DETECTING FREE/OCCUPIED PLACES IN PARKING LOTS

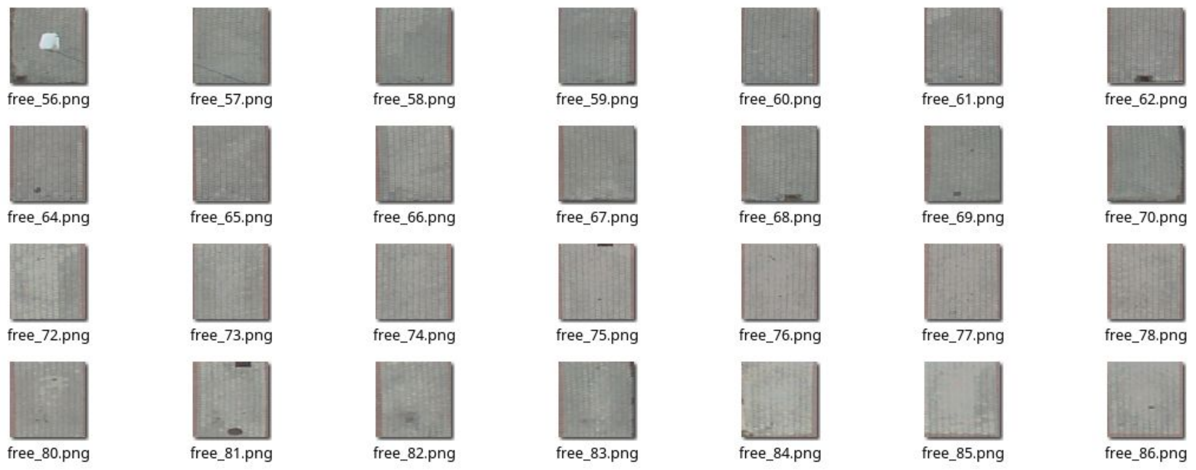
Motivation:
The vehicle detection systems using images have been very useful in the recent years. Especially nowadays in the cities, the increasing number of vehicles brings a major problem. The car detection systems can be important, especially for drivers who are looking for vacant spaces in the parking 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/parking.zip

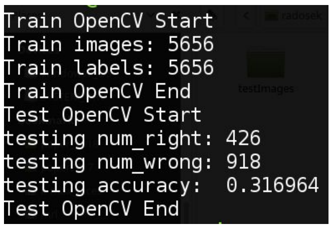
description of template:

- 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.

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
```

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
- 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.

Example projects:

Think about the topics we've discussed so far in the summer school and how you can apply them. Here are some ideas to get you started

- run a numerical optimizer over the meta parameters to find the best classifier
- does the classifier consistently miss some types of images (hint here is to cluster the misclassified images)
- model selection: build a set of classifiers using a number of different meta parameters. Analyze the data in tableau to understand the effects of the parameters on classification performance
- for neural networks: what is the best performing NN architecture? why? What alternatives are there and do they give similar performance (if not, how much worse and why)?