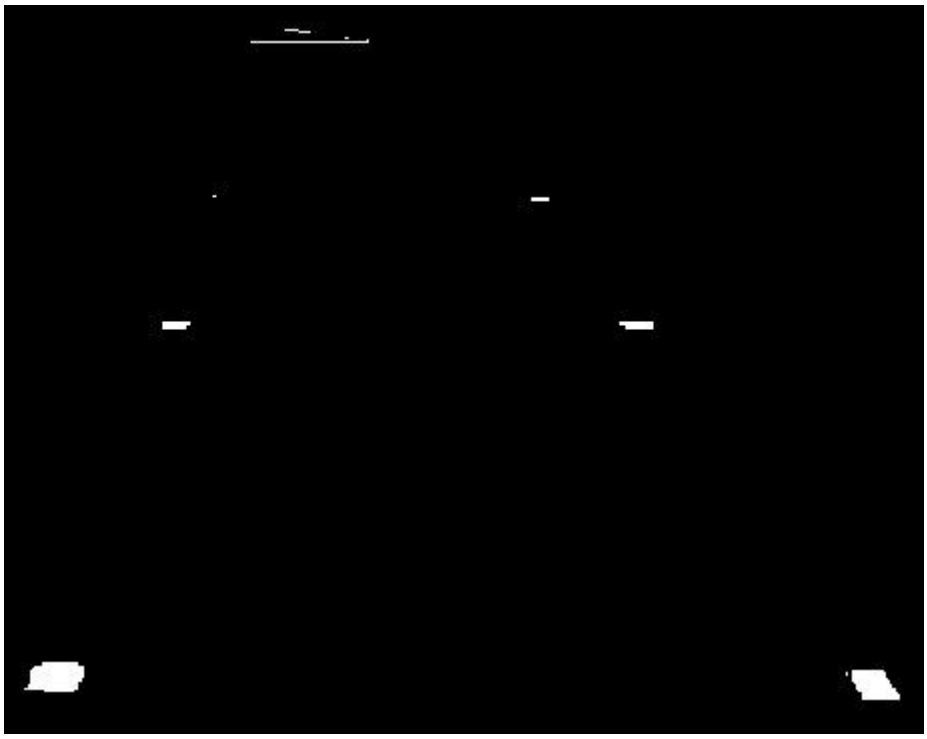
Assignment 4 – Robotics
By Pascal Müller, Friedrich Müller, Fabian Casares

Exercise 1-3



Exercise 4+5

```
# In the end we used the picture which can be found on the Assignment 4 sheet for tasks 4 and 5.
import numpy as np
import matplotlib.pyplot as plt
import cv2
import math

img = cv2.imread('bw.jpg',cv2.IMREAD_COLOR)
RGB_img = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
gray= cv2.cvtColor(RGB_img, cv2.COLOR_RGB2GRAY)

#bi_gray
bi_gray_max = 255
bi_gray_min = 20
ret,thresh=cv2.threshold(gray, bi_gray_min, bi_gray_max, cv2.THRESH_BINARY);

#Correcting some weird white-side effect
thresh[:,-1]=0

plt.imshow(thresh, cmap='gray')
```

```
plt.show()

img_points = np.zeros((6,2,1))

# find contours in the binary image
im2, contours, hierarchy = cv2.findContours(thresh,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)
for index, c in enumerate(contours):
   # calculate moments for each contour
   M = cv2.moments(c)

   # calculate x,y coordinate of center
   cX = M["m10"] / M["m00"]
   cY = M["m01"] / M["m00"]
   img_points[index]=np.array([[cX],[cY]])

fx = 614.1699
fy = 614.9002
cx = 329.9491
cy = 237.2788

camera_mat = np.zeros((3,3,1))
camera_mat[:,:,0] = np.array([[fx, 0, cx],[0, fy, cy], [0, 0, 1]])

k1 = 0.1115
k2 = -0.1089
p1 = 0
p2 = 0

dist_coeffs = np.zeros((4,1))
dist_coeffs[:,0] = np.array([[k1, k2, p1, p2]])

# far to close, left to right (order of discovery) in cm
obj_points = np.zeros((6,3,1))
obj_points[:,:,0] = np.array([[0.0, 0.0, 0.0],[21.8, 0.0, 0.0], [0.0, 30.0, 0.0], [22.2, 30.0, 0.0], [0.0, 60.0,
0.0], [22.0, 60.0, 0.0]])

retval, rvec, tvec = cv2.solvePnP(obj_points, img_points,camera_mat, dist_coeffs)

print("Rotation Vector: \n{}\n".format(rvec))
print("Translation Vector: \n{}\n".format(tvec))

#Start Aufgabe 6
rmat = np.zeros((3,3))
cv2.Rodrigues(rvec, rmat, jacobian=0)

sy = math.sqrt(rmat[0][0] * rmat[0][0] +  rmat[1][0] * rmat[1][0])
```

```
singular = sy < 1e-6

if  not singular :
    x = math.atan2(rmat[2][1] , rmat[2][2])
    y = math.atan2(-rmat[2][0], sy)
    z = math.atan2(rmat[1][0], rmat[0][0])
else :
    x = math.atan2(-rmat[1][2], rmat[1][1])
    y = math.atan2(-rmat[2][0], sy)
    z = 0

print("angles: {}".format([x, y, z]))
```