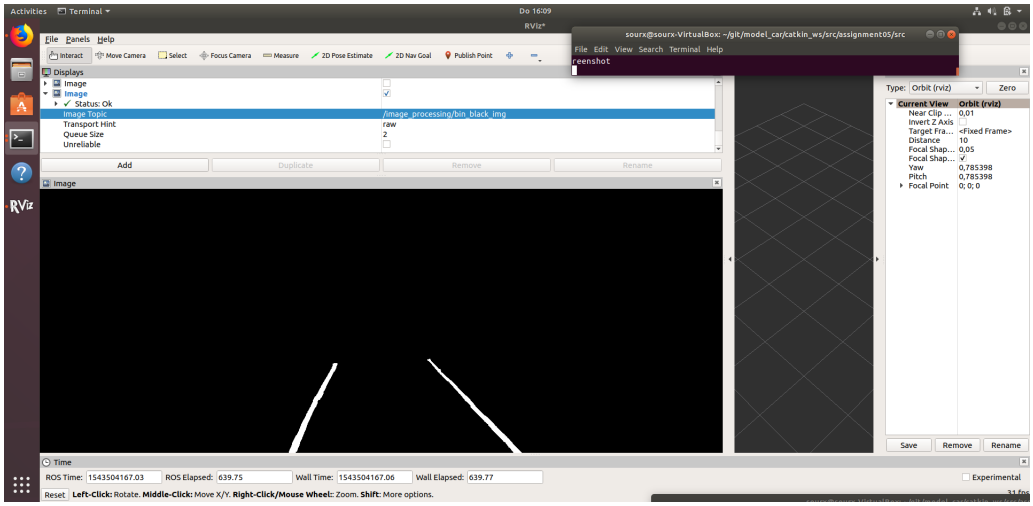
1 Aufabe 1



See the code at the end.

2 Aufgabe 2

Sadly we had a lot of obstacles costing us hours of worktime, thus the code of the second Exercise is not debugged and may not work. It is very unfortunate that there is supposedly no tutor assigned on thursdays whom we could have asked for help and also the ros wiki was down from at least 10:00 to 18:00 where we planned to have our main working time. Also the all the batteries were drained in the end and we could not test our programs on a running car, which of course may be prevented with proper rosbags in the future.

```
1  import roslib
2  import sys
3  import rospy
4  import cv2
5  import sklearn
6  import numpy as np
7
8  from std_msgs.msg import String
9  from cv_bridge import CvBridge, CvBridgeError
10 from sensor_msgs.msg import Image
11
12
13 class LaneSegment:
14     def __init__(self):
15         # Image publisher
16         self.image_gray_pub = rospy.Publisher("/image_processing/bin_gray_img", Image
           →  , queue_size = 1)
17         self.image_black_pub = rospy.Publisher("/image_processing/bin_black_img",
           →  Image, queue_size = 1)
18         self.ransac = rospy.Publisher("/image_processing/bin_ransac_lines", Image,
           →  queue_size = 1)
19
20         # Image source
21         self.image_sub = rospy.Subscriber("/camera/color/image_raw", Image, self.
           →  callback, queue_size = 1)
22         # OpenCV
23         self.bridge = CvBridge()
24
25     def callback(self, data):
```

```
26
27          # +++ Exercise 1 +++
28          try:
29              cv_image = self.bridge.imgmsg_to_cv2(data, "bgr8")
30          except CvBridgeError as e:
31              print(e)
32
33          # Convert to white only
34          gray_img = cv2.cvtColor(cv_image, cv2.COLOR_BGR2GRAY)
35          try:
36              self.image_gray_pub.publish(self.bridge.cv2_to_imgmsg(gray_img, "mono8"))
37          except CvBridgeError as e:
38              print(e)
39          # Convert to white lines only
40          bi_gray_max = 255
41          bi_gray_min = 250
42
43          ret, black_img = cv2.threshold(gray_img, bi_gray_min, bi_gray_max, cv2.
            → THRESH_BINARY)
44          try:
45              self.image_black_pub.publish(self.bridge.cv2_to_imgmsg(black_img, "mono8"
                → ))
46          except CvBridgeError as e:
47              print(e)
48
49          # +++ Exercise 2 +++
50
51          left_image = black_img[0:319]
52          right_image = black_img[320:639]
53
54          points_left = ([][])*3
55          points_right = ([][])*3
56
57          # left
58          # search from top to bottom to find a starting white point
59          for y in range(480):
60              for x in range(320):
61                  if left_image[x][y] == 255:
62                     points_left[0][0]=x
63                     points_left[0][1]=y
64
65          # search from bottom to top to find an ending white point
66          for y in range(480):
67              for x in range(320):
68                  if left_image[x][480-y] == 255:
69                     points_left[1][0]=x
70                     points_left[1][1]=y
71
72          # search in the middle of both white points to find a corresponding white
            → point
73          for x in range(320):
74              if left_image[x][points_left[1][0]-points_left[2][0] == 255:
75                 points_left[2][0]=x
76                 points_left[2][1]=y
77
78          # right
79          # analogous for the right pixels
80          for y in range(480):
81              for x in range(320):
82                  if right_image[x][y] == 255:
83                     points_right[0][0]=x
84                     points_right[0][1]=y
85
```

```
86          for y in range(480):
87              for x in range(320):
88                  if right_image[x][480-y] == 255:
89                     points_right[1][0]=x
90                     points_right[1][1]=y
91
92          for x in range(320):
93              if right_image[x][foundbot[1]-foundtop[1]] == 255:
94                 points_right[2][0]=x
95                 points_right[2][1]=y
96
97
98          ransac[0] = sklearn.linear_model.RANSACRegressor(min_samples=3, max_trials=1)
99          ransac[1] = sklearn.linear_model.RANSACRegressor(min_samples=3, max_trials=1)
100
101         try:
102             self.image_black_pub.publish(ransac)
103         except CvBridgeError as e:
104             print(e)
105
106 def main(args):
107     rospy.init_node('lanesegment', anonymous =True)
108     LS = LaneSegment()
109
110     try:
111         rospy.spin()
112     except KeyboardInterrupt:
113         print("Shutting down")
114
115 if __name__ == "__main__":
116     main(sys.argv)
```