EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

Image Analysis II - Exercises

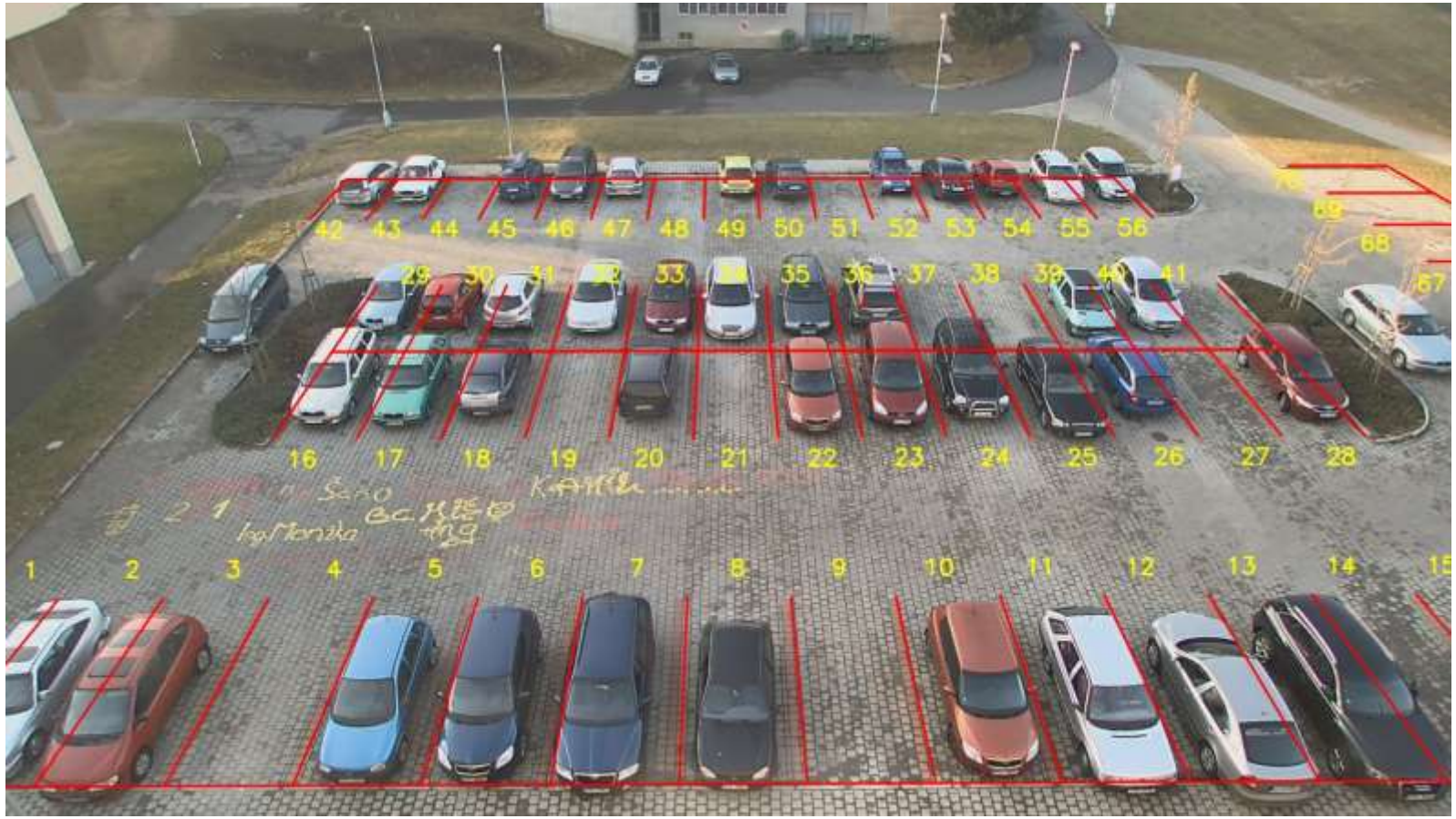
Radovan Fusek

VSB TECHNICAL UNIVERSITY OF OSTRAVA | FACULTY OF ELECTRICAL ENGINEERING AND COMPUTER SCIENCE | DEPARTMENT OF COMPUTER SCIENCE

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

Exercise (Parking, C++)

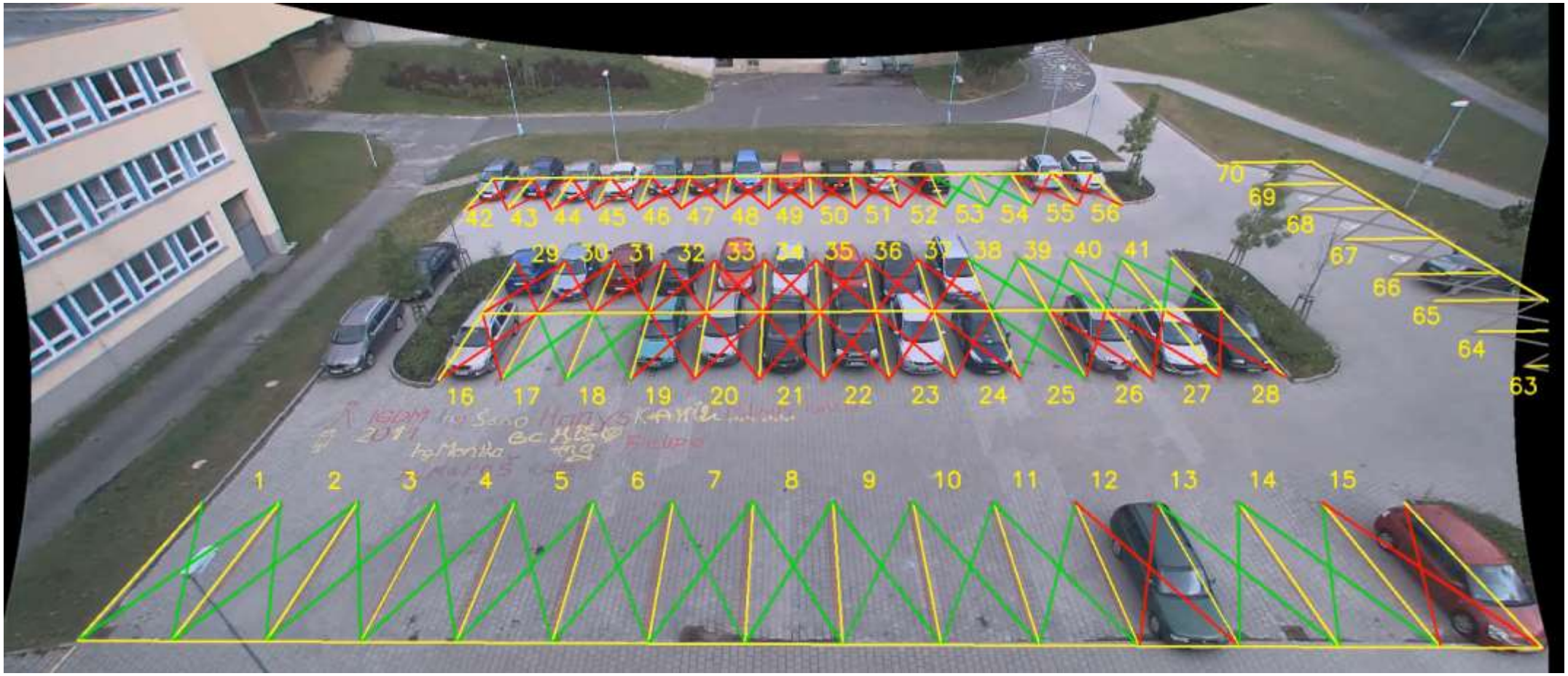
DETECTING FREE/OCCUPIED PLACES IN PARKING LOTS



EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

Exercise (Parking, C++)

DETECTING FREE/OCCUPIED PLACES IN PARKING LOTS



Exercise (Parking, C++)

DETECTING FREE/OCCUPIED PLACES IN PARKING LOTS

Motivation:
The vehicle detection systems using images have been very useful in the recent years. Especially nowadays in the cities, the increasing number of vehicles brings a major problem. The car detection systems can be important, especially for drivers who are looking for vacant spaces in the parking lots, for traffic analysis, for intelligent scheduling, for smart cities and so on.

Input Data:
The training data with a basic template (C++/OpenCV) can be found in the following link:

http://mrl.cs.vsb.cz/data/vyuka/ano2/parking_template_clear_dlib.zip

Exercise (Parking, C++)

description of template:

- training and testing data are in the “testImages” and “trainImages” folders
- each image is named as free_xx.png or full_xx.png (the name of the images represents the state of parking space)
- functions for loading training/testing images are already implemented - train_parking(), test_parking()
- **the training and prediction steps are missing** - You can use any available libraries to solve this detection task. The use of the provided main.cpp template is not required.

Exercise (Parking, C++)

description of template:

```
36 using net_type = loss_multiclass_log<
37                                 fc<2,
38                                 relu<fc<84,
39                                 relu<fc<120,
40                                 max_pool<2,2,2,2,relu<con<32,3,3,1,1,
41                                 max_pool<2,2,2,2,relu<con<16,5,5,1,1,
42                                 max_pool<2,2,2,2,relu<con<6,5,5,1,1,
43                                 input<matrix<unsigned char>>
44                                 >>>>>>>>>>>>>>>;
45
```

Exercise (Parking, C++)

description of template:

```
46 int main(int argc, char** argv)
47 {
48     cout << "Train OpenCV Start" <<endl;
49     train_parking_dlib();
50     cout << "Train OpenCV End" <<endl;
51
52     cout << "Test OpenCV Start" <<endl;
53     test_parking_dlib();
54     cout << "Test OpenCV End" <<endl;
55
56 }
```

Exercise (Parking, C++)

description of template:

```
59 void train_parking_dlib()
60 {
61     //vectors for training images and labels
62     std::vector<unsigned long> train_labels;
63     std::vector<matrix<unsigned char>> train_images;
64     //training file
65     fstream train_file("train_images_dlib.txt");
66     string train_path;
67
68     while(train_file >> train_path)
69     {
70         Mat frame;
71         //read training image
72         frame = imread(train_path, 0);
73         resize(frame, frame, Size(40, 40));
74         cv_image<unsigned char> cimg(frame);
75         matrix<unsigned char> dlibFrame = dlib::mat(cimg);
76
77         train_images.push_back(dlibFrame);
78         // label = 1;//occupied place
79         // label = 0;//free place
80         unsigned long label = 0;
81         //based on the image name set label
82         if (train_path.find("full") != std::string::npos) label = 1;
83         //training label for each parking space
84         train_labels.push_back(label);
85
86     }
87
88     //TODO TRAINING FUNCTIONS
```

Exercise (Parking, C++)

description of template:

```
111 void test_parking_dlib()
112 {
113
114     space *spaces = new space[spaces_num];
115     load_parking_geometry("parking_map.txt", spaces);
116
117     fstream test_file("test_images.txt");
118     ofstream out_label_file("out_prediction.txt");
119     string test_path;
120
121     net_type net;
122     deserialize("LeNetTest.dat") >> net;
123
124     while(test_file >> test_path)
125     {
126         //read testing images
127         Mat frame = imread( test_path, 1 );
128         Mat draw_frame = frame.clone();
129         cvtColor( frame, frame, COLOR_BGR2GRAY );
130
131         std::vector<Mat> test_images;
132         extract_space(spaces, frame, test_images);
```

Exercise (Parking, C++)

description of template:

```
132         extract_space(spaces, frame, test_images);
133
134         int colNum = 0;
135         for(int i = 0; i < test_images.size(); i++)
136         {
137             Mat pFrame = test_images[i];
138             resize(pFrame, pFrame, Size(40, 40));
139             //TODO PREDICTION FUNCTION
140             cv_image<unsigned char> cimg(pFrame);
141             matrix<unsigned char> dlibFrame = dlib::mat(cimg);
142             unsigned long predict_label = net(dlibFrame);
143
144             out_label_file << predict_label << endl;
145             spaces[i].occup = predict_label;
146             imshow("test_img", test_images[i]);
147             waitKey(2);
148         }
149
150         //draw detection
151         draw_detection(spaces, draw_frame);
152         namedWindow("draw_frame", 0);
153         imshow("draw_frame", draw_frame);
154         waitKey(0);
```

Exercise (Parking, C++)

Output:

If you successfully run the template, you obtain this output. It means that the accuracy of the detector is aprox. 32%. The accuracy is low because each parking space is labeled as occupied - line 82 in main.cpp. The goal is to implement better prediction approach.

```
Train OpenCV Start
Train images: 5656
Train labels: 5656
Train OpenCV End
Test OpenCV Start
testing num_right: 426
testing num_wrong: 918
testing accuracy:  0.316964
Test OpenCV End
```

Exercise (Parking, C++)

Hints:

Since we want to label each parking space as free (0) or occupied (1), this recognition problem can be solved using classical binary classifiers (SVM, neural networks). To train the classifiers, you can use the provided training data in the “trainImages” folder. As the input for the classifiers, you can use the whole image or you can use feature extraction approaches (e.g. histograms of oriented gradients, local binary patterns).
Alternatively, you can skip the training process and use simple color or gradient information for example. In that case, you can use only the test_parking() function without the training.
The provided template is based on the **OpenCV** library https://opencv.org/
Installation in Linux: https://www.learnopencv.com/install-opencv3-on-ubuntu/
Installation in Windows: https://www.learnopencv.com/install-opencv3-on-windows/
Installation in MacOS: https://www.learnopencv.com/install-opencv3-on-macos/
Simple install for Windows without cmake using NuGet:

http://funvision.blogspot.com/2017/04/simple-install-opencv-visual-studio.html
https://www.nuget.org/packages/opencv.win.native/320.1.1-vs141

Exercise (Parking, C++)

Hints:

Alternatively, you can install OpenCV from the Ubuntu or Debian repository:

```
sudo apt-get install libopencv-dev python3-opencv
```

You can find the several tutorials in the following link: https://docs.opencv.org/3.4.2/d9/df8/tutorial_root.html

Dlib library represents another option how to solve this detection problem

- Installation in Windows https://www.learnopencv.com/install-dlib-on-windows/
- Installation in Linux https://www.learnopencv.com/install-dlib-on-ubuntu/
- Installation in MacOS: https://www.learnopencv.com/install-dlib-on-macos/
- You can follow this tutorial: http://dlib.net/dnn_introduction_ex.cpp.html

You can also use **Keras, Caffe, TensorFlow**, etc.

Exercise (Parking, Python)

http://mrl.cs.vsb.cz/data/vyuka/ano2/parking_template_clear_python.zip

description of template:

```
17 file1 = open('parking_map_python.txt', 'r')
18 Lines = file1.readlines()
19 list_parking_coordinates = []
20 count = 0
21 # Strips the newline character
22 for line in Lines:
23     count += 1
24     oneLine = line.strip()
25     li = list(oneLine.split(" "))
26     #print(li)
27     list_parking_coordinates.append(li)
28     #print("Line{}: {}".format(count, line.strip()))
29     #print(oneLine)
30
31 train_images_free = [img for img in glob.glob("train_images/free/*.jpg")]
32 train_images_full = [img for img in glob.glob("train_images/full/*.jpg")]
33 test_images = [img for img in glob.glob("test_images/*.jpg")]
34 test_images.sort()
```

Exercise (Parking, Python)

description of template:

```
91 cv2.namedWindow("one_img", 0)
92
93 result_list = []
94
95 for img in test_images:
96     one_img = cv2.imread(img)
97     one_img_paint = one_img.copy()
98     for one_line in list_parking_coordinates:
99         #print(one_line)
100        pts = [((float(one_line[0])), float(one_line[1])),
101                ((float(one_line[2])), float(one_line[3])),
102                ((float(one_line[4])), float(one_line[5])),
103                ((float(one_line[6])), float(one_line[7]))]
104        #print(pts)
105
106        point_1 = (int(one_line[0]), int(one_line[1]))
107        point_2 = (int(one_line[2]), int(one_line[3]))
108        point_3 = (int(one_line[4]), int(one_line[5]))
109        point_4 = (int(one_line[6]), int(one_line[7]))
110
111        #https://www.pyimagesearch.com/2014/08/25/4-point-opencv-
112        warped = four_point_transform(one_img, np.array(pts))
113        warped_resize = cv2.resize(warped, (80, 80))
114
115        blur_img = cv2.GaussianBlur(warped_resize,(3,3),0)
```

Exercise (Parking, Python)

description of template:

```
115         blur_img = cv2.GaussianBlur(warped_resize,(3,3),0)
116
117         gray = cv2.cvtColor(blur_img, cv2.COLOR_BGR2GRAY)
118         edges = cv2.Canny(gray, 40, 120)
119
120         nzCount = cv2.countNonZero(edges)
121         #print(nzCount)
122
123         if(nzCount > 250):
124             #cv2.rectangle(paint_img, yolo_rect_paint[0], yolo_rect_paint[1], (0,255,255),10)
125             cv2.line(one_img_paint, point_1, point_2, (0, 0, 255), 2)
126             cv2.line(one_img_paint, point_2, point_3, (0, 0, 255), 2)
127             cv2.line(one_img_paint, point_3, point_4, (0, 0, 255), 2)
128             cv2.line(one_img_paint, point_1, point_4, (0, 0, 255), 2)
129             result_list.append(1)
130         else:
131             cv2.line(one_img_paint, point_1, point_2, (0, 255, 0), 2)
132             cv2.line(one_img_paint, point_2, point_3, (0, 255, 0), 2)
133             cv2.line(one_img_paint, point_3, point_4, (0, 255, 0), 2)
134             cv2.line(one_img_paint, point_1, point_4, (0, 255, 0), 2)
135             #cv2.circle(one_img, (centerXX, centerYY), int(10), (0, 0, 255), -1)
136             result_list.append(0)
137         cv2.imshow('warped_resize', warped_resize)
138         cv2.imshow('edges', edges)
139         cv2.waitKey(2)
```

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

Exercise (Pedestrian, Python)

Exercise (Pedestrian, Python)

description of template

```python
import cv2

# Initializing the HOG person
# detector
hog = cv2.HOGDescriptor()
hog.setSVMDetector(cv2.HOGDescriptor_getDefaultPeopleDetector())

# Reading the Image
image = cv2.imread('celluloid-shot0022.jpg')
cv2.namedWindow("resize_image", 0)

resize_image = cv2.resize(image, (1280, 720))
# Resizing the Image

# Detecting all the regions in the
# Image that has a pedestrians inside it
(regions, _) = hog.detectMultiScale(resize_image,
                                    winStride=(4, 4),
                                    padding=(4, 4),
                                    scale=1.05)

# Drawing the regions in the Image
for (x, y, w, h) in regions:
    cv2.rectangle(resize_image, (x, y),
                  (x + w, y + h),
                  (0, 0, 255), 2)

# Showing the output Image
cv2.imshow("resize_image", resize_image)
cv2.waitKey(0)
```

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

Exercise (Pedestrian, Python)

```
[ INFO:0] Initialize OpenCL runtime...
(array([[1537,  606,  189,  379],
       [1740,  803,   70,  141]], dtype=int32), array([[1.06816489],
       [0.66262385]]))
```

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

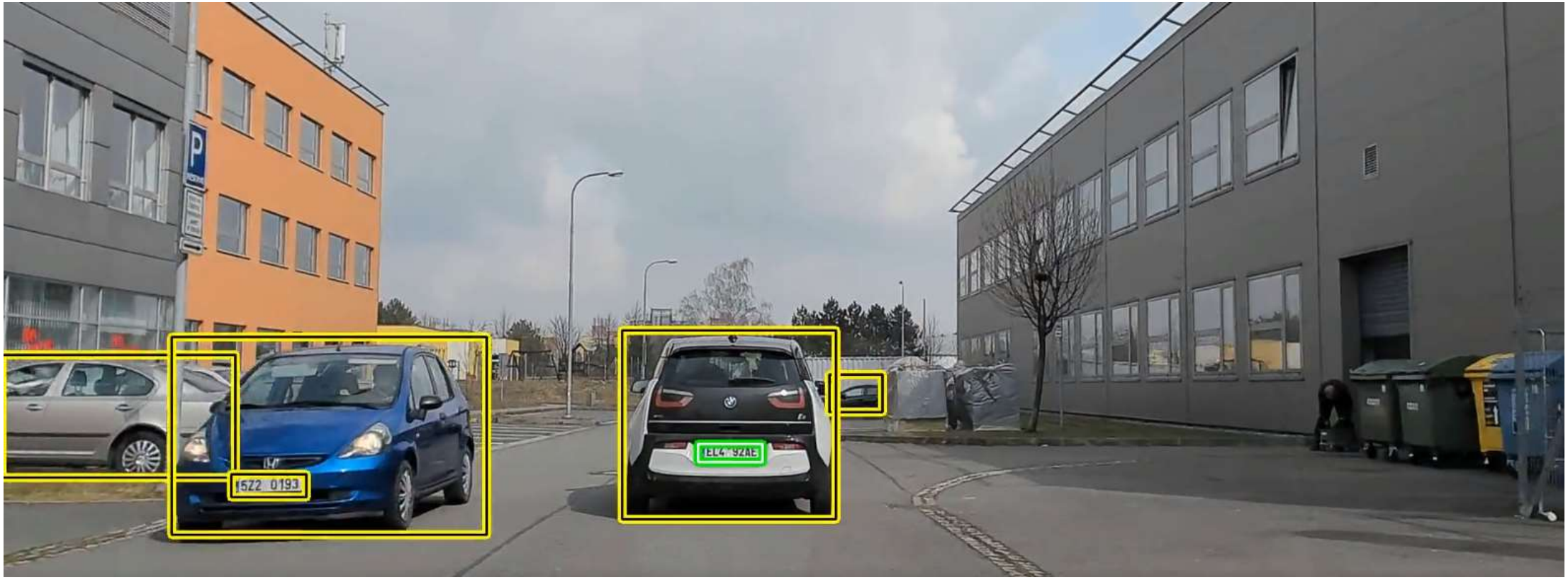
Exercise (Pedestrian, IR Images)

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

Exercise (Pedestrian, IR Images)

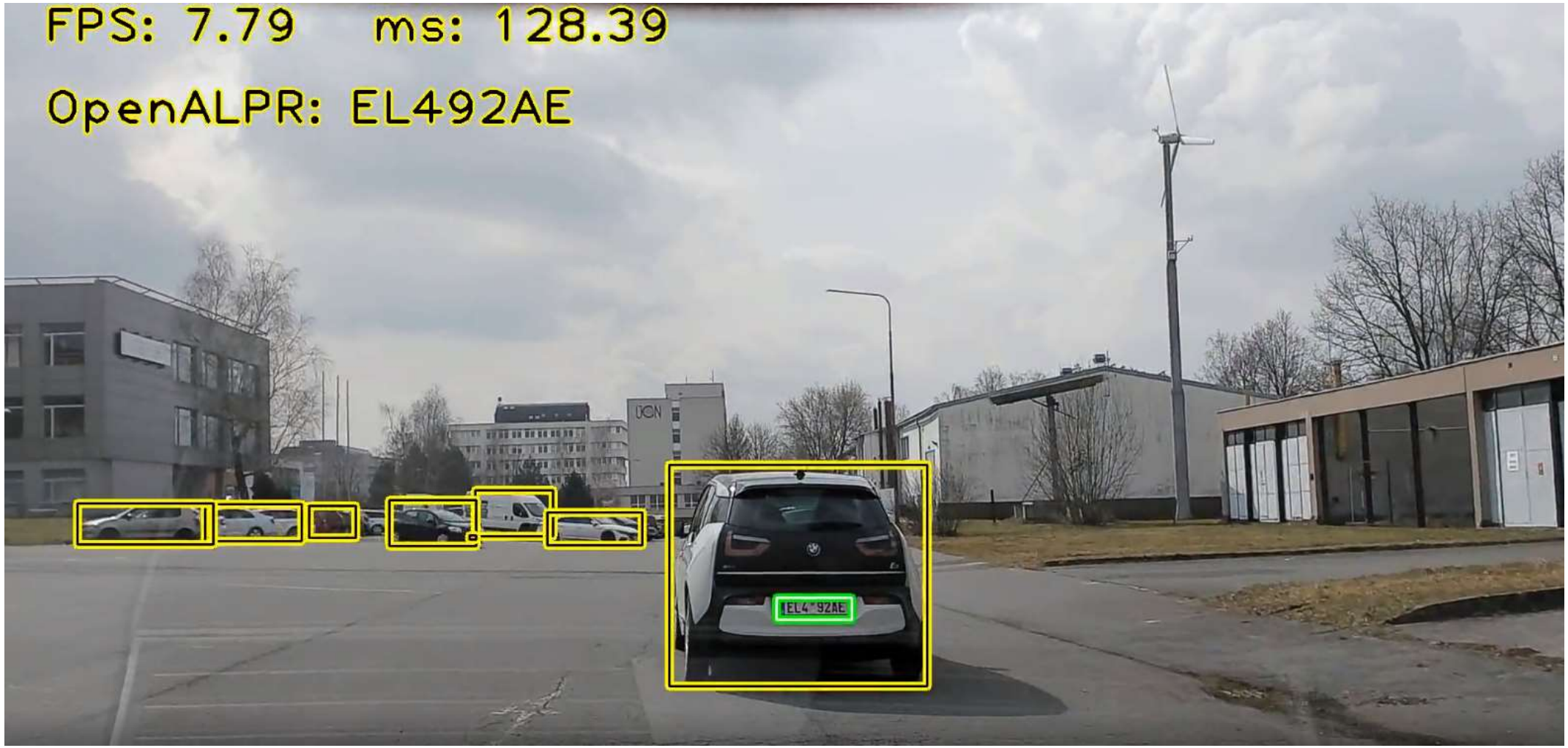
EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

Exercise (R-CNN, Python)



EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

Exercise (R-CNN, Python)



Exercise (R-CNN, Python)

description of template:

```python
# PyTorch libraries and modules
import torch
from torch.autograd import Variable
from torch.nn import Linear, ReLU, CrossEntropyLoss, Sequential, Conv2d, MaxPool2d, Module, Softmax, BatchNorm2d, Dropout
from torch.optim import Adam, SGD

from torchvision import models
from PIL import Image
import torchvision.transforms as transforms
import torchvision

coco_names = [
    '__background__', 'person', 'bicycle', 'car', 'motorcycle', 'airplane', 'bus',
    'train', 'truck', 'boat', 'traffic light', 'fire hydrant', 'N/A', 'stop sign',
    'parking meter', 'bench', 'bird', 'cat', 'dog', 'horse', 'sheep', 'cow',
    'elephant', 'bear', 'zebra', 'giraffe', 'N/A', 'backpack', 'umbrella', 'N/A', 'N/A',
    'handbag', 'tie', 'suitcase', 'frisbee', 'skis', 'snowboard', 'sports ball',
    'kite', 'baseball bat', 'baseball glove', 'skateboard', 'surfboard', 'tennis racket',
    'bottle', 'N/A', 'wine glass', 'cup', 'fork', 'knife', 'spoon', 'bowl',
    'banana', 'apple', 'sandwich', 'orange', 'broccoli', 'carrot', 'hot dog', 'pizza',
    'donut', 'cake', 'chair', 'couch', 'potted plant', 'bed', 'N/A', 'dining table',
    'N/A', 'N/A', 'toilet', 'N/A', 'tv', 'laptop', 'mouse', 'remote', 'keyboard', 'cell phone',
    'microwave', 'oven', 'toaster', 'sink', 'refrigerator', 'N/A', 'book',
    'clock', 'vase', 'scissors', 'teddy bear', 'hair drier', 'toothbrush'
]
```

Exercise (R-CNN, Python)

description of template:

```
37 cv2.namedWindow("detection", 0)
38
39 test_images = [img for img in glob.glob("test_images/*.jpg")]
40 test_images.sort()
41
42 def main():
43
44     model = torchvision.models.detection.fasterrcnn_resnet50_fpn(pretrained=True)
45
46     device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
47
48     model.eval().to(device)
49     #print(model)
50
51     transformRCNN = transforms.Compose([
52         transforms.ToTensor(),
53     ])
54
```

Exercise (R-CNN, Python)

description of template:

```
55    for img in test_images:
56        one_img = cv2.imread(img)
57        one_img_paint = one_img.copy()
58
59        one_img_rgb = cv2.cvtColor(one_img, cv2.COLOR_BGR2RGB)
60        img_pil = Image.fromarray(one_img_rgb)
61
62        imageRCNN = transformRCNN(img_pil).to(device)
63        imageRCNN = imageRCNN.unsqueeze(0) # add a batch dimension
64        outputsRCNN = model(imageRCNN) # get the predictions on the image
65        pred_classes = [coco_names[i] for i in outputsRCNN[0]['labels'].cpu().numpy()]
66        pred_scores = outputsRCNN[0]['scores'].detach().cpu().numpy()
67        print(pred_scores)
68        pred_bboxes = outputsRCNN[0]['boxes'].detach().cpu().numpy()
69        for i, box in enumerate(pred_bboxes):
70            if ( (pred_classes[i] == "stop sign") and (pred_scores[i] > 0.6)):
71                cv2.rectangle(one_img_paint, (int(box[0]), int(box[1])), (int(box[2]), int(box[3])), (255, 255, 255), 9)
72                cv2.rectangle(one_img_paint, (int(box[0]), int(box[1])), (int(box[2]), int(box[3])), (0, 0, 255), 2)
73
74        cv2.imshow('detection', one_img_paint)
75        key = cv2.waitKey(0)
76        if key == 27: # exit on ESC
77            break
```

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

Exercise (Depth, IR Images)

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

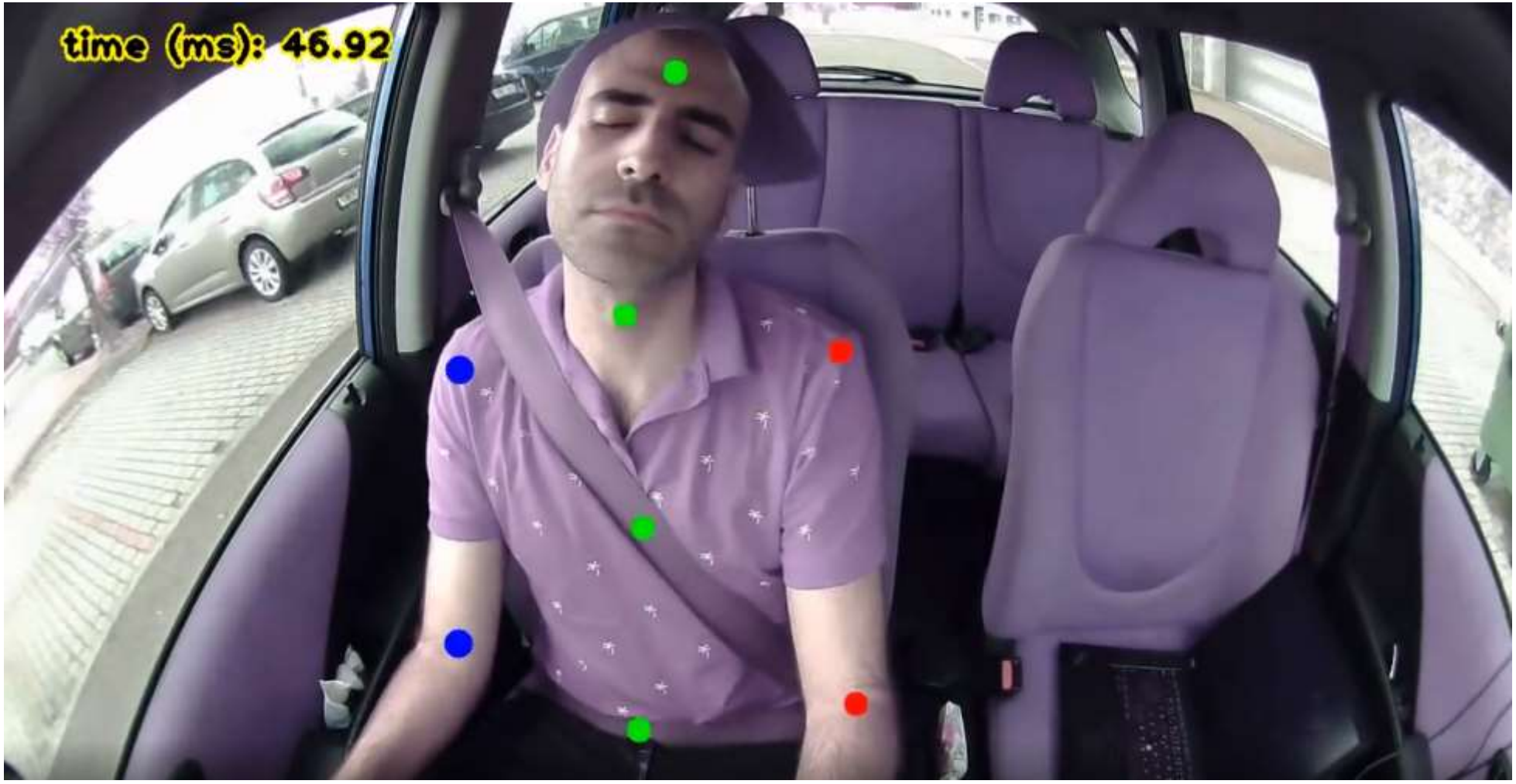
Openpose Library

Exercise (checking the driver state)

Z. Cao, T. Simon, S. Wei and Y. Sheikh, "Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 1302-1310.

EVROPSKÁ UNIE
Evropské strukturální a investiční fondy
Operační program Výzkum, vývoj a vzdělávání

Exercise (checking the driver state)



Z. Cao, T. Simon, S. Wei and Y. Sheikh, "Realtime Multi-person 2D Pose Estimation Using Part Affinity Fields," 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, HI, 2017, pp. 1302-1310.

Exercise (checking the driver state)

description of template:

```
48 def main():
49
50     cv2.namedWindow("detection", 0)
51
52     test_images = [img for img in glob.glob("test_images/*.jpg")]
53     test_images.sort()
54     #print(test_images)
55
56     model = torchvision.models.detection.keypointrcnn_resnet50_fpn(pretrained=True, num_keypoints=17, )
57     device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
58
59     model.eval().to(device)
60     #print(model)
61
62     transformRCNN = transforms.Compose([
63         transforms.ToTensor(),
64     ])
```

Exercise (checking the driver state)

description of template:

```
66    for img in test_images:
67
68        one_img = cv2.imread(img)
69        one_img = cv2.resize(one_img, (1280, 720))
70        one_img_paint = one_img.copy()
71
72        start = time.time()
73
74        one_img_rgb = cv2.cvtColor(one_img, cv2.COLOR_BGR2RGB)
75
76        img_pil = Image.fromarray(one_img_rgb)
77
78        imageRCNN = transformRCNN(img_pil).to(device)
79        imageRCNN = imageRCNN.unsqueeze(0) # add a batch dimension
80
81        with torch.no_grad():
82            outputsRCNN = model(imageRCNN)
83
84        output_image = draw_keypoints(outputsRCNN, one_img_paint)
85
86        end = time.time()
87        print(round(1.0/(end-start),1))
88        image = cv2.putText(output_image, 'FPS: {} ({})'.format(round(1.0/(end-start),1), "pytorch")
89        cv2.imshow('detection', output_image)
90        key = cv2.waitKey(2)
91        if key == 27: # exit on ESC
92            break
```