Exercise 8 for MA-INF 2201 Computer Vision WS19/20
Submission on 14.12.2019

1. **Eigen Faces**: Use the Eigen Faces idea to perform face detection and face recognition. For this exercise you are permitted to use the *scikit-learn* library.

   (a) Using the LFW dataset (first split it randomly into train and test set and then only use the train for learning. Use 25% of the dataset for test) and PCA learn the k principal components of the face images. Show the first 10 Eigen Faces. Find the reconstruction error of an image from `data/exercise1/detect/face` using the k principal components. Set k to 100.
   (3 Points)

   (b) Using the learned PCA, detect the face images from `data/exercise1/detect` folder. The images are organized into two folders `face` and `other` which indicated whether an image is of a face or not. You should perform all the necessary preprocessing on the images. Calculated the accuracy of your method.
   (2 Points)

   (c) Using the learned PCA, recognize the images in the created test set. This means that given a face image find out which person this image belongs to. You should use a Nearest Neighbor classifier. Calculate the accuracy of your recognition method.
   (2 Points)
   NOTE: You should **not** submit the images in the LFW dataset with your solution.

2. **Corner Detectors**: Implement the Harris corner detector and the Förstner corner detector

   (a) Implement the structural tensor $M = \sum w(x,y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}$. Where I_d is the image gradient in the d direction and w is a convolutional kernel. Use a box kernel for simplicity.
   (3 Points)

   (b) Implement the Harris corner detector and display the response function as well as the detected corners for `data/exercise2/building.jpeg`.
   (2 Points)

   (c) Implement the Förstner corner detector and display $w = \frac{det(M)}{Tr(M)}$ after thresholding, $q = \frac{4det(M)}{Tr(M)^2}$ after thresholding, as well as the detected corners for `data/exercise2/building.jpeg`.
   (2 Points)

3. **Keypoint Matching**:

   (a) Read the images `data/exercise3/mountain1.png` and `data/exercise3/mountain2.png`. Use SIFT to extract keypoints and their corresponding descriptors for both images. You can use the SIFT class from the `cv2.xfeatures2d` library for this.
   (2 Points)

(b) Implement the best match ratio test with the threshold 0.4. For finding the best and second best match you may use the *scikit-learn* library. You should implement the matching yourself. Display your matching keypoints using the function `cv2.drawMatchesKnn`.
(4 Points)