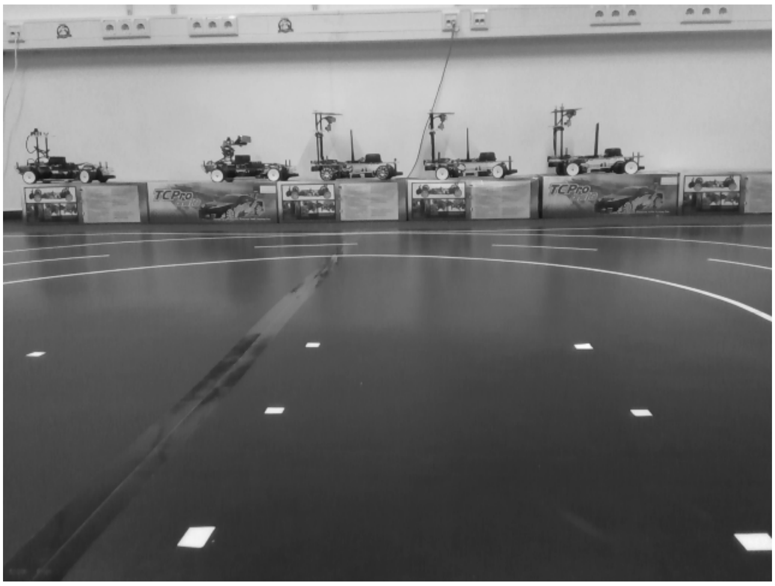
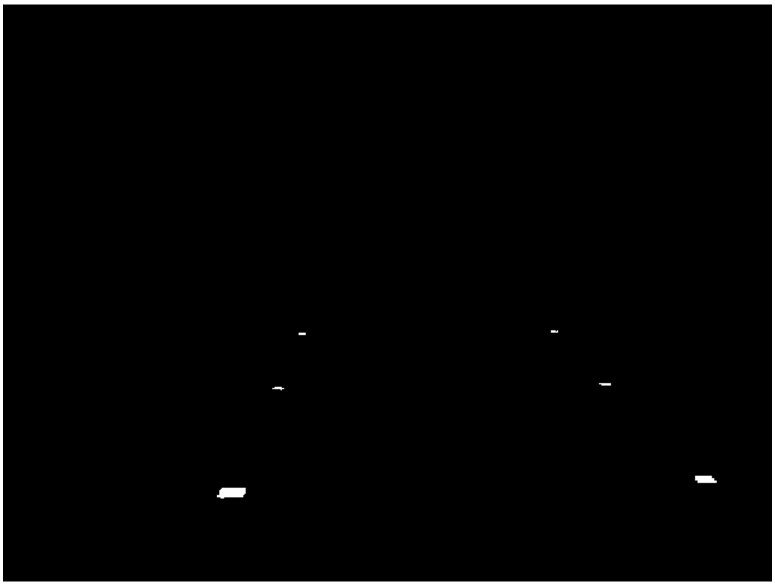
# 1 Aufabe 2



# 2 Aufabe 3



# 3 Aufabe 4, 5, 6

## 3.1 Terminal

```
367 Coordinates:
 [425, 271], [426, 271], [427, 271], [422, 272], [423, 272], [424, 272], [425, 272],
  → [426, 272], [427, 272], [206, 277], [207, 277], [208, 277], [209, 277], [210,
  → 277], [211, 277], [212, 277], [213, 277], [214, 277], [215, 277], [460, 315],
  → [461, 315], [462, 315], [463, 315], [464, 315], [465, 315], [466, 315], [467,
  → 315], [468, 315], [469, 315], [460, 316], [461, 316], [462, 316], [463, 316],
  → [464, 316], [465, 316], [466, 316], [467, 316], [468, 316], [469, 316], [184,
  → 324], [185, 324], [190, 324], [191, 324], [192, 324], [182, 325], [183, 325],
  → [184, 325], [185, 325], [186, 325], [187, 325], [188, 325], [189, 325], [190,
    → 325], [191, 325], [192, 325], [193, 325], [182, 326], [183, 326], [184, 326],
    → [185, 326], [186, 326], [187, 326], [188, 326], [189, 326], [190, 326], [191,
    → 326], [192, 326], [536, 394], [537, 394], [538, 394], [539, 394], [540, 394],
    → [541, 394], [542, 394], [543, 394], [544, 394], [545, 394], [546, 394], [547,
    → 394], [548, 394], [549, 394], [550, 394], [536, 395], [537, 395], [538, 395],
    → [539, 395], [540, 395], [541, 395], [542, 395], [543, 395], [544, 395], [545,
    → 395], [546, 395], [547, 395], [548, 395], [549, 395], [550, 395], [551, 395],
    → [536, 396], [537, 396], [538, 396], [539, 396], [540, 396], [541, 396], [542,
    → 396], [543, 396], [544, 396], [545, 396], [546, 396], [547, 396], [548, 396],
    → [549, 396], [550, 396], [551, 396], [552, 396], [538, 397], [539, 397], [540,
    → 397], [541, 397], [542, 397], [543, 397], [544, 397], [545, 397], [546, 397],
    → [547, 397], [548, 397], [549, 397], [550, 397], [551, 397], [552, 397], [553,
    → 397], [538, 398], [539, 398], [540, 398], [541, 398], [542, 398], [543, 398],
    → [544, 398], [545, 398], [546, 398], [547, 398], [548, 398], [549, 398], [550,
    → 398], [551, 398], [552, 398], [553, 398], [554, 398], [540, 399], [541, 399],
    → [542, 399], [543, 399], [544, 399], [545, 399], [546, 399], [547, 399], [548,
    → 399], [549, 399], [550, 399], [551, 399], [552, 399], [553, 399], [554, 399],
    → [555, 399], [148, 414], [149, 414], [150, 414], [133, 415], [134, 415], [135,
    → 415], [140, 415], [141, 415], [142, 415], [143, 415], [144, 415], [145, 415],
    → [146, 415], [147, 415], [148, 415], [149, 415], [150, 415], [151, 415], [152,
    → 415], [153, 415], [132, 416], [133, 416], [134, 416], [135, 416], [136, 416],
    → [137, 416], [138, 416], [139, 416], [140, 416], [141, 416], [142, 416], [143,
    → 416], [144, 416], [145, 416], [146, 416], [147, 416], [148, 416], [149, 416],
    → [150, 416], [151, 416], [152, 416], [153, 416], [130, 417], [131, 417], [132,
    → 417], [133, 417], [134, 417], [135, 417], [136, 417], [137, 417], [138, 417],
    → [139, 417], [140, 417], [141, 417], [142, 417], [143, 417], [144, 417], [145,
    → 417], [146, 417], [147, 417], [148, 417], [149, 417], [150, 417], [151, 417],
    → [152, 417], [153, 417], [130, 418], [131, 418], [132, 418], [133, 418], [134,
    → 418], [135, 418], [136, 418], [137, 418], [138, 418], [139, 418], [140, 418],
    → [141, 418], [142, 418], [143, 418], [144, 418], [145, 418], [146, 418], [147,
    → 418], [148, 418], [149, 418], [150, 418], [151, 418], [152, 418], [130, 419],
    → [131, 419], [132, 419], [133, 419], [134, 419], [135, 419], [136, 419], [137,
    → 419], [138, 419], [139, 419], [140, 419], [141, 419], [142, 419], [143, 419],
    → [144, 419], [145, 419], [146, 419], [147, 419], [148, 419], [149, 419], [150,
    → 419], [151, 419], [129, 420], [130, 420], [131, 420], [132, 420], [133, 420],
    → [134, 420], [135, 420], [136, 420], [137, 420], [138, 420], [139, 420], [140,
    → 420], [141, 420], [142, 420], [143, 420], [144, 420], [145, 420], [146, 420],
    → [147, 420], [148, 420], [149, 420], [150, 420], [151, 420], [128, 421], [129,
    → 421], [130, 421], [131, 421], [132, 421], [133, 421], [134, 421], [135, 421],
    → [136, 421], [137, 421], [138, 421], [139, 421], [140, 421], [141, 421], [142,
    → 421], [143, 421], [144, 421], [145, 421], [146, 421], [147, 421], [148, 421],
    → [149, 421], [150, 421], [151, 421], [128, 422], [129, 422], [130, 422], [131,
    → 422], [132, 422], [133, 422], [134, 422], [135, 422], [136, 422], [137, 422],
    → [138, 422], [139, 422], [140, 422], [141, 422], [142, 422], [143, 422], [144,
    → 422], [145, 422], [146, 422], [147, 422], [148, 422], [149, 422], [150, 422],
    → [128, 423], [129, 423], [130, 423], [131, 423], [132, 423], [133, 423], [134,
    → 423], [135, 423], [136, 423], [137, 423], [138, 423], [139, 423], [140, 423],
    → [141, 423], [142, 423], [143, 423], [144, 423], [145, 423], [146, 423], [147,
    → 423], [130, 424], [131, 424]
RETVAL:
 True
 RVEC:
 [[ 0.03855443]
 [ 1.87075877]
 [ 1.2258401 ]]
 TVEC:
 [[ 20.27287811]
 [-20.56860063]
 [ 14.63319462]]
Rotation Matrix:
 [[-0.61747952 -0.40752017  0.67278998]
 [ 0.45416253  0.51364266  0.72794754]
 [-0.64222694  0.75504869 -0.13208343]]
```

```

Inverse Rotation Matrix:
 [[-0.61747952  0.45416253 -0.64222694]
 [-0.40752017  0.51364266  0.75504869]
 [ 0.67278998  0.72794754 -0.13208343]]

Translation Vec:
 [[ 20.27287811]
 [-20.56860063]
 [ 14.63319462]]

Point:
 [[ 39.8]
 [ 55.5]
 [  0. ]]

inverse of Homogenous transform:
 [[-12.51808709   9.20718158 -13.0197884 ]
 [  8.38211961 -10.56491075 -15.53029502]
 [  9.84506665  10.65219796  -1.9328026 ]]

Euler Angles:
 [[-0.61747952 -0.40752017  0.67278998]
 [ 0.45416253  0.51364266  0.72794754]
 [-0.64222694  0.75504869 -0.13208343]]
```

## 3.2 Source

```python
#!/usr/bin/env python
import roslib
import sys
import rospy
import cv2
import numpy as np

from std_msgs.msg import String
from cv_bridge import CvBridge, CvBridgeError
from sensor_msgs.msg import Image


class CameraCalibration:
    def __init__(self):
        # Image publisher
        self.image_gray_pub = rospy.Publisher("/image_processing/bin_gray_img", Image , queue_size = 1)
        self.image_black_pub = rospy.Publisher("/image_processing/bin_black_img", Image, queue_size = 1)
        # Image source
        self.image_sub = rospy.Subscriber("/camera/color/image_raw", Image, self.callback, queue_size = 1)
        # OpenCV
        self.bridge = CvBridge()

    def callback(self, data):
        try:
            cv_image = self.bridge.imgmsg_to_cv2(data, "bgr8")
        except CvBridgeError as e:
            print(e)

        # Convert to grayscale
        gray_img = cv2.cvtColor(cv_image, cv2.COLOR_BGR2GRAY)
        try:
            self.image_gray_pub.publish(self.bridge.cv2_to_imgmsg(gray_img, "mono8"))
        except CvBridgeError as e:
            print(e)

        # Convert to B/W image
        bi_gray_max = 255
        bi_gray_min = 250
        ret, black_img = cv2.threshold(gray_img, bi_gray_min, bi_gray_max, cv2.THRESH_BINARY)
        try:
            self.image_black_pub.publish(self.bridge.cv2_to_imgmsg(black_img, "mono8"))
        except CvBridgeError as e:
            print(e)

        # Scan for white pixels and remember coordinates
        coordinates = []
        for y in range(480):
            for x in range(640):
                if black_img[y, x] == 255:
                    coordinates.append([x, y])
        print("%s Coordinates: \n %s" % (str(len(coordinates)), ', '.join(str(x) for x in coordinates)))

        # Compute the extrinsic parameters with SolvePNP
        fx = 614.1699
        fy = 614.9002
        cx = 329.9491
        cy = 237.2788
        k1 = 0.1115
        k2 = -0.1089
        p1 = 0
        p2 = 0

        camera_mat = np.zeros((3, 3, 1))
        camera_mat[:, :, 0] = np.array([[fx, 0, cx],
            [0, fy, cy],
            [0, 0, 1]])
        dist_coeffs = np.zeros((4, 1))
        dist_coeffs[:, 0] = np.array([[k1, k2, p1, p2]])

        # Object points
        obj_points = np.zeros((6, 3, 1))
        obj_points[:, :, 0] = np.array([
            [00.0, 00.0, 00.0], [40.2, 00.0, 00.0],
            [00.0, 27.7, 00.0], [39.7, 28.0, 00.0],
            [00.0, 55.8, 00.0], [39.8, 55.5, 00.0]])
        # Cluser image points
        criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
        ret, label ,center = cv2.kmeans(np.float32(np.array(coordinates)), 6, None, criteria, 10, cv2.KMEANS_RANDOM_CENTERS)

        retval, rvec, tvec = cv2.solvePnP(obj_points, center, camera_mat, dist_coeffs)
        print("RETVAL: \n %s \n RVEC: \n %s \n TVEC: \n %s" % (retval, rvec, tvec))

        # Calculate rotation matrix
        rmat = np.zeros((3,3))
        cv2.Rodrigues(rvec, rmat, jacobian=0)
        print("Rotation Matrix: \n %s \n" % rmat)
        print("Inverse Rotation Matrix: \n %s \n" % np.linalg.inv(rmat))
        print("Translation Vec: \n %s \n" % tvec)
        print("Point: \n %s \n" % obj_points[-1])

        print("inverse of Homogenous transform: \n %s \n" % np.multiply(np.linalg.inv(rmat), tvec))

        # Calculate angles
        print("Euler Angles: \n %s" % rmat)

    # Checks if a matrix is a valid rotation matrix.
    def isRotationMatrix(R) :
        Rt = np.transpose(R)
        shouldBeIdentity = np.dot(Rt, R)
        I = np.identity(3, dtype = R.dtype)
        n = np.linalg.norm(I - shouldBeIdentity)
        return n < 1e-6


    # Calculates rotation matrix to euler angles
    # The result is the same as MATLAB except the order
    # of the euler angles ( x and z are swapped ).
    def rotationMatrixToEulerAngles(R) :

        assert(isRotationMatrix(R))

        sy = math.sqrt(R[0,0] * R[0,0] +  R[1,0] * R[1,0])

        singular = sy < 1e-6

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

        return np.array([x, y, z])

def main(args):
    rospy.init_node('camera_calibration', anonymous = True)
    cc = CameraCalibration()

    try:
        rospy.spin()
    except KeyboardInterrupt:
        print("Shutting down.")

if __name__ == '__main__':
    main(sys.argv)
```