Make It Aesthetic

Anna-Maria Hohmann
Konrad Kring
Meret Lindanis

3rd presentation
19.11.2019

About Us

Meret

- computer science
- medical technologies

Anna

- business informatics
- project management

Konrad

- pedagogics
- music

Goals of Our Project

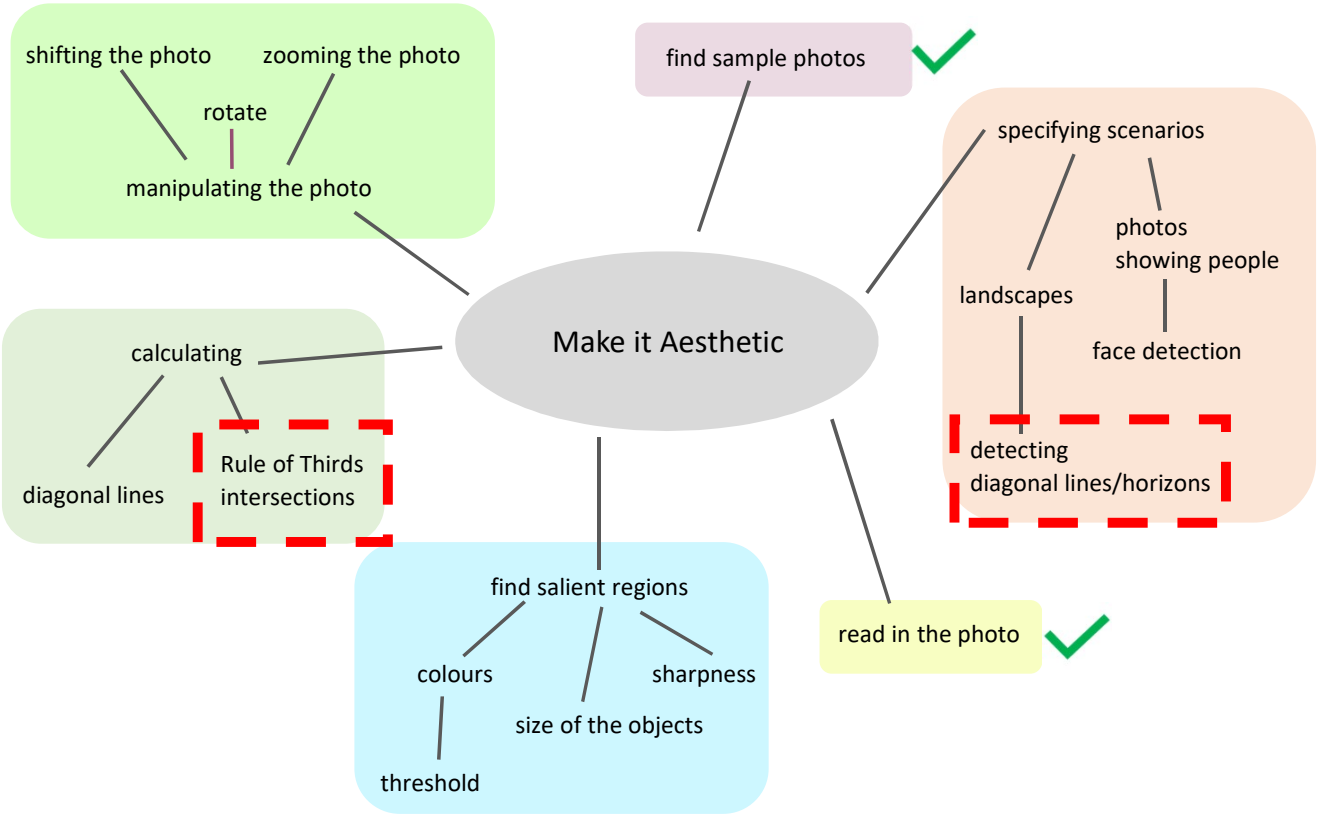
our motivation:

- 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
shifting the photo
zooming the photo
rotate
manipulating the photo
specifying scenarios
landscapes
photos showing people
face detection
detecting diagonal lines/horizons
calculating
diagonal lines
Rule of Thirds intersections
find salient regions
colours
sharpness
size of the objects
threshold
read in the photo

Use-Cases

- make photos more aesthetic for a photo album / website
  - wedding /big events --> people/creature
  - vacation --> landscape, buildings, people
  - art --> all categories
- let the algorithms editing all your photos instead of doing it manually

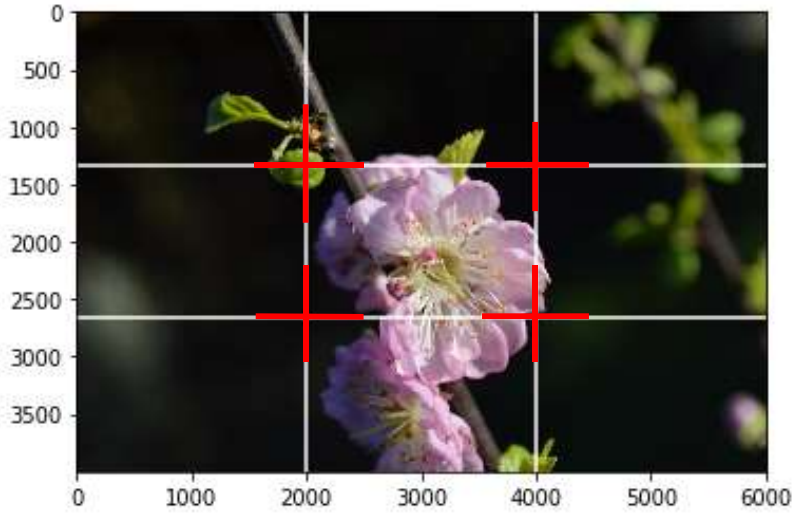
Scenarios

- using clear colour contrast first
  - bright background, dark main object
  - dark background, bright main object
  - only one object
- adding more and more complicated scenarios
  - more objects
  - fewer colour contrasts
  - i.e. bright background, bright main object
  - and vice versa

Focus

Calculation of the points of interest

- to move objects to these points
- to make the picture more vivid



Implementation

Function getting the points of interest

```
def poi(img):
  third_of_height_1, third_of_height_2, third_of_width_1, third_of_width_2 =generate_image_data(img)

  poi1 = tuple([third_of_height_1, third_of_width_1])
  poi2 = tuple([third_of_height_1, third_of_width_2])
  poi3 = tuple([third_of_height_2, third_of_width_1])
  poi4 = tuple([third_of_height_2, third_of_width_2])

  print(poi1)
  print(poi2)
  print(poi3)
  print(poi4)
```

```
poi(img_snail)
```

Implementation

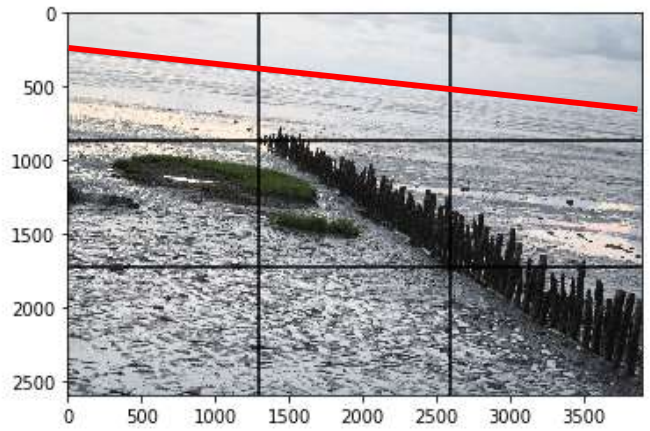
Result

```
(1333, 2005)
(1333, 4010)
(2666, 2005)
(2666, 4010)
```

Focus

Detection of horizon

- to align the image on the lines of interest
- to make the horizon straight



Implementation

First Attempt

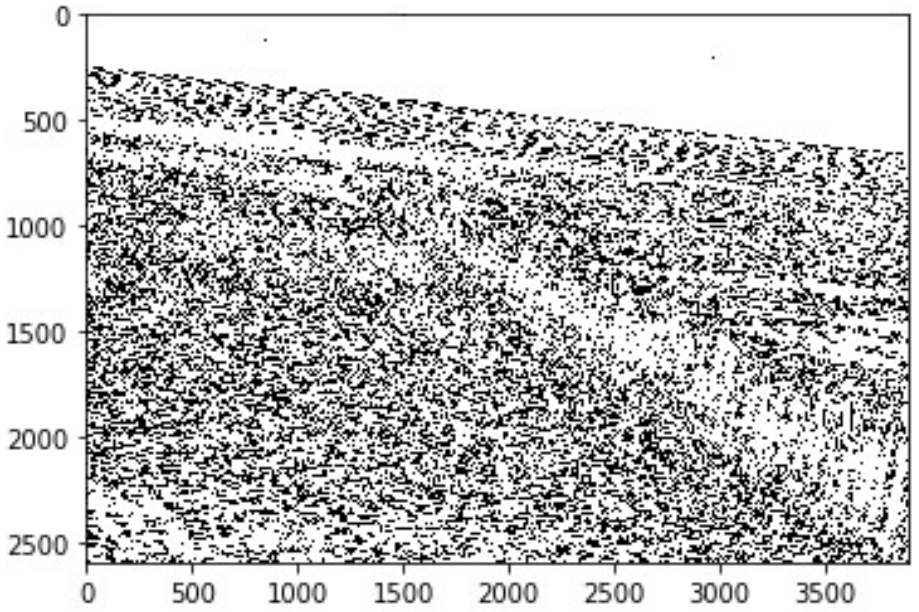
Median blur and thresholding for recognizing the horizon more easily

```
img = cv2.imread('beach_diagonal.jpg',0)
img = cv2.medianBlur(img,5)

ret,th1 = cv2.threshold(img,127,255,cv2.THRESH_BINARY)
th2 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_MEAN_C,\
            cv2.THRESH_BINARY,11,2)
th3 = cv2.adaptiveThreshold(img,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C,\
            cv2.THRESH_BINARY,11,2)

titles = ['Original Image', 'Global Thresholding (v = 127)',
            'Adaptive Mean Thresholding', 'Adaptive Gaussian Thresholding']
images = [img, th1, th2, th3]

#for i in range(4):
    #plt.subplot(2,2,i+1),plt.imshow(images[i],'gray')
    #plt.title(titles[i])
    #plt.xticks([]),plt.yticks([])
plt.imshow(th3,'gray')
```

Implementation

First Attempt

Result



Implementation

Second Attempt

Preprocessing

- Canny Edge Detection
- Calculating the diagonal of the image

```
img = cv2.imread('beach_diagonal.jpg')
gray = cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
edges = cv2.Canny(gray,50,150,apertureSize = 3)

diagonale = np.sqrt(img.shape[0]**2 + img.shape[1]**2)
print(diagonale)
```

Implementation

Second Attempt

Detection via Hough Line Transformation

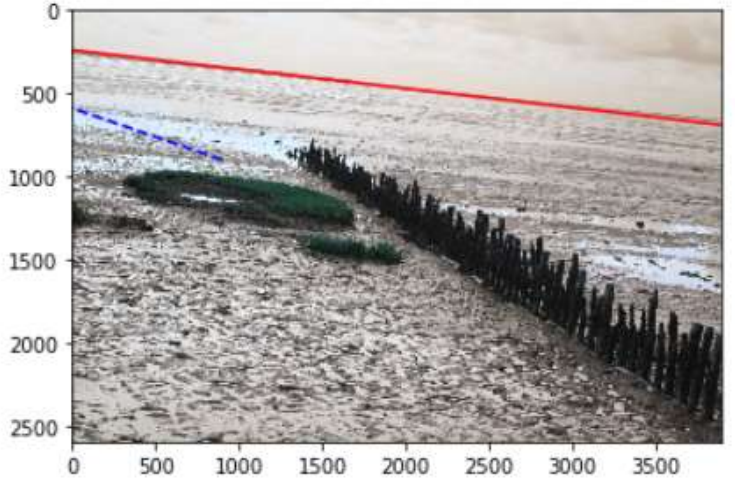
```
lines = cv2.HoughLines(edges,diagonale,np.pi/180,200)
for rho,theta in lines[0]:
    print('rho', rho)
    print('theta', theta)
    a = np.cos(theta)
    print('a', a)
    b = np.sin(theta)
    print('b', b)
    x0 = a*rho
    print('x0', x0)
    y0 = b*rho
    print('y0', y0)
    x1 = int(x0 +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

    #img_lines = cv2.line(img,(0,240),(4000,700),(255,0,0),10)
    plt.plot([0,4000], [240,700], color = 'r', linestyle = '-')
    plt.plot([x1,y1], [x2,y2], color='b', linestyle='--')

#cv2.imwrite('houghlines3.jpg',img_lines)
plt.imshow(img)
```

Implementation

Second Attempt

Result

```
4672.79445300133
rho 0.0
theta 2.024582
a -0.43837112
b 0.89879405
x0 -0.0
y0 0.0
x1 898
y1 -438
x2 898
y2 438
<matplotlib.image.AxesImage at 0x7f8eebf5cdd8>
```



Implementation

Circle Detection

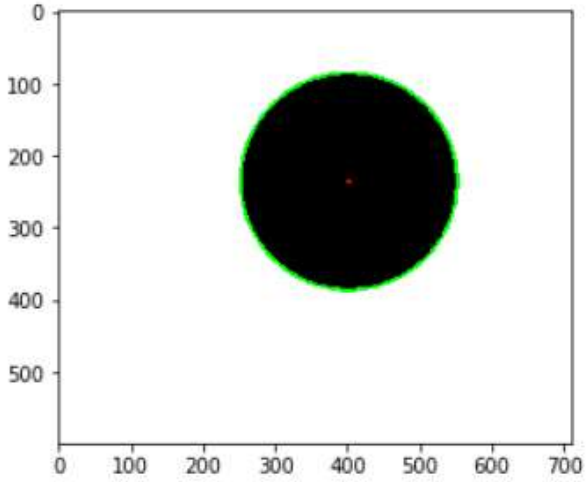
- Median Blur
- Hough Circle Transformation

```
! wget -q https://raw.githubusercontent.com/.../Pictures/circle.png
img = cv2.imread('circle.png')
output = img.copy()
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
#cimg = cv2.cvtColor(img, cv2.COLOR_GRAY2BGR)
median = cv2.medianBlur(gray, 5)
plt.imshow(median, 'gray')

circles = cv2.HoughCircles(median, cv2.HOUGH_GRADIENT, 1, 30, param1=50,
                           param2=30, minRadius=0, maxRadius=0)
detected_circles = np.uint16(np.around(circles))
print(circles)
for (x, y, r) in detected_circles[0,:]:
  img_circle = cv2.circle(output, (x, y), r, (0, 255, 0), 3)
  cv2.circle(output, (x, y), 2, (255, 0, 0), 3)

plt.imshow(img_circle)
```

Implementation

Result

```
[[[401.5 236.5 150.1]]]
<matplotlib.image.AxesImage at 0x7f8eebd80128>
```



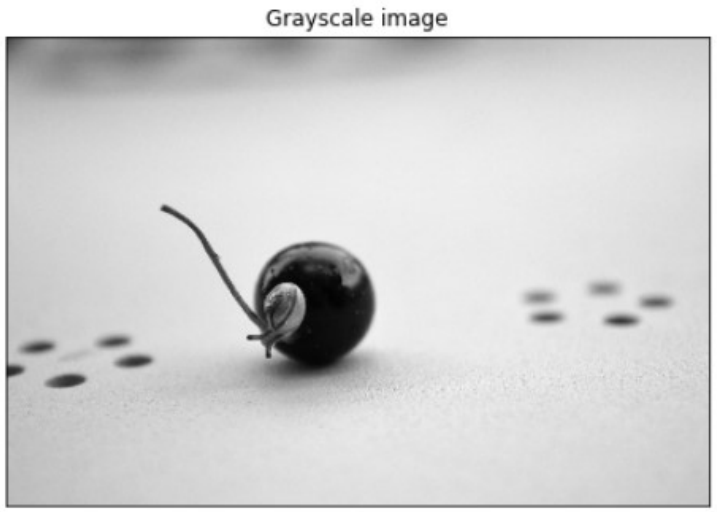
Implementation

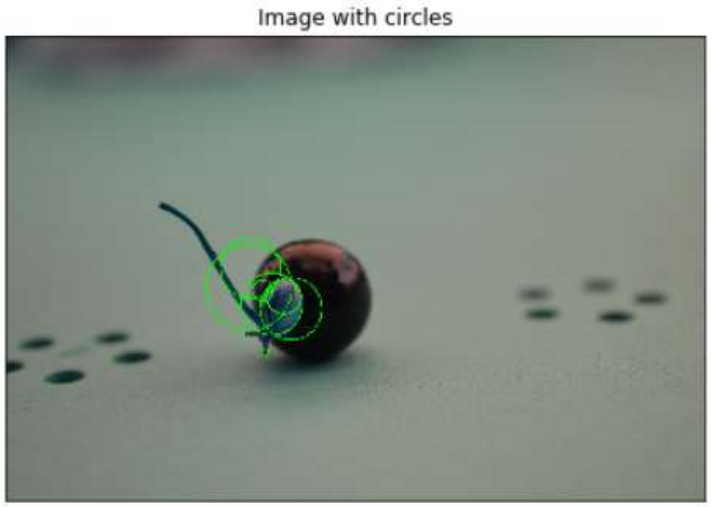
Circle Detection

```
! wget -q https://raw.githubusercontent.com/.../Pictures/grafik.png
img = cv2.imread('snail.JPG')
output = img.copy()
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
median = cv2.medianBlur(gray, 5)

circles = cv2.HoughCircles(median, cv2.HOUGH_GRADIENT, 0.9, 120, param1=50,
                           param2=30, minRadius=0, maxRadius=0)
detected_circles = np.uint16(np.around(circles))
print(detected_circles)
for i in circles[0,:]:
  #draw the outer circle
  img_circle = cv2.circle(img, (i[0], i[1]), i[2], (0, 255, 0), 10)
  #draw center of circle
  img_center = cv2.circle( img, (i[0], i[1]), 2, (0, 0, 255), 5)

titles = ['Grayscale image', 'Image with circles']
images = [gray, img_circle]

plt.imshow(img_circle)
for i in range(2):
    plt.subplot(1,2,i+1),plt.imshow(images[i], 'gray')
    plt.subplots_adjust(left=1.0, bottom=1.0, right=3.0, top=2.0)
    plt.title(titles[i])
    plt.xticks([]),plt.yticks([])
plt.show()
```

Implementation

Result






```
[[[2396 2252  149]
  [2458 2356  268]
  [2292 2316  266]
  [2096 2178  372]
  [2110 2012  258]
  [2214 2198   87]
  [2238 2726   24]
  [2076 2572   23]]]
```

Challenges

- find the right values for the parameters for the Hough Transformations
  - object detection
  - detection of horizons

Next Steps

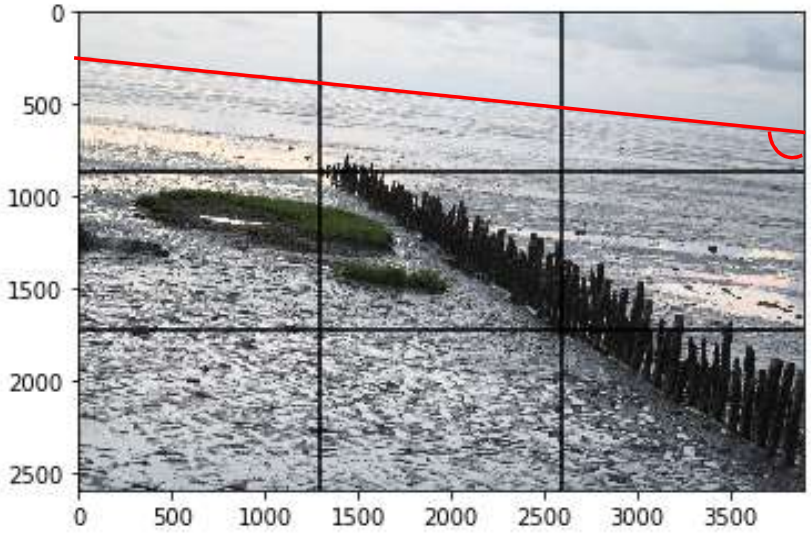
- detect object in a simple image
- detect horizon to align it on the line of interest

Next Steps

- optimize **Hough Line Transformation** to detect the lines in the image
- calculate the angle between the frame and the detected line of the horizon
- rotate the image by the angle to make the horizon parallel
- calculate the distance to the RT-lines
- find the line closest to the horizon line
- cut the height by the calculated distance

Next Steps

- calculate the angle between the frame and the detected line of the horizon

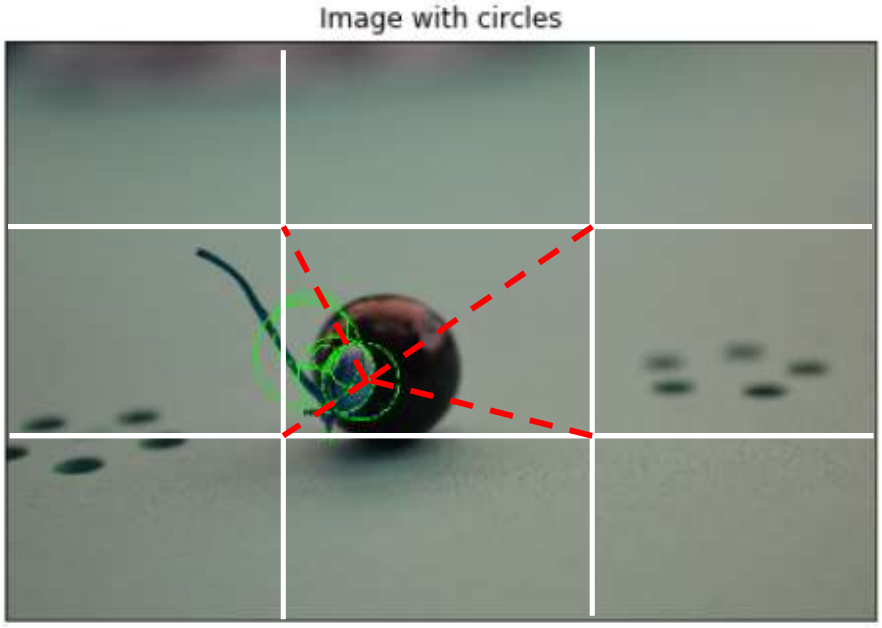


Next Steps

- optimize **Hough Circle Transformation** to detect simple circular forms in the image
- calculate the center of the dected objects
- calculate the difference between the x-coordinate of the object center and the POI‘s
- find the nearest POI‘s
- similarly for the y-coordinate
- cut the image in the x- and y-direction by the calculated difference

Next Steps

- calculate the difference between the x-coordinate of the object center and the pois



Make It Aesthetic

Anna-Maria Hohmann
Konrad Kring
Meret Lindanis

3rd presentation
19.11.2019