Edge Detection 5p

```
def main(argv):
    cv2.namedWindow("park_image", 0)
    cv2.namedWindow("one_park_place_image", 0)

    pkm_file = open('parking_map_python.txt', 'r')
    pkm_lines = pkm_file.readlines()
    pkm_coordinates = []

    for line in pkm_lines:
        st_line = line.strip()
        sp_line = list(st_line.split(" "))
        pkm_coordinates.append(sp_line)

    test_images = [img for img in glob.glob("test_images/*.jpg")]
    test_images.sort()
    for img in test_images:
        park_image = cv2.imread(img)
        cv2.waitKey(2)
        cv2.imshow('park_image', park_image)
        for one_c in pkm_coordinates:
            pts = [((float(one_c[0])), float(one_c[1])),
                   ((float(one_c[2])), float(one_c[3])),
                   ((float(one_c[4])), float(one_c[5])),
                   ((float(one_c[6])), float(one_c[7]))]
            # print(pts)
            # https://www.pyimagesearch.com/2014/08/25/4-point-opencv-getperspective-transform-example/
            warped_image = four_point_transform(park_image, np.array(pts))
            one_park_place_image = cv2.resize(warped_image, (80, 80))

            cv2.imshow('one_park_place_image', one_park_place_image)
            cv2.waitKey(0)
        cv2.imshow('park_image', park_image)
        key = cv2.waitKey(0)
        if key == 27:  # exit on ESC
            break
```

TASK – parking lot occupation detection

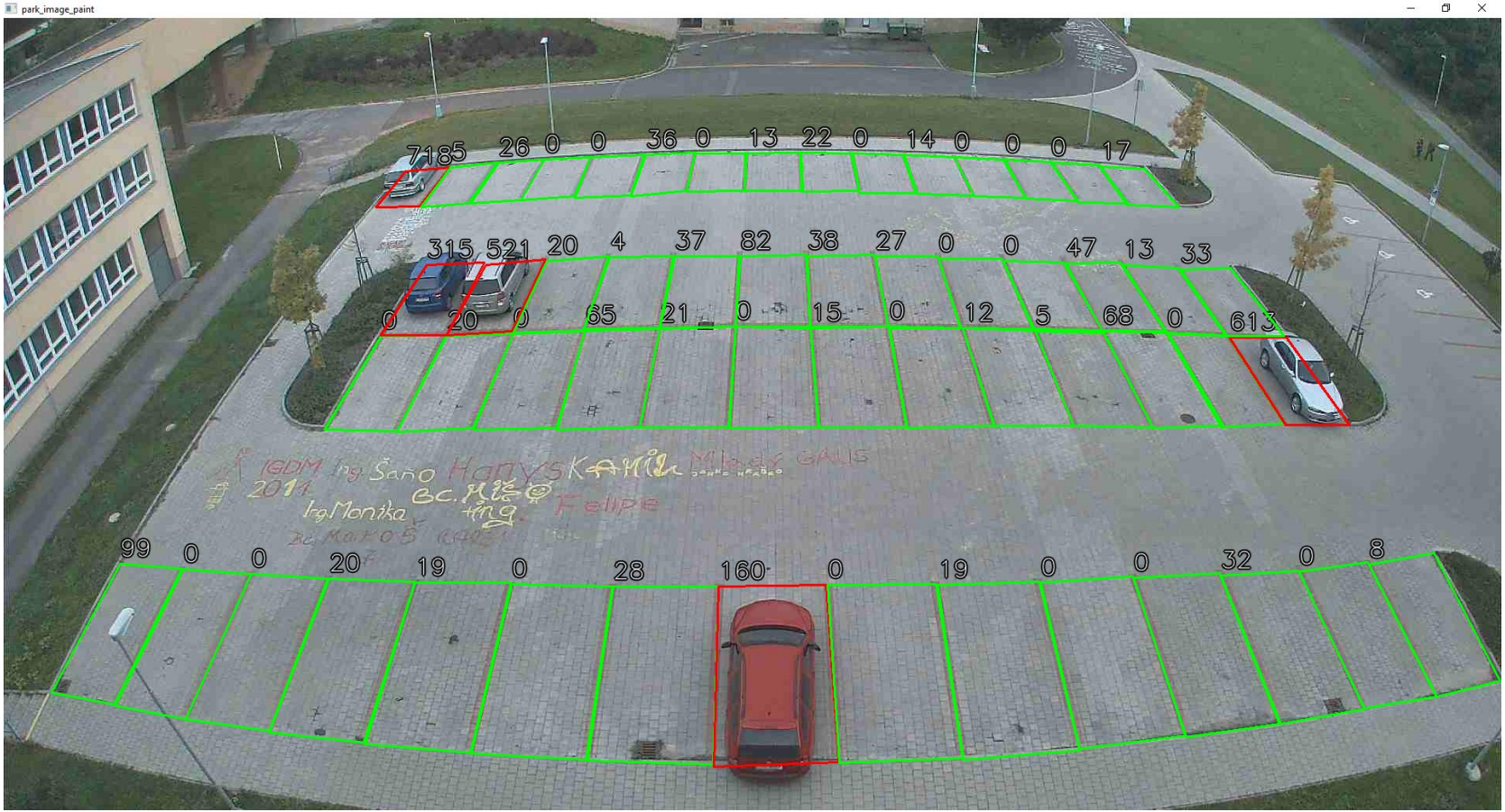
- continue with the template:
https://mrl.cs.vsb.cz/data/vyuka/zao/parking/parking_zao_template.zip

- experiment with different types of image filtering
- experiment with different types of edge detectors
- Hint: `cv.CountNonZero` after edge detection
- read .txt file with ground truth data of each parking place image (in "test_images_zao" folder) and calculate accuracy for each detector configuration:

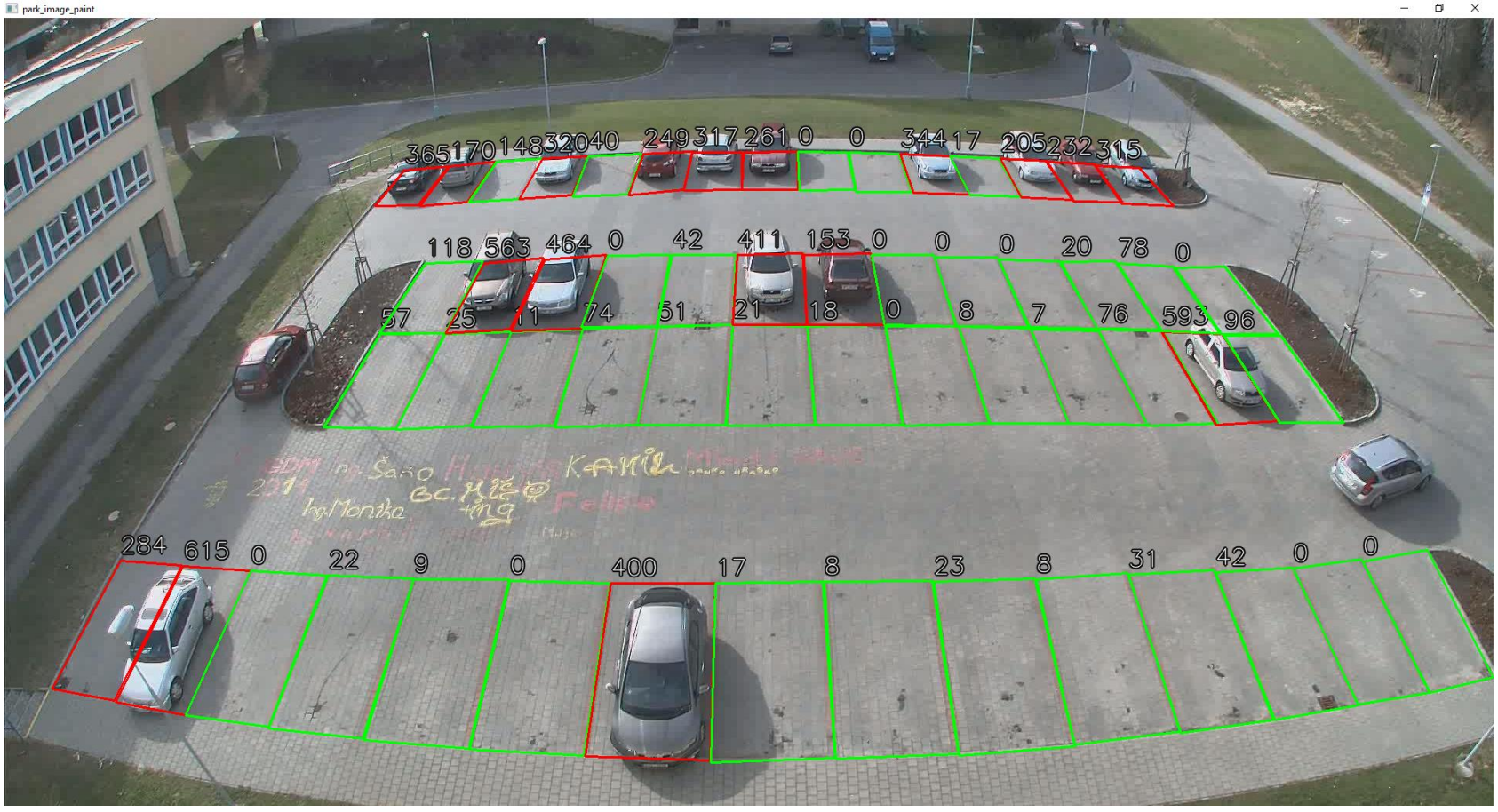
$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

where TP = True positive; FP = False positive; TN = True negative; FN = False negative

Edge Detection



Edge Detection



Edge Detection