Exercise 9 for MA-INF 2201 Computer Vision WS18/19
08.12.2018
Submission on 15.12.2018
Corner Detectors and Image Alignment

1. **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 *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 *building.jpeg.*
   (2 Points)

2. **Keypoint Matching**:

   (a) Read the images `mountain1.png` and `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. In case this library is not found, reinstall python-opencv via

   ```
   conda install -c conda-forge opencv
   ```

   if you use Anaconda. If you use pip, install opencv via

   ```
   pip install opencv-python==3.4.2.16
   pip install opencv-contrib-python==3.4.2.16
   ```

   (2 Points)

   (b) Implement the best match ratio test with the threshold 0.4. To this end, compute the square of the Euclidean distance between all pairs of keypoints accross the two images. For each keypoint, find the most similar and the second most similar keypoint from the other image. Only keep keypoints which are consistent in both ways and pass the ratio test. Display your matching keypoints using the function `cv2.drawMatchesKnn`.
   (3 Points)

3. **RANSAC**: Take the consistent matches obtained from Task 2 and

   (a) Implement `RANSAC` to compute a transformation **T** between `mountain2.png` and `mountain1.png`, where **T** is a projective transformation and can be defined by a set of four pairs of matched keypoints.
   (4 Points)

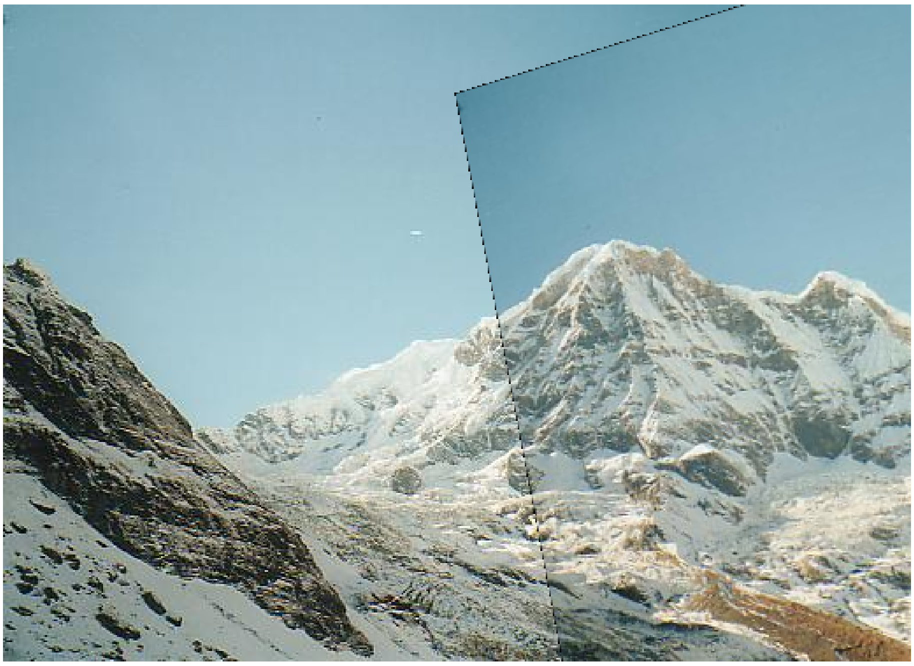

Figure 1: Stitched images.

(b) Transform `mountain2.png` to overlap `mountain1.png`.
(2 Points)

(c) Stitch `mountain1.png` and transformed `mountain2.png` to obtain an image mosaic similar to Figure 1
(2 Points)

Hints: You can use the functions `cv2.getPerspectiveTransform` to calculate a projective transformation and `cv2.warpPerspective` to apply it on an image.