COMP9517 Computer Vision

S1 2018

Assignment 2 Specification

This assignment is worth 15%

The assignment files should be submitted online. **Deadline for submission is week 6 Thursday April 12th, 23:59:59**
Instructions for submission will be posted closer to the deadline.

Preliminaries: The assessment environment

Please ensure that your code compiles and runs on the generic CSE (linux) vlab environment. You can find details about access to vlab remotely here:
https://www.cse.unsw.edu.au/~cs1511/17s2/home-computing/vlab/

The following software is available to you in the testing environment:
Python 2.7.13
gcc (Debian 4.9.2-10+deb8u1) 4.9.2 (supports C++11)

The following libraries are also available:
OpenCV 2.4.9.1
Numpy
(don't assume other libraries are available!)

If you are using the most recent release of OpenCV to develop with, note there are a variety of differences between old and new summarized here:
https://docs.opencv.org/master/db/dfa/tutorial_transition_guide.html

Panoramic stitching is a success story of computer vision. Automatic panorama stitching has been widely adopted in many applications used daily, such as Google Street View, Google Earth and many camera apps in smartphones.

In this assignment, we aim to build a single panoramic image from multiple images. The main steps to achieve the goal include:

1. Feature point detection and description

2. Feature point matching
3. Homography matrix estimation
4. Stitching and generating panoramic images

We describe each of these steps in more detail below.

Exemplar outputs will be demonstrated in the following sections using samples from Lunchroom datasets, two samples are shown below.

img01.jpg img02.jpg

1. Feature point detection and description (3 points)

A panoramic image is built by stitching a series of images of a scene. Usually, the images are likely to be taken from different orientations, and with different resolutions and aspect ratios. Thus, when choosing feature detectors and descriptors, these imaging variables should be taken into consideration. In this step, you should carefully choose an appropriate feature detector and descriptor and display the resulting features.

- Input: a set of images
- Output: image set with all detected features plotted on each image
- Example of ONE image with extracted features displayed

2. Feature points matching (4 points)

In an image series, there may be overlapping areas in each pair of images. Identifying common feature points in the overlapping areas of a pair of images is a key part of image stitching, which may be achieved by feature matching. This step focuses on finding the corresponding feature points in image pairs. Ideally, two images which overlap should contain multiple pairs of matching feature points. Based on these matching points, the relationship between the pair of images may be estimated, such as whether one is rotated and overlapped with the other or scaled up/down with respect to the other.

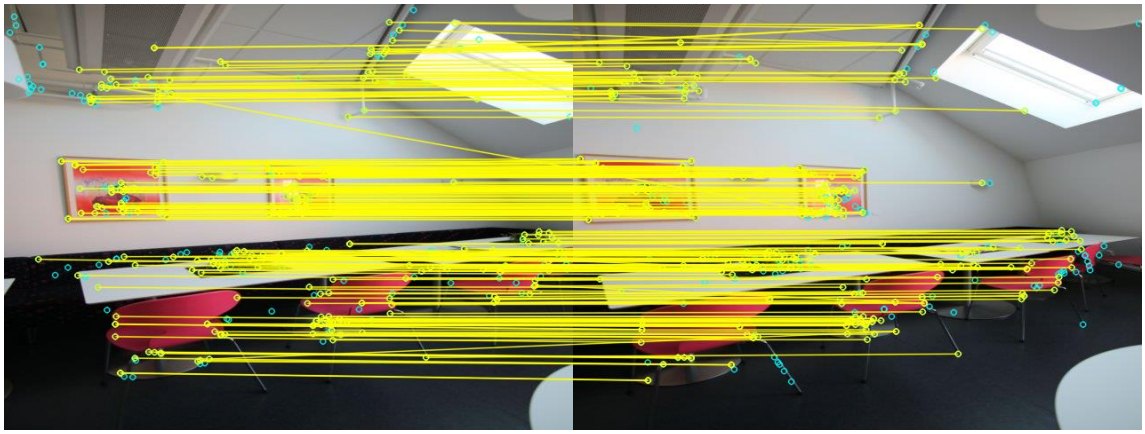
However, an image pair without an overlapping area may also contain pairs of matching feature points. Your algorithm should include a mechanism for rejecting such potential mismatches, such as taking into consideration the number of matching features between a pair of images.

In this step you may not use any of the Descriptor Matcher classes such as cv::BFMatcher, in the OpenCV library. Instead, you should implement a matching algorithm based on k-nearest neighbours along the lines of what is used in cv::BFMatcher.knnmatch**.**

- Input: A set of images
- Output: Matched image pairs with matching results (as shown below). Each matching pair should be saved as a single file with filename '$imagenameA_#featurePointA_$imagenameB_#featurePointB_#matchingPair.jpg', in which,
  - **$imagenameA** is the actual file name
  - **#featurePoint** is the actual number of feature points in each image

- ❖ **#matchingPair** is the number of pairs of matching feature points

- Example of Matching Result:



3. Homography matrix estimation (5 points)

In this step you will align the image pairs with overlap based on the results of feature point matching from the previous step. As noted before, feature point matching is rarely perfect: some features pairs are correct, while some others are completely wrong.

Remember an image homography is a 3x3 affine transformation matrix which maps between two sets of image coordinates. Linear algebra shows that a homography matrix $\boldsymbol{H}$ exists, which describes a transformation between the image pairs $\boldsymbol{I_x}$ and $\boldsymbol{I_y}$ and through which the coordinates of one image will be projected on the other image:

$$I_x = H \times I_y$$

The homography matrix looks like

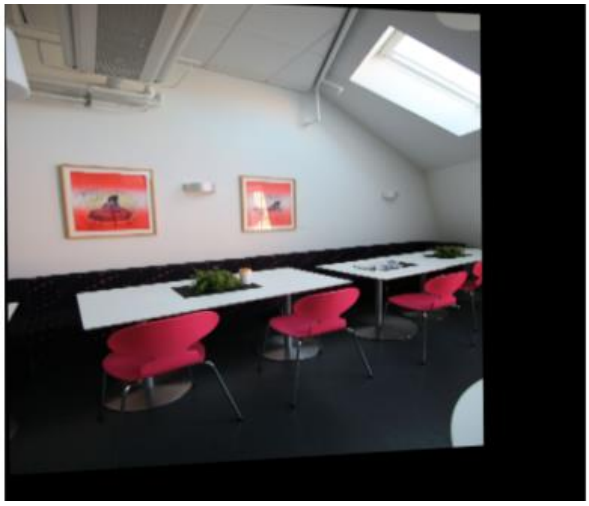
$$\begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix}$$

Normally, the method of least squares is used to minimize the linear projection error between the two images and thus estimate the homography matrix. Do not use bad matching pairs in calculating image homography, as the final result will be poor quality alignment. The RANSAC algorithm covered in the lectures may be helpful for this purpose.

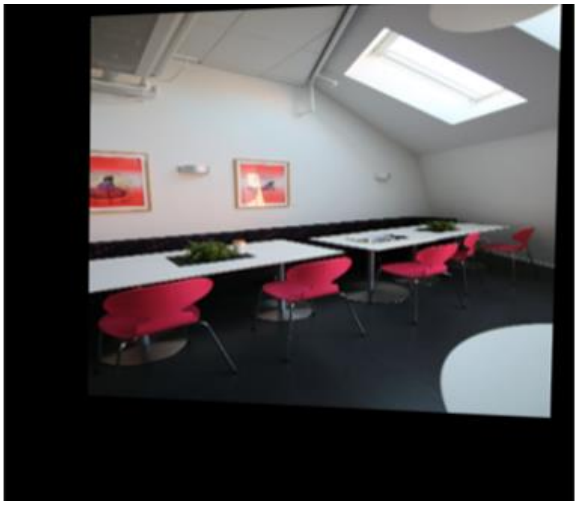
Once we have estimated a homography, i.e. we know how one image will look from the perspective of a reference, we need to transform the input image to the new coordinate system. This transformation is called a warping.

Opencv functions cv::findHomography and cv::warpPerspective are forbidden in this step.

- Input: A set of images
- Output: Set one image in the matched pair as reference image, return the warped version of the other. The output contains two images, each of which is the warped version of the image in the pairs.
- Example:



warped image1 (image2 as reference)

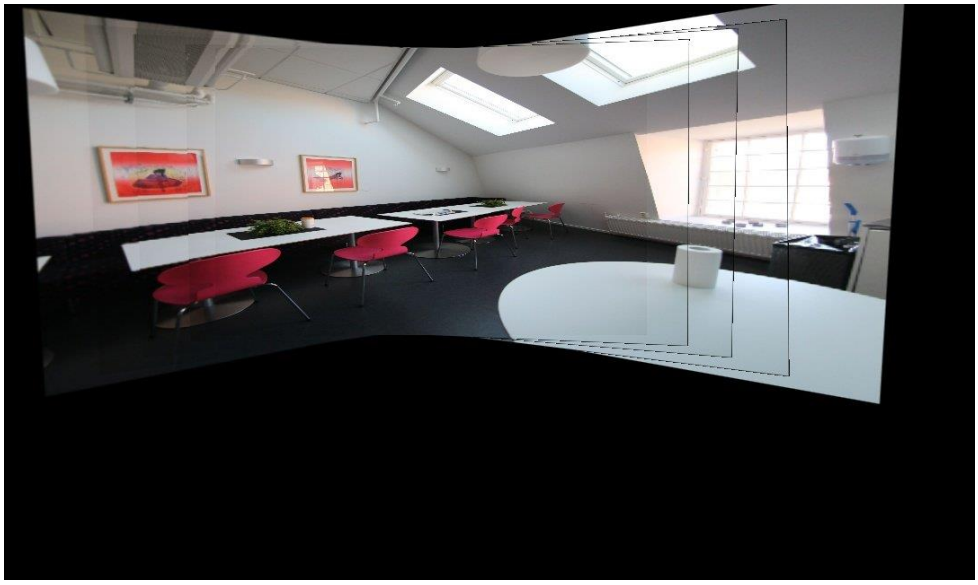


warped image2 (image1 as reference)

4. Combine the image set to generate a panoramic image (3 points)

In the final step, you will use the algorithms implemented in steps 2 and 3 to generate a panoramic image. You will need to find the best matches between every pair of images, find appropriate homographies and warping's, then generate a composite image using the modified image set.

- Input: A set of images
- Output: A panorama image.
- Example



5. Optional Bonus Question (2 points)

Bonus marks will be given for any extensions of this image stitching pipeline. Possible extensions include:

- Border blending and distortion correction
  When capturing the same scene from different orientations, there may be strong differences in lighting due to the fact that exposure times or the white balance of the camera changes. In order to make a better panorama, it may be helpful to smooth the discontinuities between the two overlapping pictures in the resulting panorama.

- Stitching quality assessment
  Evaluating the accuracy of image stitching is tricky and is often done by visual assessment. Write a benchmark test to evaluate the performance of your pipeline on known results. Include references in you code for any evaluation metrics.

You may try other extensions too.

Submission

1. Each step should be run as a single command similar to assignment 1. A skeleton file 'imagestitching.py' has been provided, and you are expected to implement the algorithm using it.
2.

The code should be able to run as a command line as follows:

```
python imagestitching.py num input_folder output_folder
```

where
`num` is an integer in {1,2,3,4} and represents at which step the pipeline terminates
`input_folder` is a folder of input images
`output_folder` is a folder into which either the final panoramic image or the intermediate results will be written.

3. Project architecture should be organized as follows:

```
| code |
   | ImageStitching.py
| input |
   | img1.jpg
   | img2.jpg
   | and so on …
| output |
   | step1|
      | output1.jpg
      | output2.jpg
      | and so on …
   | step2 |
      | output1.jpg
      | output2.jpg
      | and so on …
   | …. and so on|
   | bonus|
```

3. Three datasets have been made available for your use. Make sure to test your algorithms as you develop them.
   - Synthetic
   - Lunch Room Blue
   - Lunch Room

   The datasets may be downloaded from here.

23 March 2018