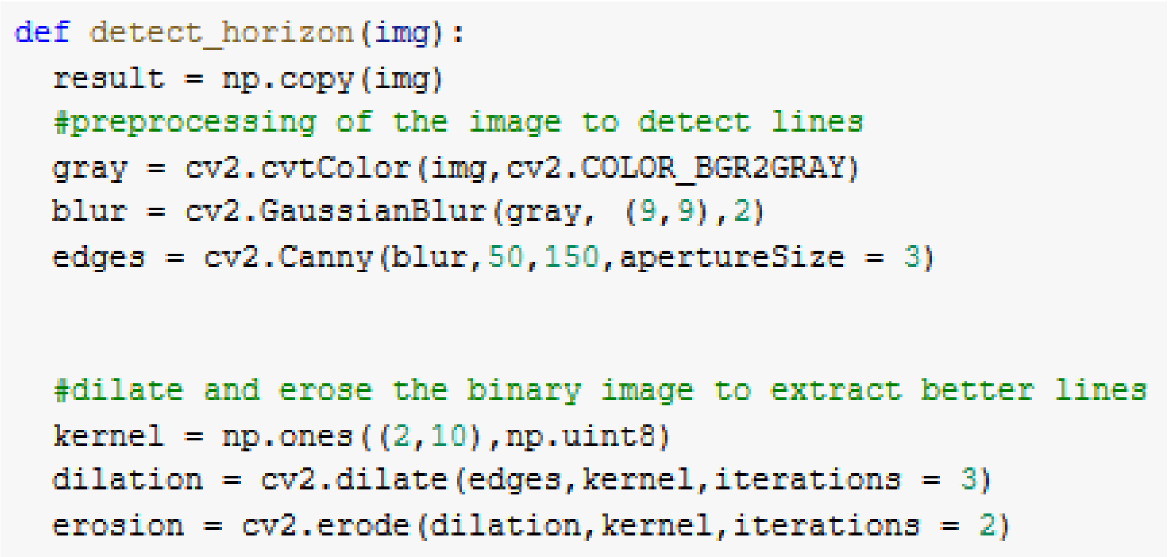
zooming the photo

Implementation



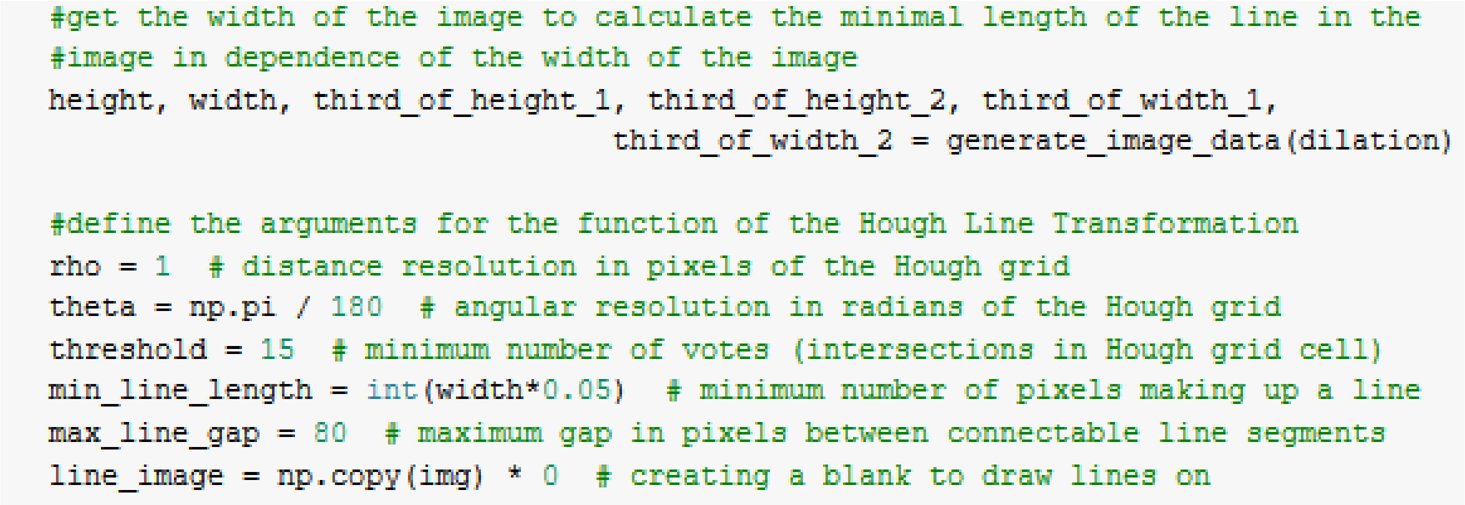
```
def detect_horizon(img):
  result = np.copy(img)
  #preprocessing of the image to detect lines
  gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
  blur = cv2.GaussianBlur(gray, (9,9),2)
  edges = cv2.Canny(blur,50,150,apertureSize = 3)


  #dilate and erose the binary image to extract better lines
  kernel = np.ones((2,10),np.uint8)
  dilation = cv2.dilate(edges,kernel,iterations = 3)
  erosion = cv2.erode(dilation,kernel,iterations = 2)

  #get the width of the image to calculate the minimal length of the line in the
  #image in dependence of the width of the image
  height, width, third_of_height_1, third_of_height_2, third_of_width_1,
                                  third_of_width_2 = generate_image_data(dilation)

  #define the arguments for the function of the Hough Line Transformation
  rho = 1  # distance resolution in pixels of the Hough grid
  theta = np.pi / 180  # angular resolution in radians of the Hough grid
  threshold = 15  # minimum number of votes (intersections in Hough grid cell)
  min_line_length = int(width*0.05)  # minimum number of pixels making up a line
  max_line_gap = 80  # maximum gap in pixels between connectable line segments
  line_image = np.copy(img) * 0  # creating a blank to draw lines on

  # Run Hough on edge detected image
  # Output "lines" is an array containing endpoints of detected line segments
  lines = cv2.HoughLinesP(dilation, rho, theta, threshold, np.array([]),
                    min_line_length, max_line_gap)
  if lines is not None:
    for line in lines:
      for x1,y1,x2,y2 in line:
        print(x1,y1,x2,y2)
        #draw the detected lines into the image
        image_line = cv2.line(line_image,(x1,y1),(x2,y2),(255,0,0),10)
        lines_edges = cv2.addWeighted(result, 0.8, line_image, 1, 0)
    return img, edges, dilation, erosion, image_line, lines_edges
  else: #if no lines are detected
    print("Konnte leider keine Linien erkennen.")
```

Implementation



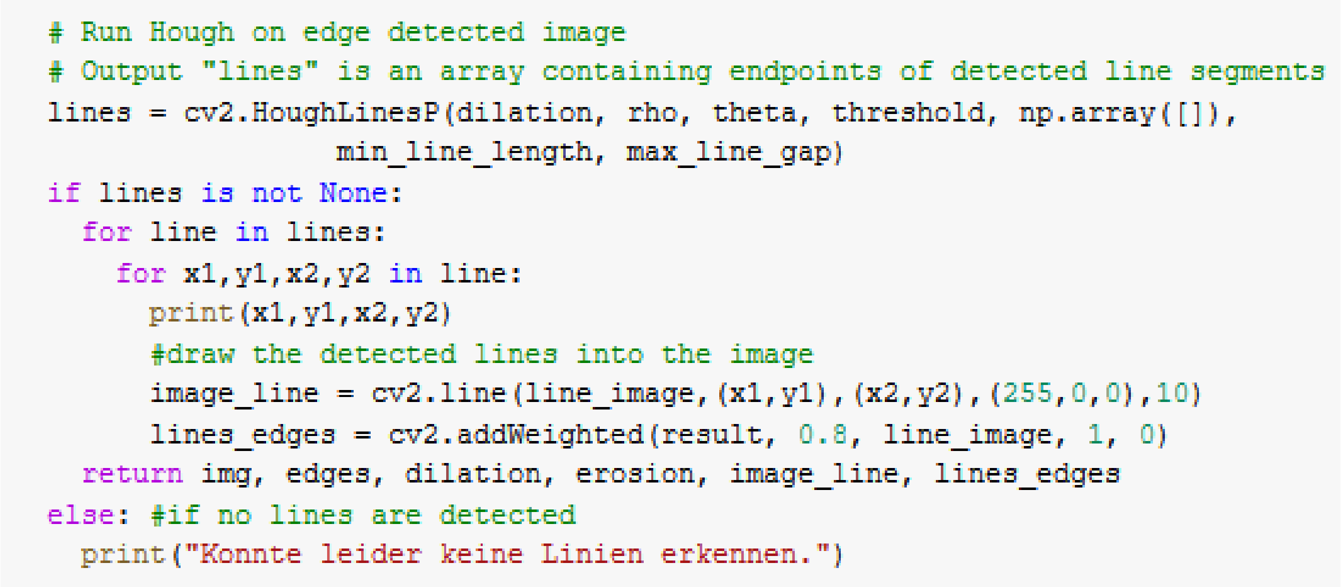
```
def detect_horizon(img):
  result = np.copy(img)
  #preprocessing of the image to detect lines
  gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
  blur = cv2.GaussianBlur(gray, (9,9),2)
  edges = cv2.Canny(blur,50,150,apertureSize = 3)


  #dilate and erose the binary image to extract better lines
  kernel = np.ones((2,10),np.uint8)
  dilation = cv2.dilate(edges,kernel,iterations = 3)
  erosion = cv2.erode(dilation,kernel,iterations = 2)
```

Implementation



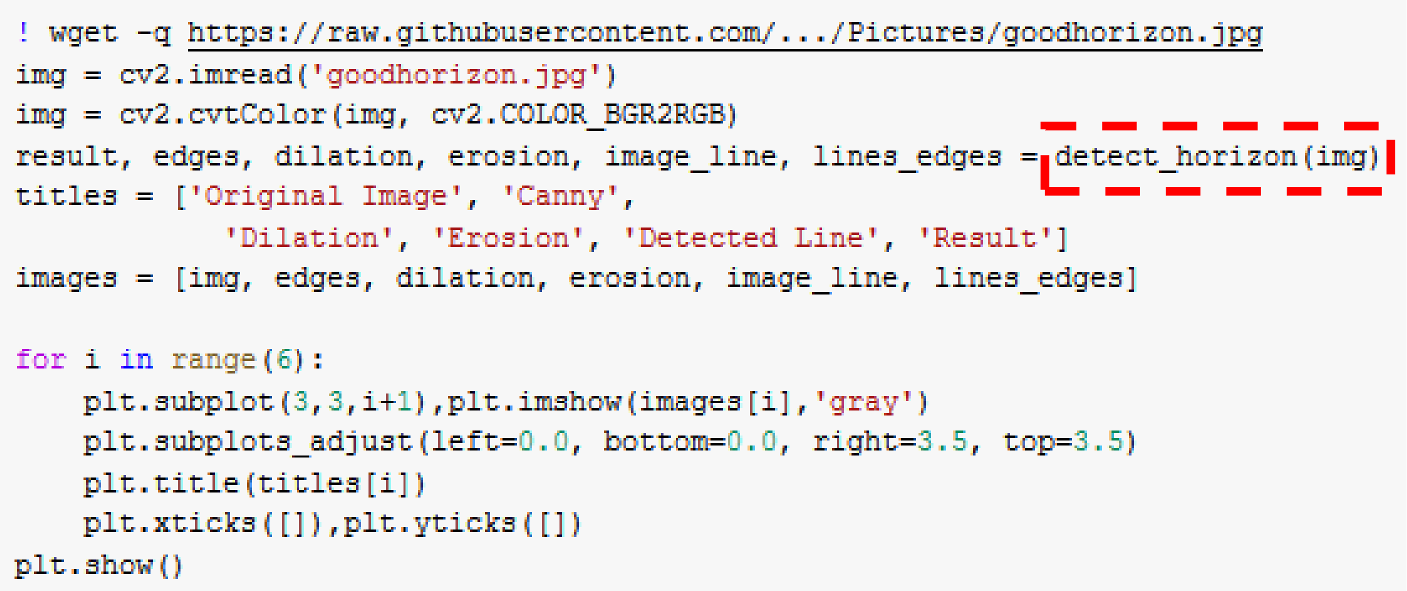
```
#get the width of the image to calculate the minimal length of the line in the
#image in dependence of the width of the image
height, width, third_of_height_1, third_of_height_2, third_of_width_1,
                                third_of_width_2 = generate_image_data(dilation)

#define the arguments for the function of the Hough Line Transformation
rho = 1  # distance resolution in pixels of the Hough grid
theta = np.pi / 180  # angular resolution in radians of the Hough grid
threshold = 15  # minimum number of votes (intersections in Hough grid cell)
min_line_length = int(width*0.05)  # minimum number of pixels making up a line
max_line_gap = 80  # maximum gap in pixels between connectable line segments
line_image = np.copy(img) * 0  # creating a blank to draw lines on
```

Implementation



```
# Run Hough on edge detected image
# Output "lines" is an array containing endpoints of detected line segments
lines = cv2.HoughLinesP(dilation, rho, theta, threshold, np.array([]),
                 min_line_length, max_line_gap)
if lines is not None:
  for line in lines:
    for x1,y1,x2,y2 in line:
      print(x1,y1,x2,y2)
      #draw the detected lines into the image
      image_line = cv2.line(line_image,(x1,y1),(x2,y2),(255,0,0),10)
      lines_edges = cv2.addWeighted(result, 0.8, line_image, 1, 0)
  return img, edges, dilation, erosion, image_line, lines_edges
else: #if no lines are detected
  print("Konnte leider keine Linien erkennen.")
```

Implementation



```
! wget -q https://raw.githubusercontent.com/.../Pictures/goodhorizon.jpg
img = cv2.imread('goodhorizon.jpg')
img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
result, edges, dilation, erosion, image_line, lines_edges = detect_horizon(img)
titles = ['Original Image', 'Canny',
            'Dilation', 'Erosion', 'Detected Line', 'Result']
images = [img, edges, dilation, erosion, image_line, lines_edges]

for i in range(6):
    plt.subplot(3,3,i+1),plt.imshow(images[i],'gray')
    plt.subplots_adjust(left=0.0, bottom=0.0, right=3.5, top=3.5)
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.show()
```

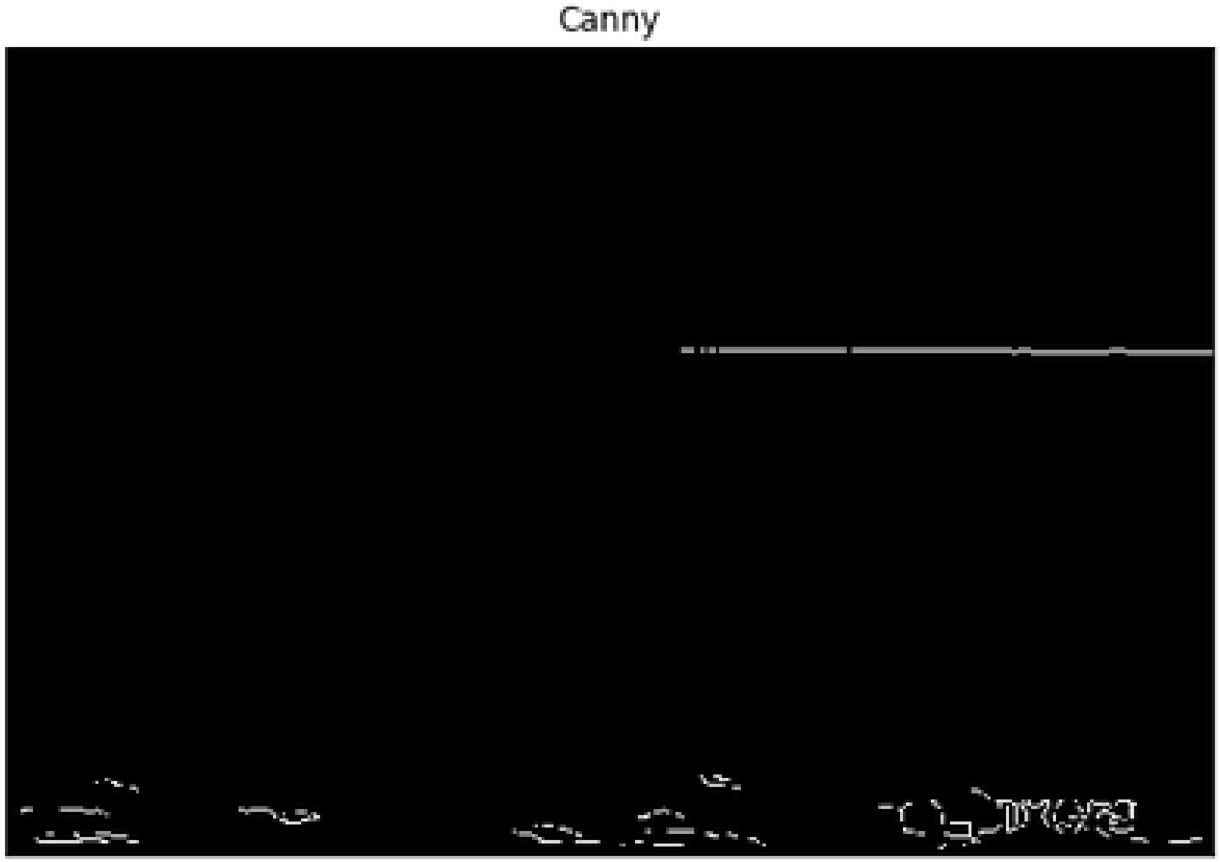
Implementation

Sample 1

[1]

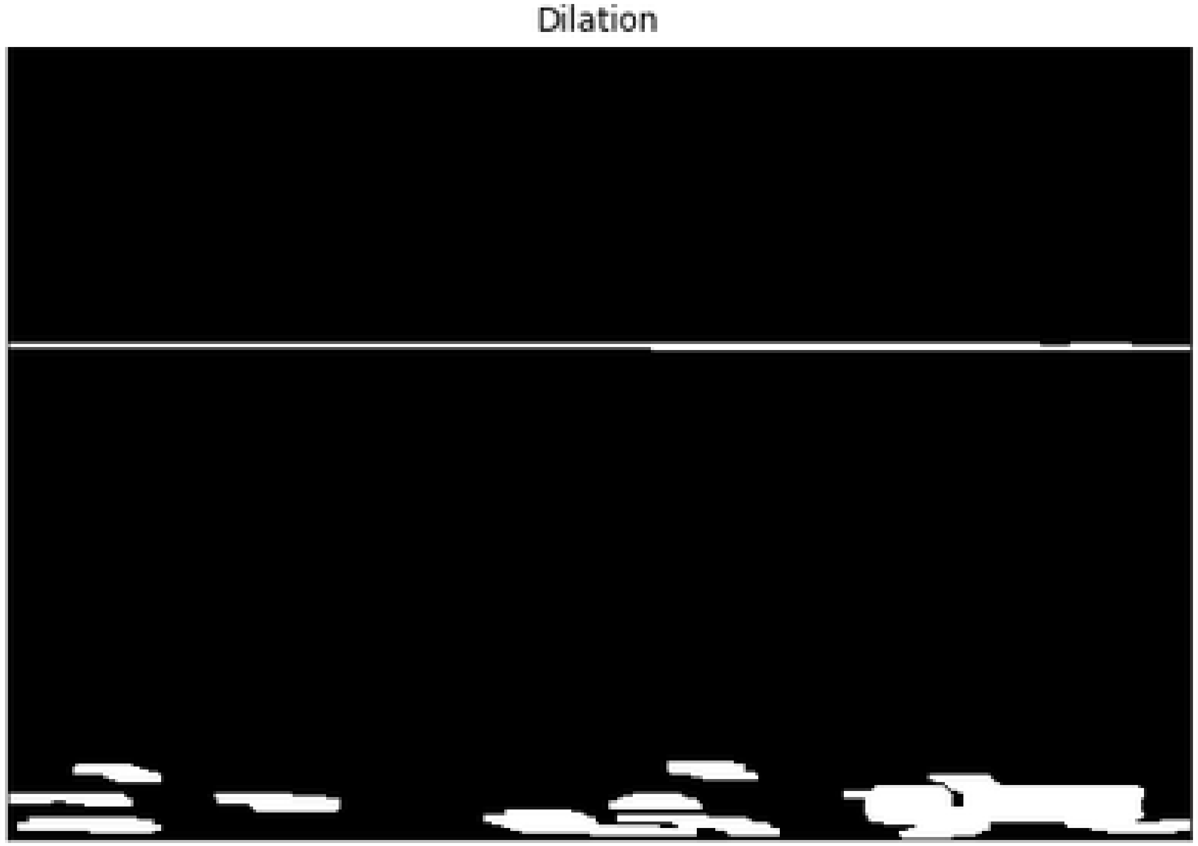
Implementation

Sample 1



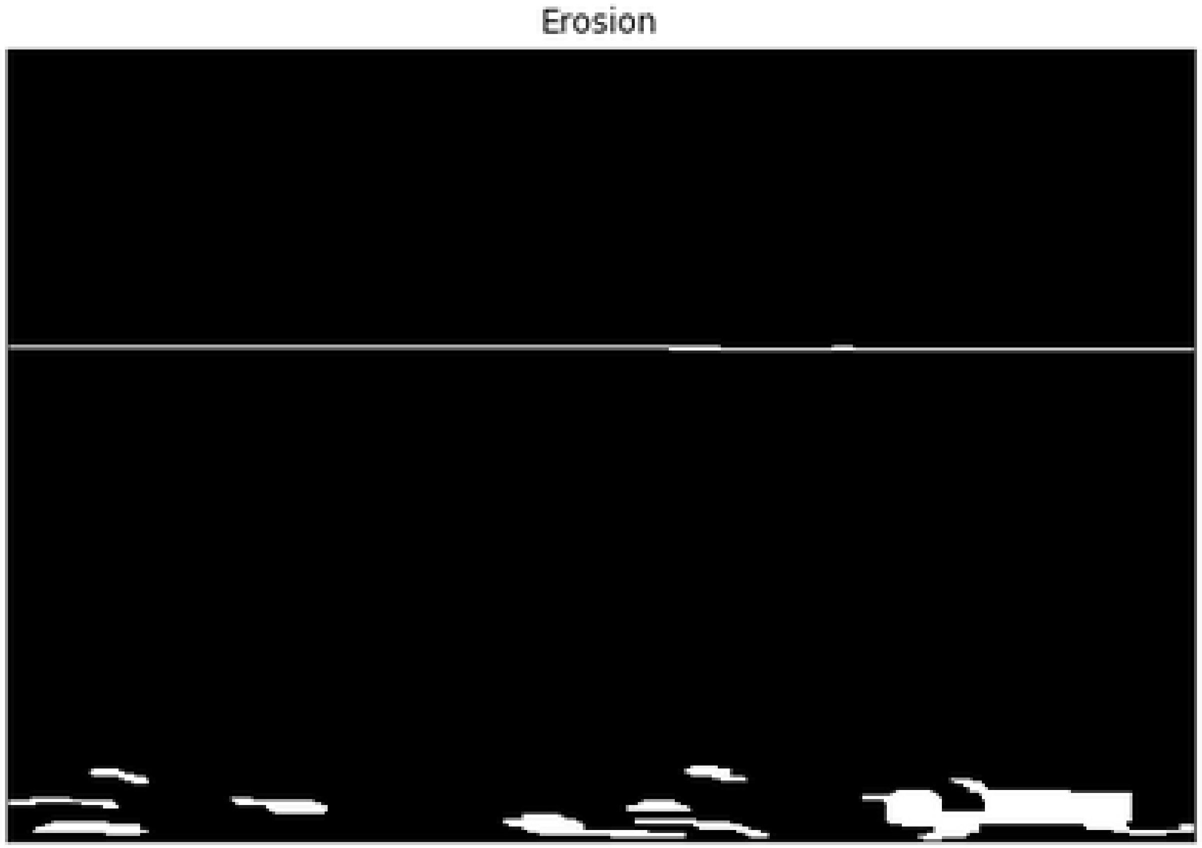
Implementation

Sample 1



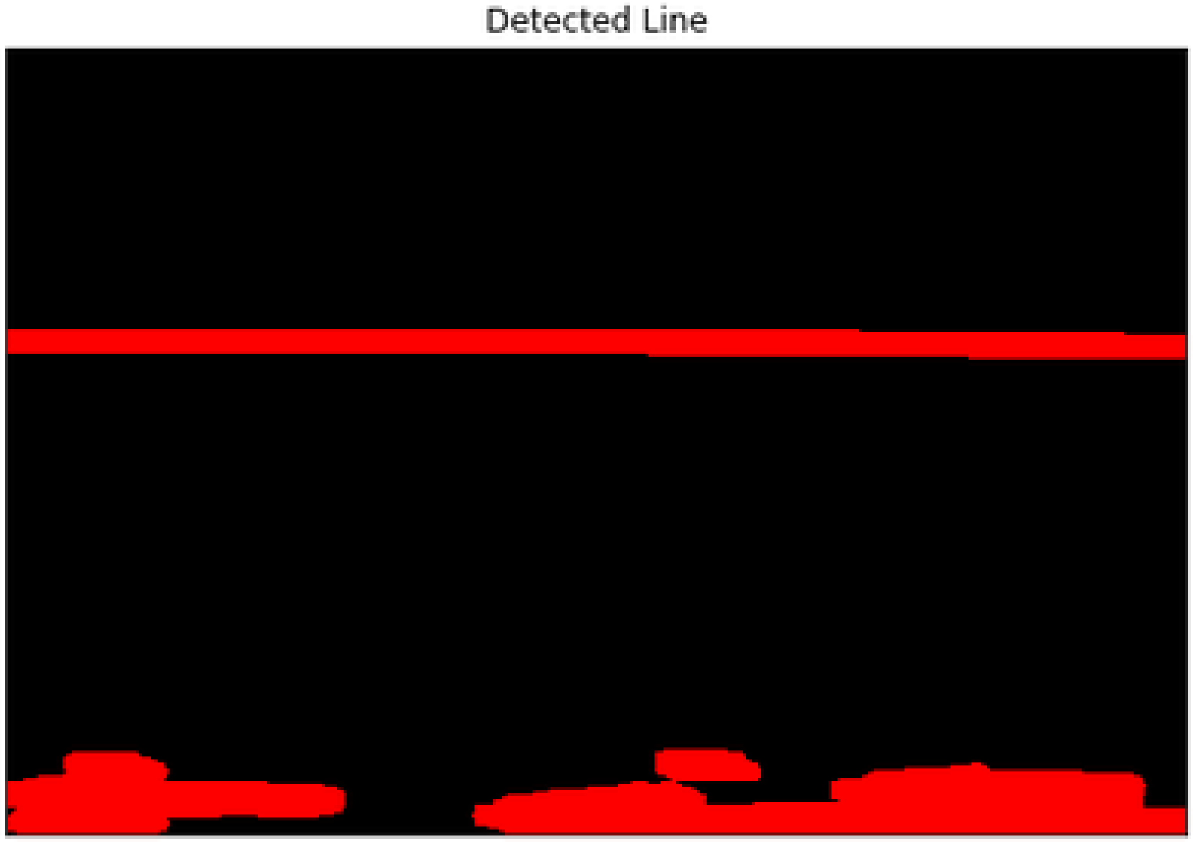
Implementation

Sample 1



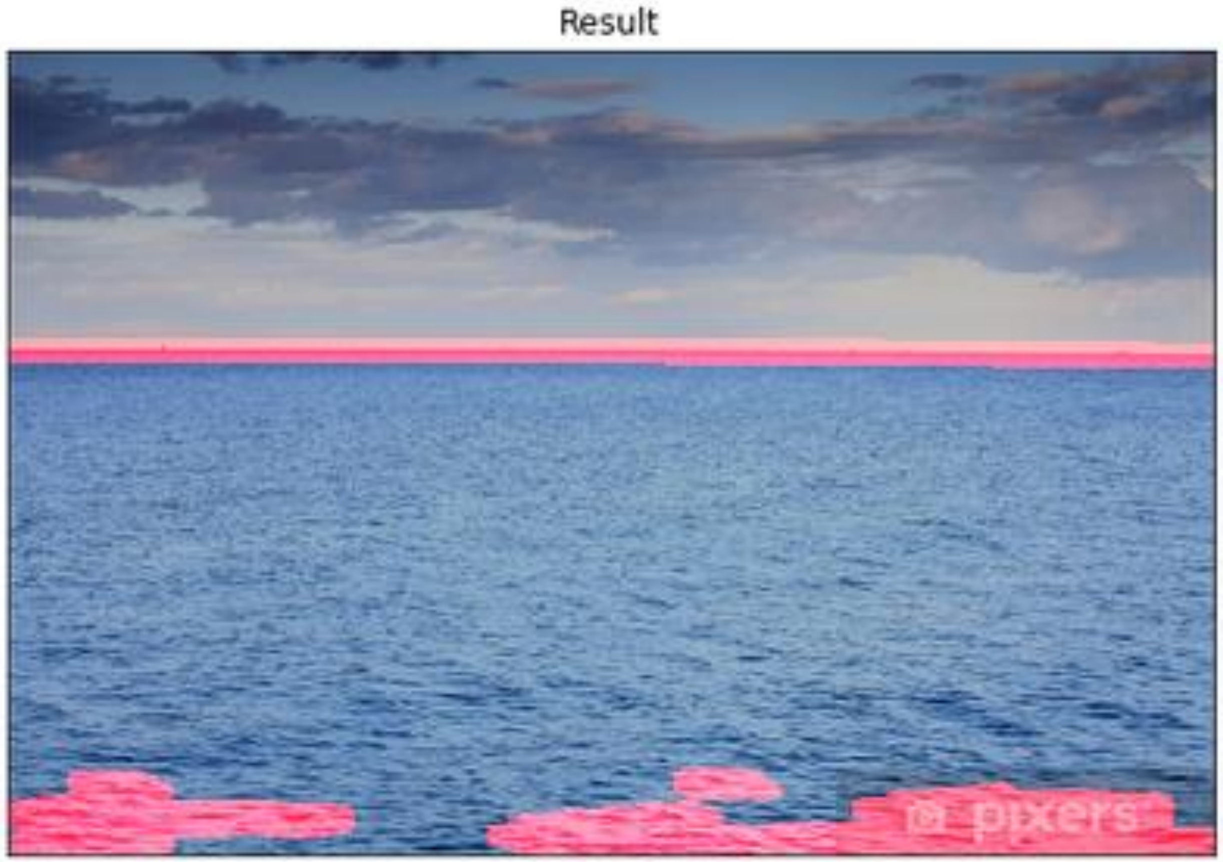
Implementation

Sample 1



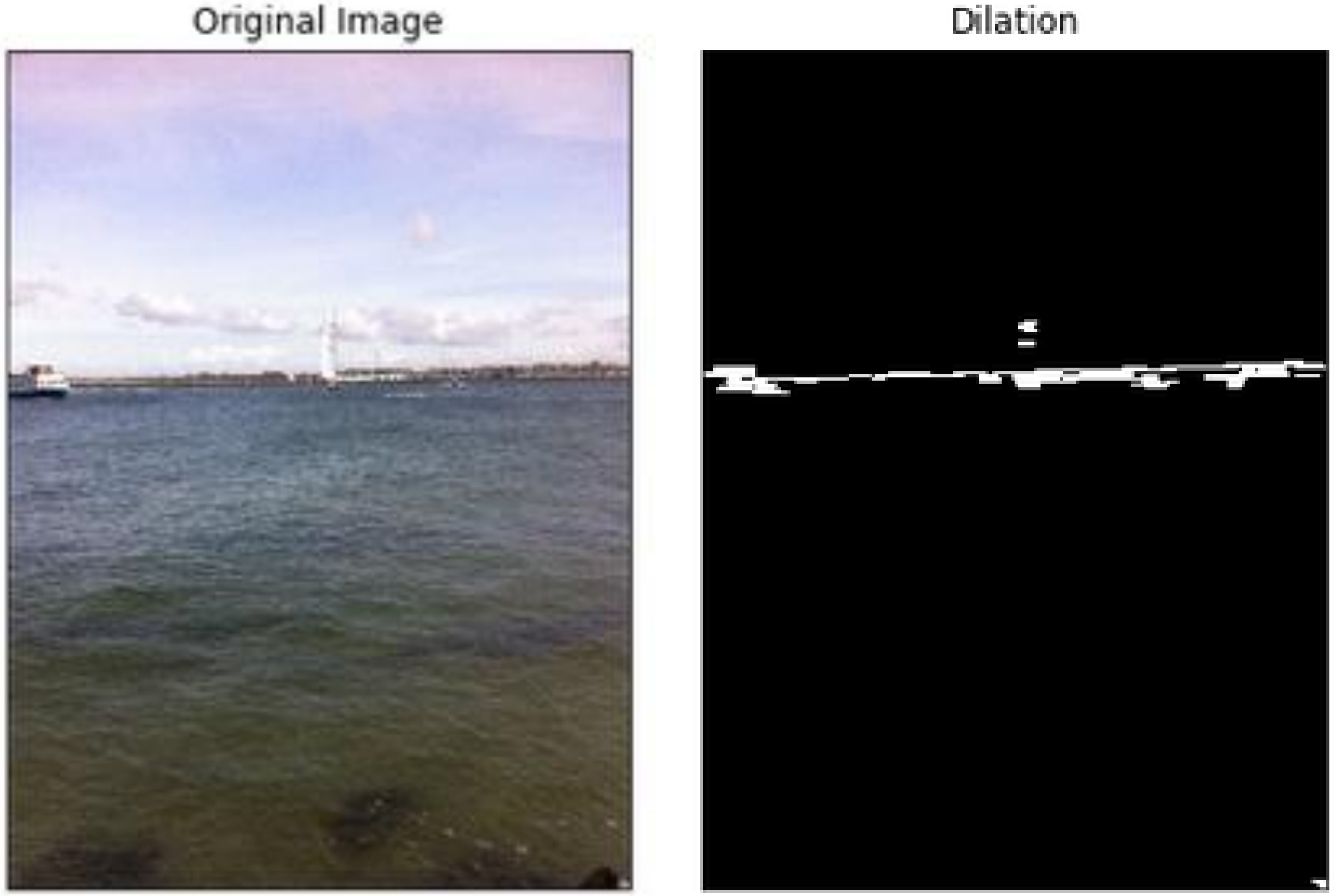
Implementation

Sample 1



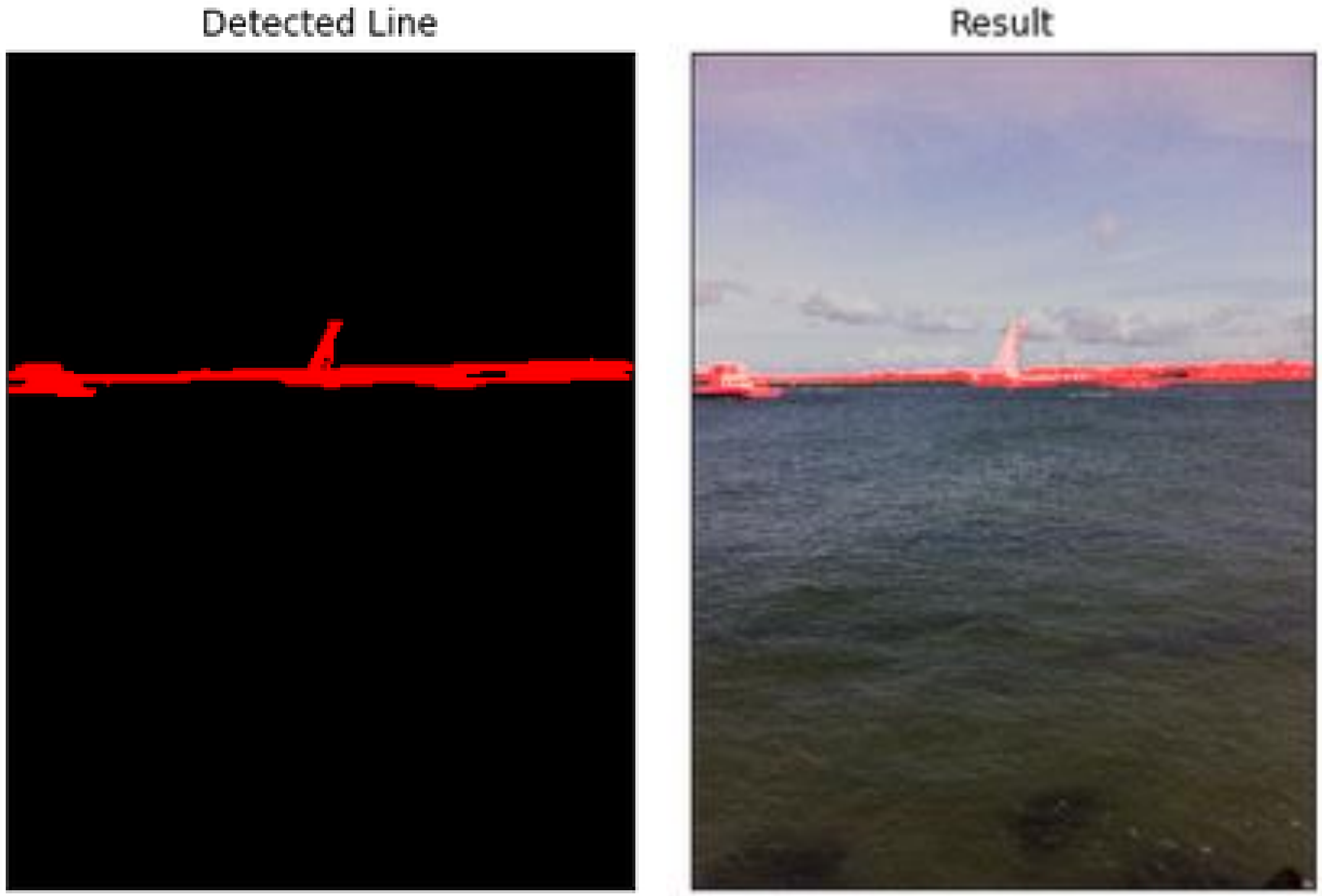
Implementation

Sample 2



Implementation

Sample 2



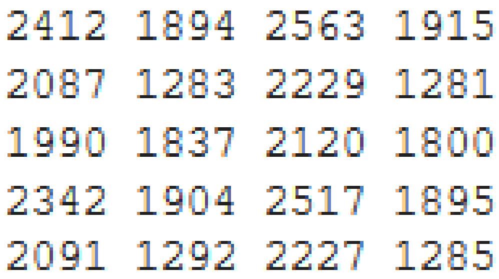
Detected Line

Result

Implementation

Sample 3

Coordinates of the lines



x_1	y_1	x_2	y_2
2412	1894	2563	1915
2087	1283	2229	1281
1990	1837	2120	1800
2342	1904	2517	1895
2091	1292	2227	1285

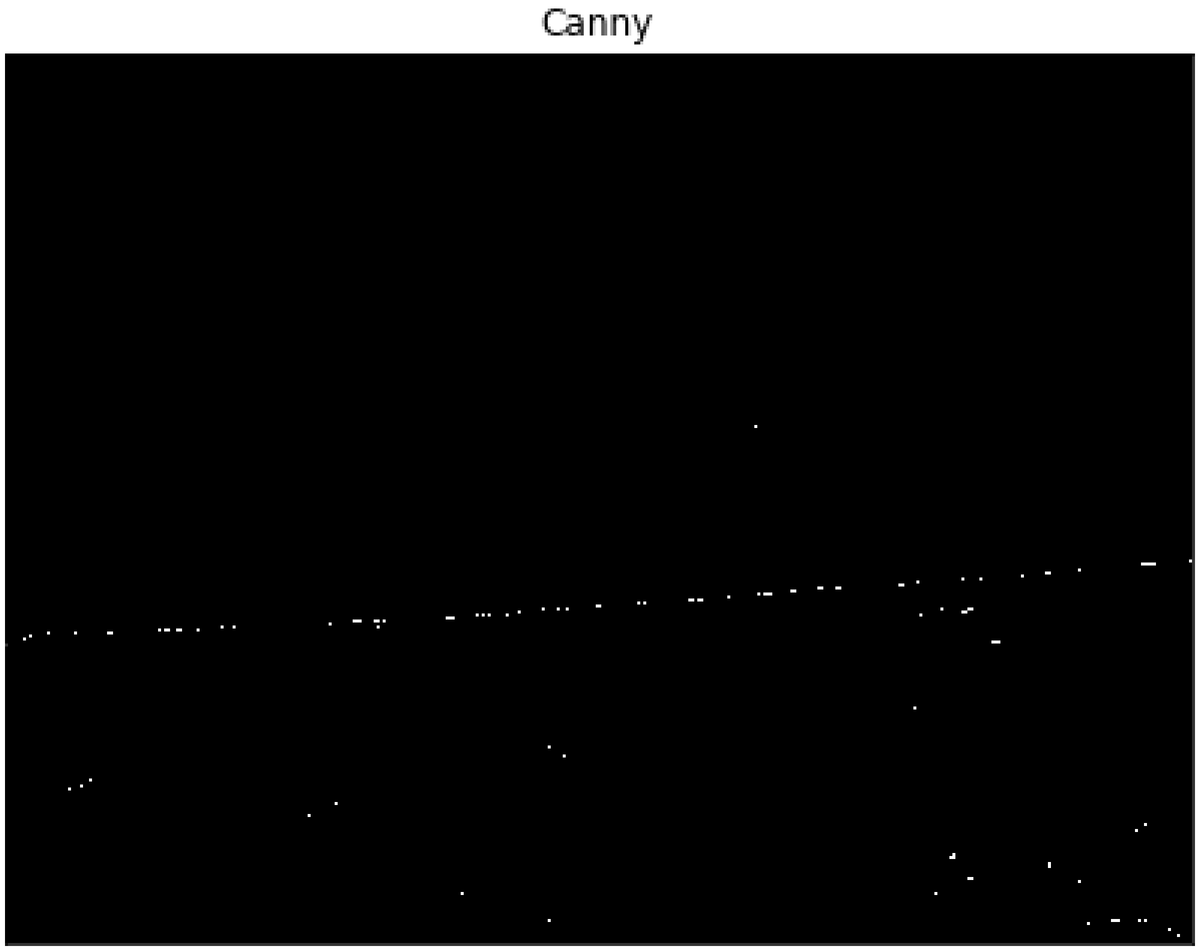
Implementation

Sample 3

Original Image

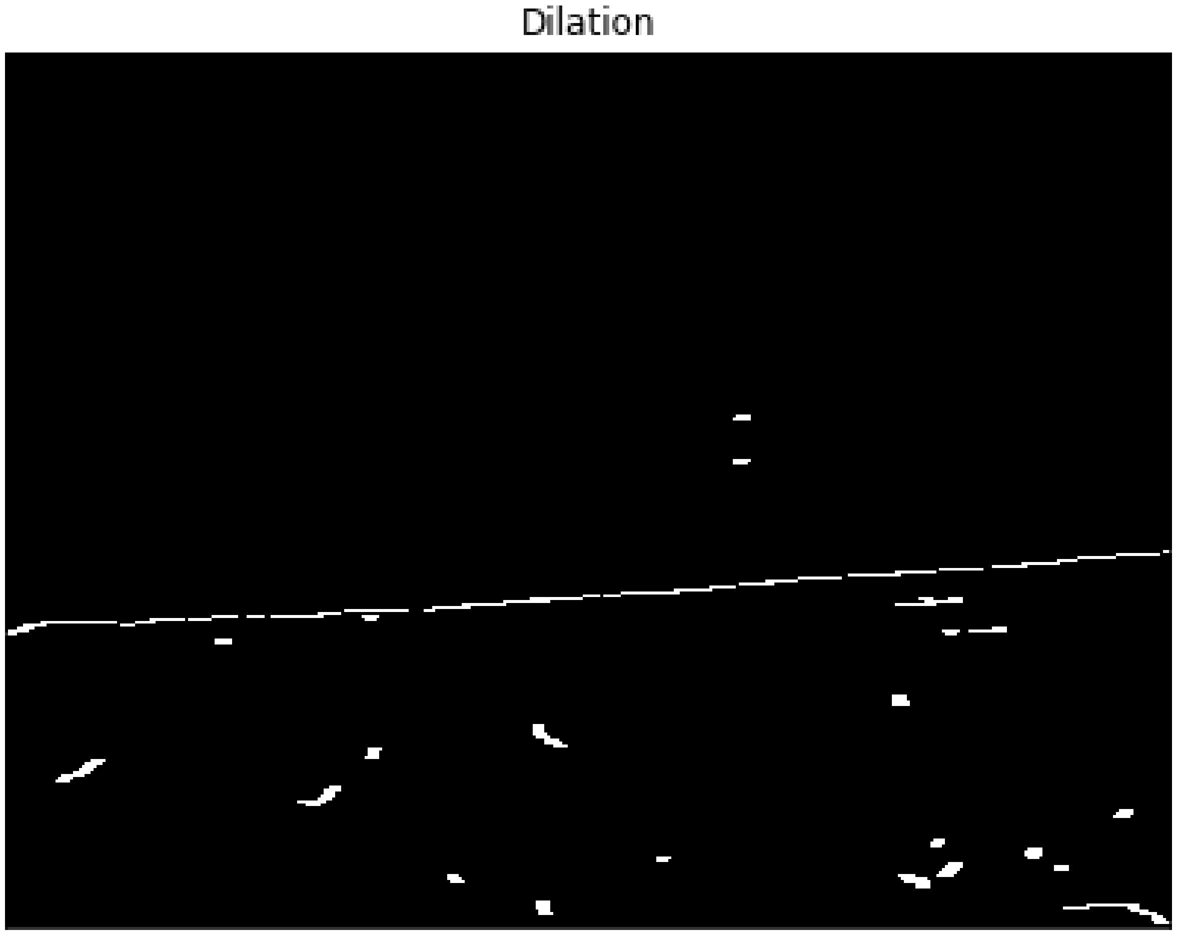
Implementation

Sample 3



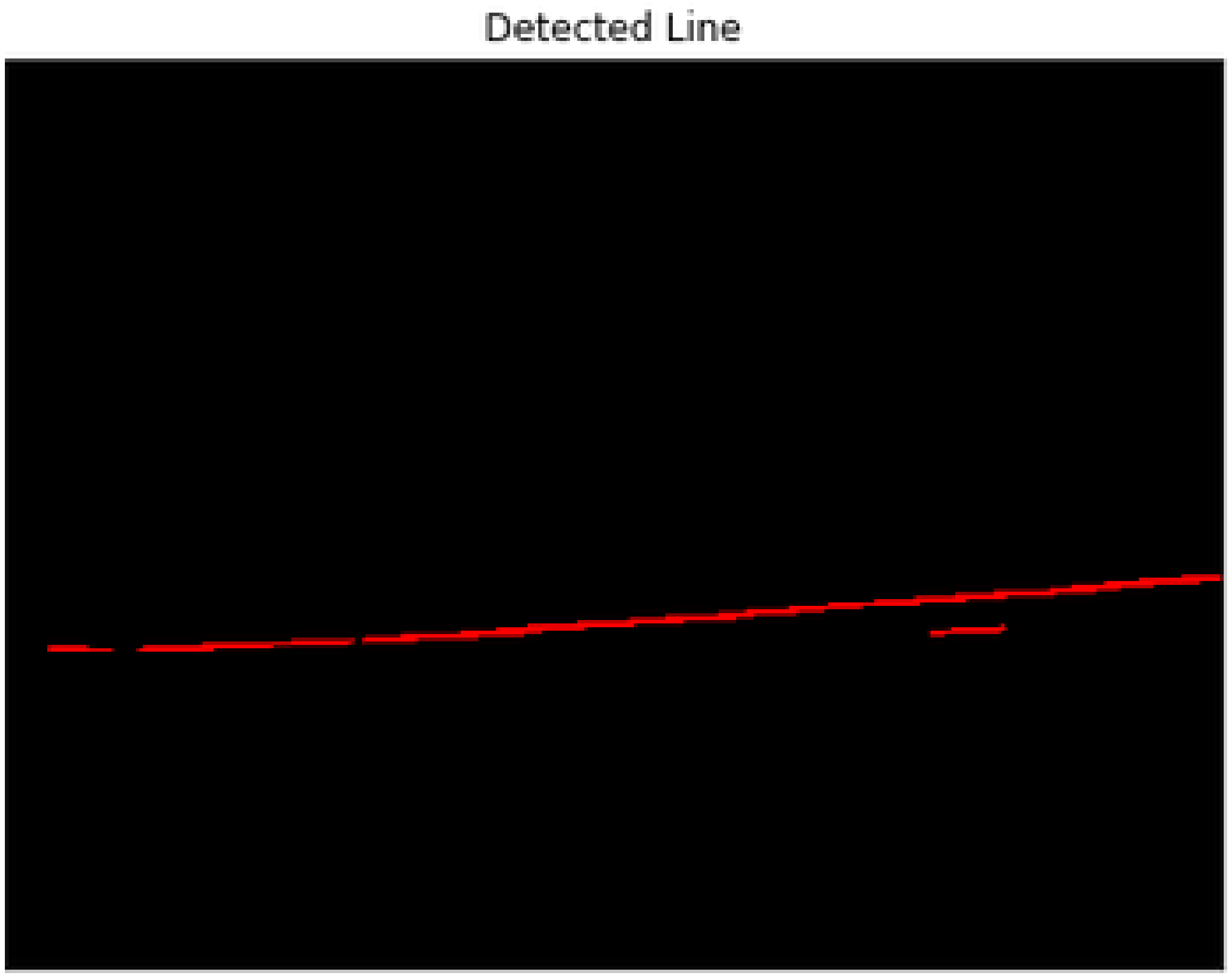
Implementation

Sample 3



Implementation

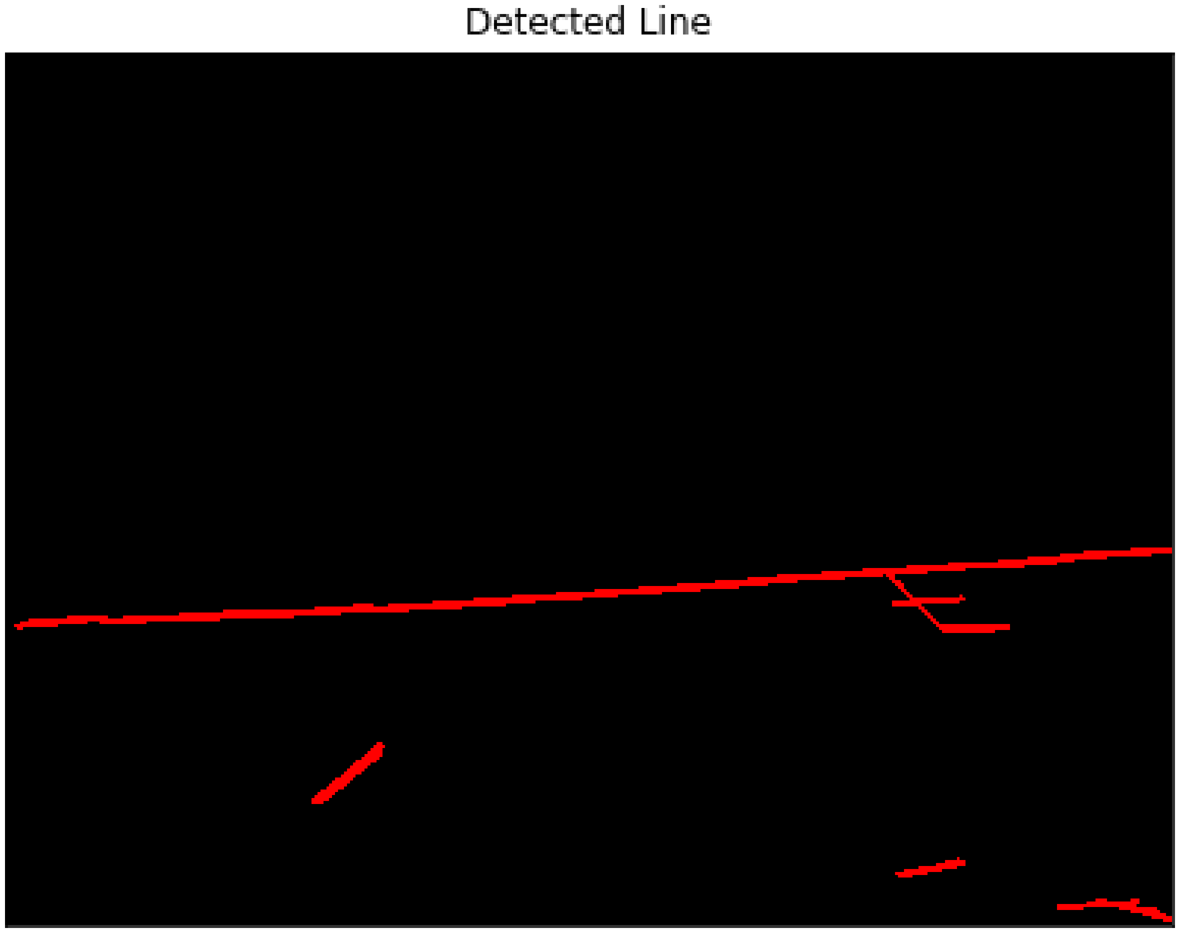
Sample 3



maximum line gap = 20

Implementation

Sample 3

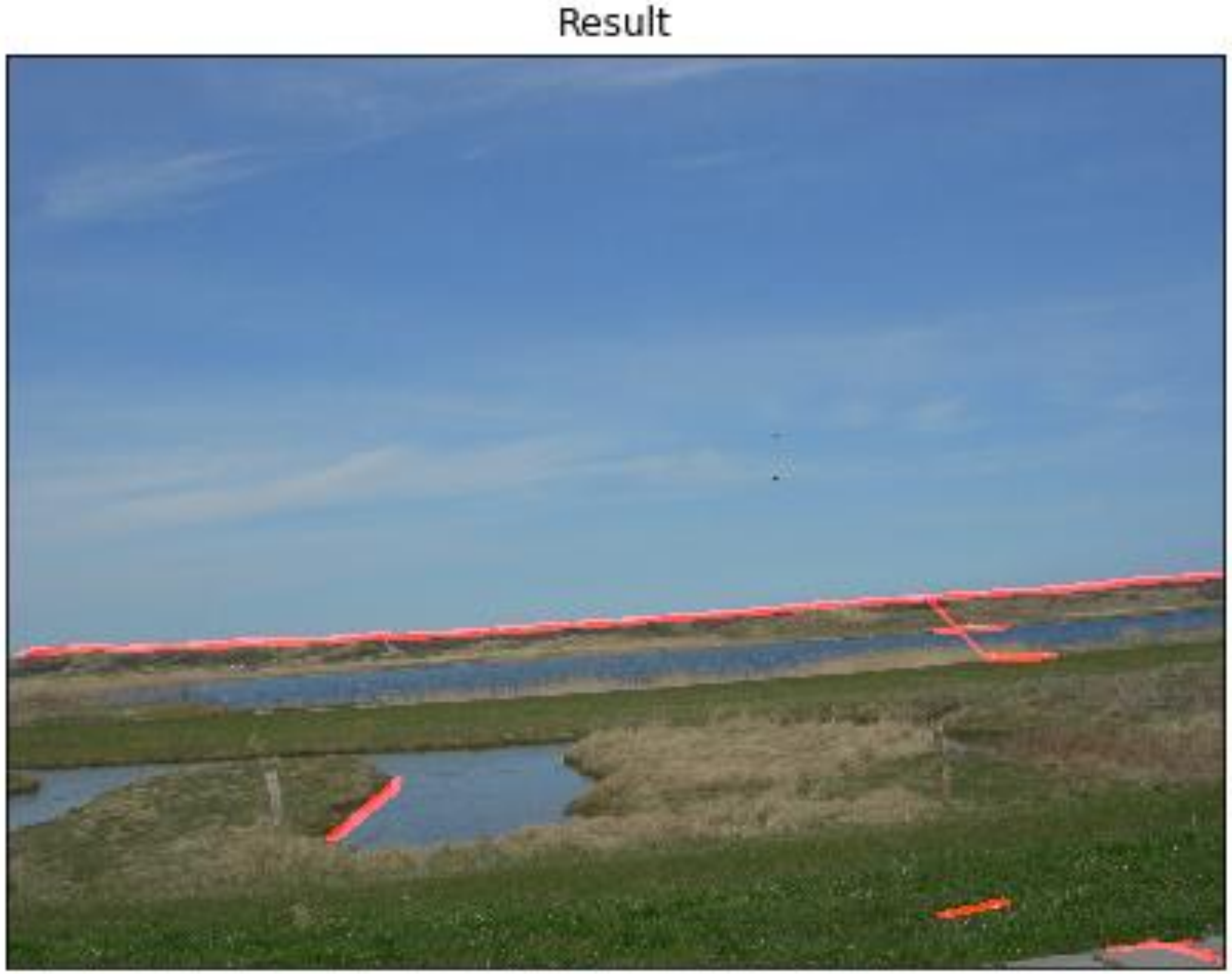


Detected Line

maximum line gap = 80

Implementation

Sample 3



Implementation

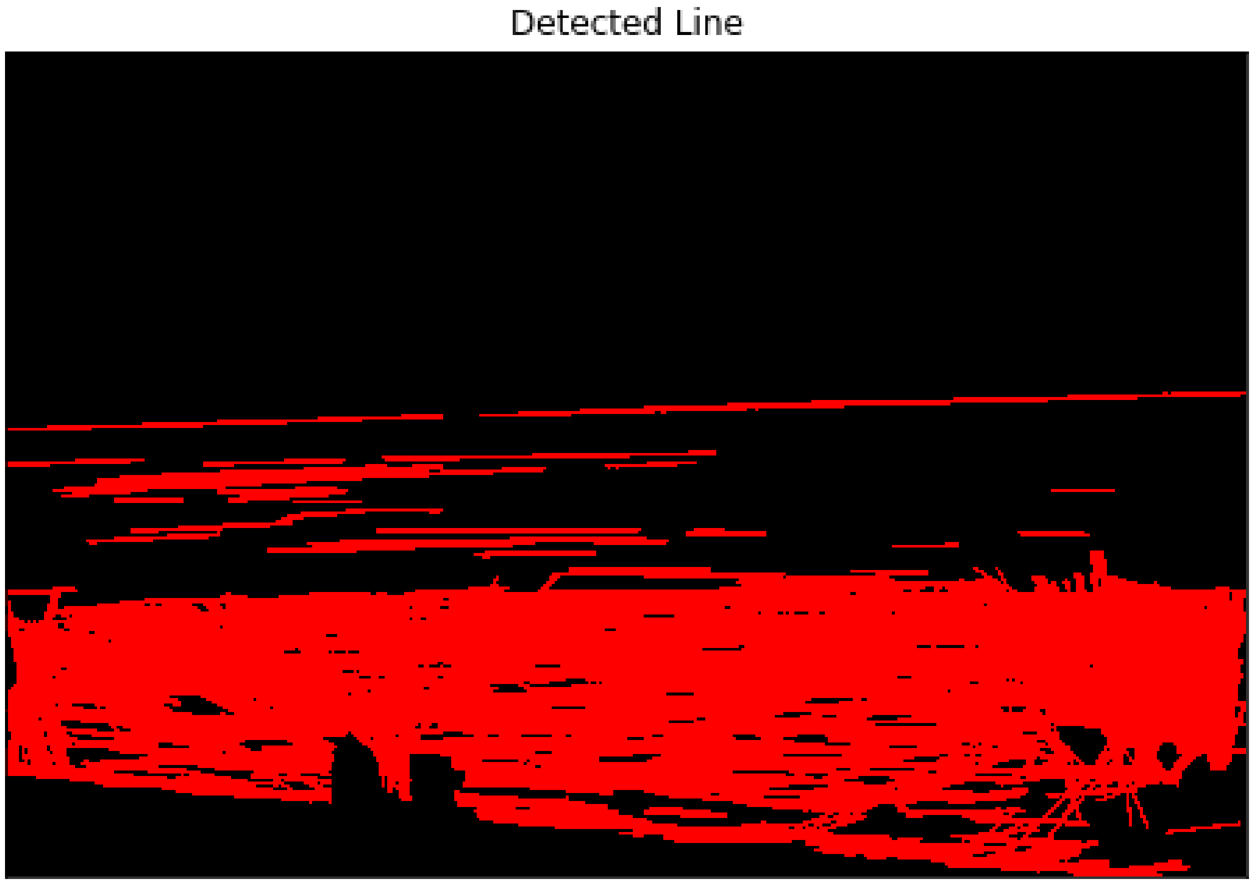
Sample 3

Implementation

Sample 4

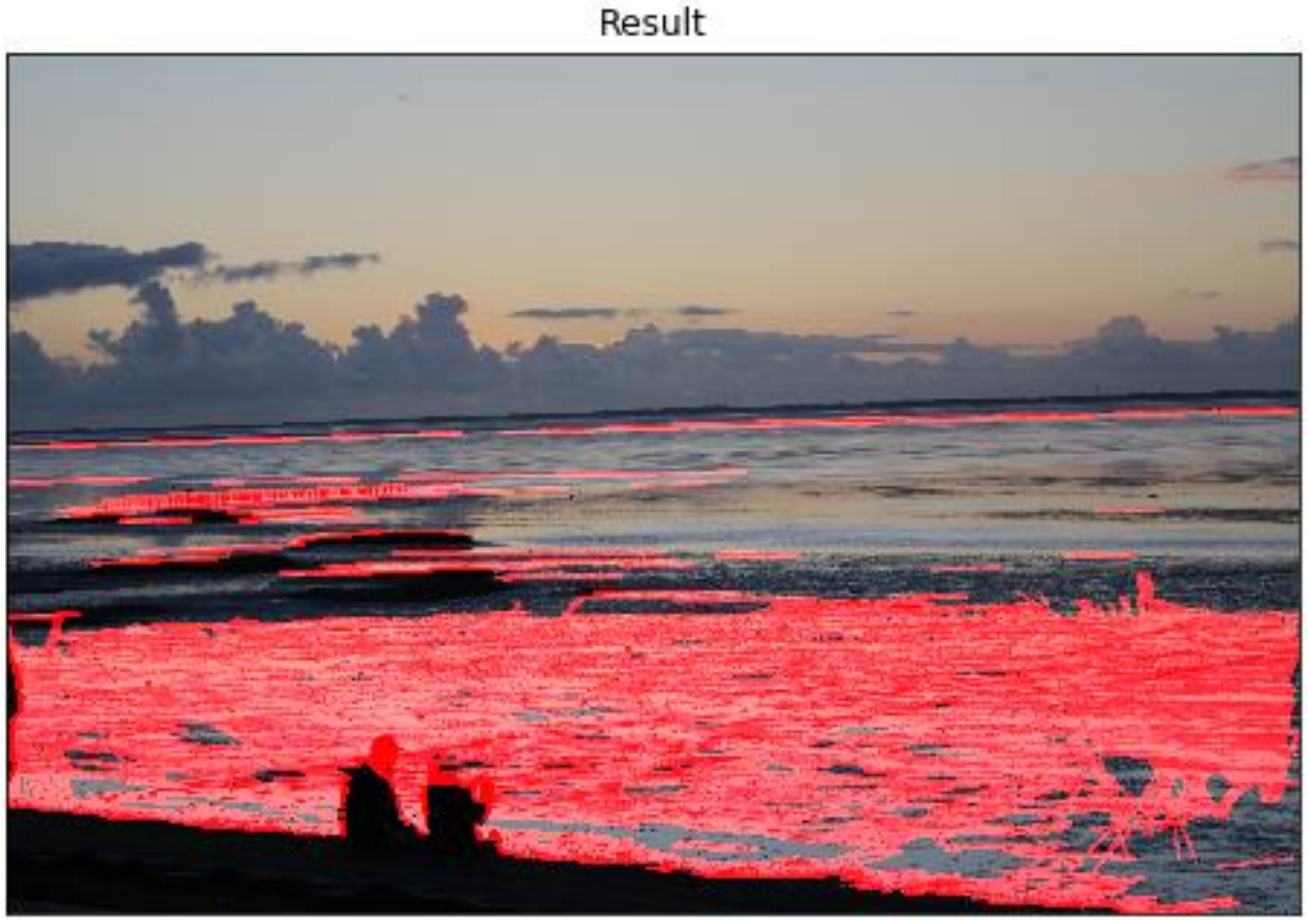
Implementation

Sample 4



Implementation

Sample 4



Challenges

- find the right parameters for the line detection
- the function isn‘t robust yet
  - cannot find smooth horizon
  - cannot find horizon with less colour contrast
- filter interesting lines in the image for further operation

Next Steps

- make the parameters more suitable and the function more robust
- test the function with even more pictures
- Concatenate a series of lines to longer lines if they fit the same linear equation[2]
- getting the coordinates of the horizon to adjust it on the line of interest
- calculate the angle between frame and horizon line
- rotate the image by the calculated angle

Sources

[1] https://pixers.net.au/canvas-prints/cloudy-blue-sky-leaving-for-horizon-blue-surface-sea-45502886 (28.11.2019)

[2] https://stackoverflow.com/questions/53750209/detecting-lines-vertical-and-horizontal-that-are-not-straight-and-align-image (02.12.2019)

Make It Aesthetic

Anna-Maria Hohmann
Konrad Kring
Meret Lindanis

4th presentation
03.12.2019