Visual Recognition Assignment 2

Huihuang Zheng, huihuang@utexas.edu

hz4674 Spring 2016

1 Using Code

To run my code, run the **main.m** with Matlab. First, it will call **matchComparsion** and shows figures of output matching lines between 3 images for SIFT for Thresholded nearest neighbors, Thresholded ratio test and Inliers. Second, it will call **detectObject** and outputs figures of object matching results. For every output image, you can see what it's for in the Matlab console. You need to input "dbcont" in Matlab console to see next output image. (as demo in the provided code)

2 Algorithm Implementation

2.1 Threshold Nearest Matching

See **thresMatch.m**. This function inputs two sets of descriptors. For each descriptor d_1 in first set, we calculate the Euclidean distances between all descriptors in second set. Then, we compute the mean distance. Every descriptor d_i in first set is matched to d_j in second set with smallest Euclidean distance. Then we eliminate those smallest distance is less than mean distance times threshold. In this assignment, I set threshold as 0.8.

2.2 Threshold Ratio Matching

See **ratioMatch.m**. Similar to **thresMatch.m**. This function inputs two sets of descriptors. For each descriptor d_1 in first set, we calculate the Euclidean distances between all descriptors in second set. But then we sort these distances. Every descriptor d_i in first set is matched to d_j in second set except whose nearest distance is greater than threshold times second nearest distance. In this assignment, I set threshold as 0.6;

2.3 RANSAC Matching

See **ransac.m**. This function takes location of points in image1 and that of corresponding points in image2 as input. The corresponding points are just nearest Euclidean matching. Then, using the RANSAC algorithm to get best affine function. We eliminate those points who are outliers of RANSAC. There are two problems here. How to get affine transformation from corresponding points and how to do RANSAC.

2.3.1 Implement Affine

Given

$$H = \begin{pmatrix} a & b & c \\ d & e & f \end{pmatrix}, p_1 = \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}, p_2 = \begin{pmatrix} x' \\ y' \end{pmatrix}$$

We know $p_2 = Hp_1$. The H is parameters for affine transformation. So:

$$x' = ax + by + c$$

$$y' = dx + ey + f$$

We have

$$\begin{pmatrix} x & y & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & x & y & 1 \\ \dots & & & & & \end{pmatrix} \begin{pmatrix} a \\ b \\ c \\ d \\ e \\ f \end{pmatrix} = \begin{pmatrix} x' \\ y' \\ \dots \end{pmatrix}$$

So when we have points p_1 and corresponding points $p2$ with location x, y, x', y', we can get the affine transformation.

2.3.2 Implement RANSAC

From above, we know affine has 6 unknowns and each points pair can offer 2 equations. For every iteration, we choose 3 points randomly (at least we should have 3 points) to get an affine transformation. Then, we calculate the location q_i' which is point p_i in image1

affine transforming to image2. Then, for p_i's corresponding point q_i, if distance between q_i and q_i' is less than threshold (I set threshold as $\frac{1}{10}$ of sum of height and width of image2), we said the q_i is inlier. If number of inliers reaches a threshold (I set over 30% points are inliers), I will calculate the average Euclidean distance of all inliers. Finally, I choose the affine transformation with smallest average Euclidean distance of all inliers.

2.4 Object Matching

See **detectObject.m**. For matching algorithm, I combine all three matching above: the matching points are under both constraints of nearest threshold and ratio threshold, then I use RANSAC to get affine transformation and draw rectangle in image2 to show matched object.

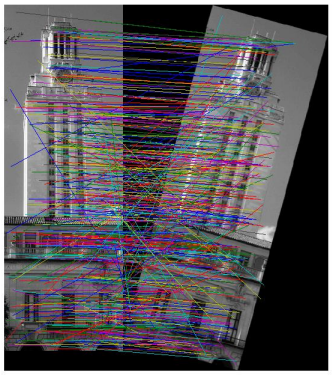
3 Experiment

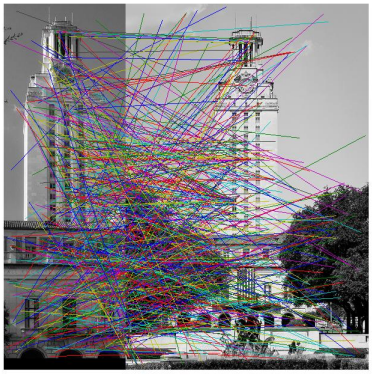
I will show figures of result of my programs and extra examples in this section.

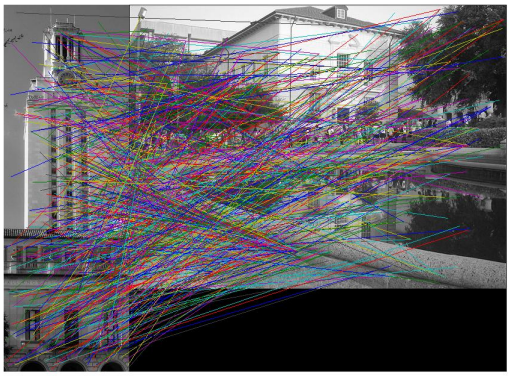
3.1 Show SIFT matching lines between UT Tower

In this section, I will show total 9 images which is the requirement of this assignment and analysis for result at last.

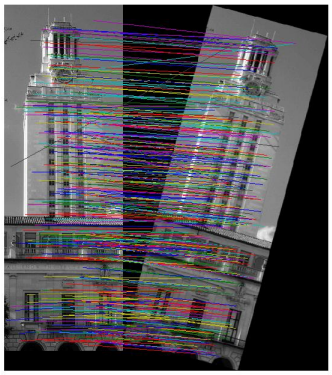
3.1.1 Thresholded Nearest Matching

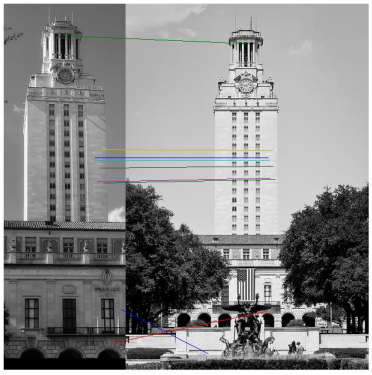


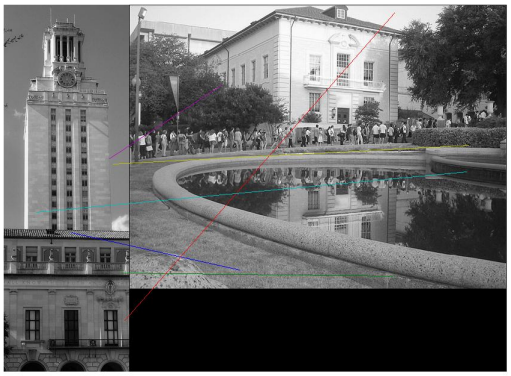




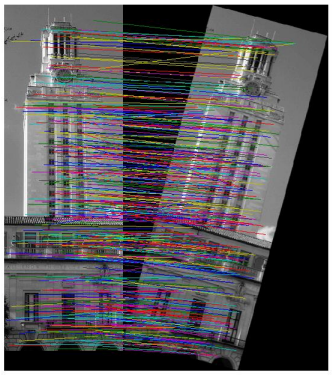
3.1.2 Thresholded Ratio Matching

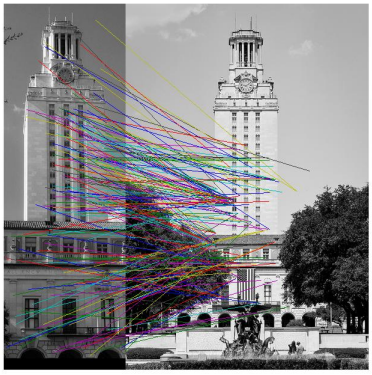






3.1.3 Ransac Matching





3.1.4 Analyze Matching

From those results, we can see that the thresholded nearest matching is the most unstable one, for first two images, a lot of unparallel matching, which suggest it's wrong matching. In addition, it keeps a lot of matching in third image which doesn't have UT Tower at all.

Thresholed ratio matching is better. Fewer several unparallel matching in first two images, eliminate lots of matching pairs in image3.
RANSAC matching does best in image3 and image1, because RANSAC seeks location agreement, it eliminate all matchings in image3. In image1, it only keeps those almost parallel matching, because few unparallel matching was eliminated outliers. However, in image2, it isn't as better as Threshold ratio matching. It should be RANSAC can only find few location agreement and be mislead by raw nearest matching.

So the RANSAC and ratio matching have their different advantages. In object matching, I combine them: using ratio matching to get matching points and using RANSAC to check the spatial agreement.

3.2 Object Matching

3.2.1 UT TOWER

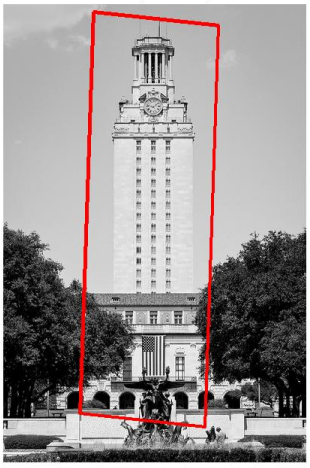
In this section I show my matching results of UT tower image. You can see the program works well. Notice that in image3, we also have a rectangle. When I was hacking, I found there are only 3 pairs of matching survived after thresholded nearest and ratio checking. So these 3 points are major of RANSAC...

The object

Matched Object in The Image

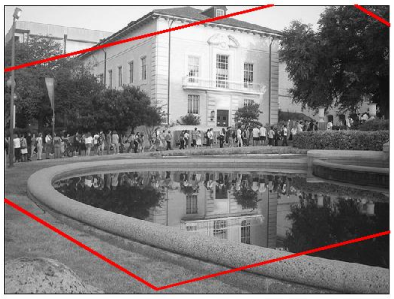
The object

Matched Object in The Image



The object

Matched Object in The Image



3.3 Extra Examples

In this section, I will show two examples of matching, which is extra requirement.

3.3.1 Stronger Variation

This is a example of book in different rotation and illusion of the book on desk.