EECS 442: Computer Vision, Winter 2017

Homework 4: Image stitching

Due date: March 6 by 11:59PM

Your task is to use feature-based homography estimation to stitch together panoramas consisting of multiple images. Write your code following the two parts of the coding instructions. For your report, stick to the REPORT INSTRUCTIONS. Assign one section to each list items in the report instructions. Do not worry about coding instructions when writing your report.

Coding part 1: Stitching pairs of images

The first step is to write code to stitch together a single pair of images. For this part, you will be working with the following pair of images (found in the data folder as uttower_left.jpg and uttower_right.jpg):

1. Load both images, convert to double and to grayscale.

2. Detect feature points in both images. You can use the corner detector code provided (`harris.m`) or the blob detector you have developed as part of the last homework.

3. Extract local neighborhoods around every keypoint in both images, and form descriptors simply by "flattening" the pixel values in each neighborhood to one-dimensional vectors. Experiment with different neighborhood sizes to see which one works the best. If you're using your Laplacian detector, use the detected feature scales to define the neighborhood scales.

4. Compute distances between every descriptor in one image and every descriptor in the other image. You can use `dist2.m` for fast computation of Euclidean distance. Alternatively, experiment with computing normalized correlation, or Euclidean distance after normalizing all descriptors to have zero mean and unit standard deviation. Feel free to experiment with SIFT descriptors. We have provided basic code (`find_sift.m`) for computing SIFT descriptors of circular regions, such as the ones returned by your blob detector.

5. Select putative matches based on the matrix of pairwise descriptor distances obtained above. You can select all pairs whose descriptor distances are below a specified threshold, or select the top few hundred descriptor pairs with the smallest pairwise distances.

6. Run RANSAC to estimate a homography mapping one image onto the other. Report the number of inliers and the average residual for the inliers (squared distance between the point coordinates in one image and the transformed coordinates of the matching point in the other image). Also, display the locations of inlier matches in both images.

7. Warp one image onto the other using the estimated transformation. To do this, you will need to learn about `maketform` and `imtransform` functions.

8. Create a new image big enough to hold the panorama and composite the two images into it. You can composite by simply averaging the pixel values where the two images overlap. Your result might look something like this:

For estimating the transformation it is sufficient just to use grayscale images. You can apply the same compositing step to the color images to get a final result.

Tips and Details

- Refer to the lecture slides here and here.
- For RANSAC, a very simple implementation is sufficient. Use four matches to initialize the homography in each iteration. You should output a single transformation that gets the most inliers in the course of all the iterations. For the various RANSAC parameters (number of iterations, inlier threshold), play around with a few "reasonable" values and pick the ones that work best. For randomly sampling matches, you can use the `randperm` or `randsample` functions.
- In MATLAB, the solution to a nonhomogeneous linear least squares system `AX=B` is given by `X = A\B;`
- Homography fitting calls for homogeneous least squares. The solution to the homogeneous least squares system AX=0 is obtained from the SVD of A by the singular vector corresponding to the smallest singular value: `[U,S,V]=svd(A); X = V(:,end);`

Coding part 2: Stitching multiple images

Now that you have code for pairwise stitching working, the next step is to extend it to work on multiple images. You will be working with the provided data, consisting of three sequences of three images each. For the "pier" sequence, sample output can look as follows (although yours may be different if you choose a different order of transformations):

Source of data and results: Arun Mallya

1. Develop your own algorithm to determine the order of merging the images in each set.
   Hint: One idea is to first run RANSAC between every pair of images to determine the number of inliers to each transformation, and then use this information to determine which pair of images should be merged first, and in the pair, which one should be merged to which.

2. Stitch each set of images in the order determined by your algorithm. Output the stitching results just as what you did in part 1.

Report instructions

1. Pairwise stitching:
   a. Briefly describe your algorithms for feature extraction (if you implemented your own), feature distance computation, putative matching, and RANSAC.
   b. For the `uttower` pair, report the number of homography inliers. An inlier is defined as a pair of matched feature points, of which the Euclidean distance is at most 10 in the stitched image.
   c. Display two images of your stitching result. One is the stitched image without annotations; the other is the stitched image with the locations of the inliers marked. Spot any issues or artifacts in your output. Discuss why.
2. Multiple image stitching:
   a. Briefly describe the algorithm you used to dertermine the merging order.
   b. Report the total number of homography inliers for each set of the images.
   c. Show ALL of your stitched images. For each set of the input images, display two output images. One is your stitching result without annotations, and the other is the stitching result with the locations of the inliers marked. Spot any issues or artifacts in your output. Discuss why.

As usual, submit your code to Canvas, and your report to Gradescope.

Acknowledgements

This homework was developed by Lana Lazebnik at UIUC.