Exercise for MA-INF 2218 Video Analytics SS18
Submission on 02.05.2018
Spatio-Temoral Interest Points

You have to implement your solution using python2.7. You are allowed to use OpenCV 2.x to its full extend. You do not need to implement every single component like Optical Flow on your own. Please comment your code appropriately. You will continue with the code (or the model solution) for the next exercises, so try to organize it neatly.

1. Implement spatio-temporal interest points. This includes

   - smoothing of the image (Eq. 5 from [1])
   - computing the image derivatives in spatial and temporal directions
   - constructing the second moment matrix and smoothing it (Eq. 7 from [1])
   - defining the Harris function (Eq. 8 from [1]) and finding the interest points via non-maximum suppression
   - extract spatio-temporal interest points over multiple scales (see Section 3.1 from [2])

   Hint: Use the video `dummy.avi` for debugging. Among others, you should find interest points at the corners of the rectangle when it changes its direction. Note that due to spatio-temporal scaling, you will probably also find interest points in the white area in the dummy video. *(7 Points)*

2. For each interest point you detected, create a HoG and a HoF feature descriptor. Following [2] (Sec. 3.1), use a spatio-temporal $3 \times 3 \times 2$ grid around the detected point. Then, create an 4-bin HoG and HoF for each cuboid of the grid. Concatenate the descriptors to one large vector. So, in the end you should have for each spatio-temporal interest point a 144 dimensional feature vector $(3 \cdot 3 \cdot 2 \cdot (4 + 4))$. Save a numpy matrix per video (of size $N \times 144$ where N is the number of interest points) using np.save *(7 Points)*

3. Implement a bag-of-words (bag-of-features) encoding using kMeans clustering with 256 clusters to quantize all 144 dimensional vectors you found for a video to a single 256 dimensional histogram. You can use scikit-learn's kmeans implementation. *(4 Points)*

4. On the provided dataset, train your system with the videos specified in `train.txt` and report the accuracy on the test set (`test.txt`). Each line of the (`train.txt` or `test.txt`) contains file name and its label separated by a blank space. There are 10 different classes. After feature quantization with the bag-of-words approach, you have one feature vector per video, so you have a simple classification task. Train a support vector machine (scikit-learn's SVM implementation is recommended) and classify the test videos. Report the accuracy of your classifier. For terms of simplicity, you can use a linear kernel for your SVM. *(2 Points)*

Since video processing is expensive, make sure to write efficient code. There will be up to 4 points per sheet dedicated to your code quality. The classification accuracy is about 63%. If you have any questions, feel free to contact me (**iqbalm@iai.uni-bonn.de**).

[1] I. Laptev: On Space-Time Interest Points, Int. Journal of Computer Vision, 2005
[2] I. Laptev, M. Marszalek, C. Schmid, B. Rozenfeld: Learning realistic human actions from movies, IEEE Conf. on Computer Vision and Pattern Recognition, 2008