Computer Vision
Lab Exercise 3
Image Alignment and Stitching

Students should work on this lab exercise in groups of two people. In this assignment you will write a function that takes two images as input and computes the affine transform between them. You are provided three template files, that have to be filled in using the below information:

- $< imageAling.m >$ (aligns two images using the RANSAC algorithm for finding the affine transformation),
- $< ransac\_affine.m >$ (implements the RANSAC algorithm for finding the affine transformation),
- $< mosaic.m >$ (stitches the two images into one).

1 Image Alignment

You will work with the supplied boat images. The overall scheme is as follows:

1. Detect interest points in each image.
2. Describe the local appearance of the regions around interest points.
3. Get the set of possible matches between descriptors in two image.

4. Implement RANSAC to discover the transformation between images.

The first three steps can be performed using David Lowe's SIFT (lab exercise 2): You can use the following functions to obtain the pairs for your RANSAC algorithm.

```
[frames1, desc1]=sift(im1)
[frames2, desc2]=sift(im2)

[matches] = vl_ubcmatch(desc1, desc2)
```

Compare these results with your own implementation from lab 2.
Note: For the final project you will be graded for your own Harris corner point detection and feature matching implementation.

For step 4, RANSAC, should be implemented as follows:

1. Pick P matches at random from the total set of matches T.

```
    perm = randperm(?);
    seed = perm(1:P);
```

How many matches do we need to solve an affine transformation which can be formulated as shown in Fig 1

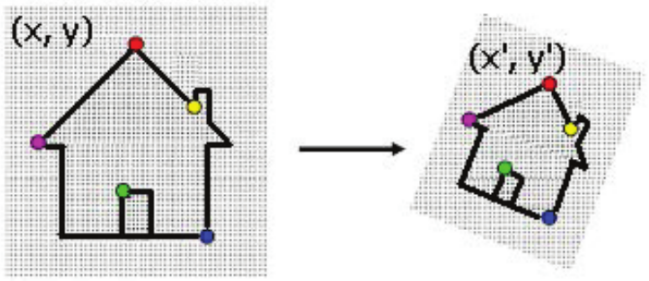

Figure 1: Affine transform

2. Construct a matrix **A** and vector **b** using the P pairs of points to find the affine transformation parameters (Fig.1) $(m_1, m_2, m_3, m_4, t_1, t_2)$ by

solving the equation $\mathbf{Ah} = \mathbf{b}$, where:

$$A = \begin{bmatrix} x & y & 0 & 0 & 1 & 0 \\ 0 & 0 & x & y & 0 & 1 \end{bmatrix}, h = \begin{bmatrix} m1 \\ m2 \\ m3 \\ m4 \\ t1 \\ t2 \end{bmatrix}, b = \begin{bmatrix} x' \\ y' \end{bmatrix} \tag{1}$$

3. Fit the model

   Hint: Equation $\mathbf{Ah} = \mathbf{b}$ can be solved using pseudo-inverse. In Matlab use:

```
h = pinv(A)* b'
```

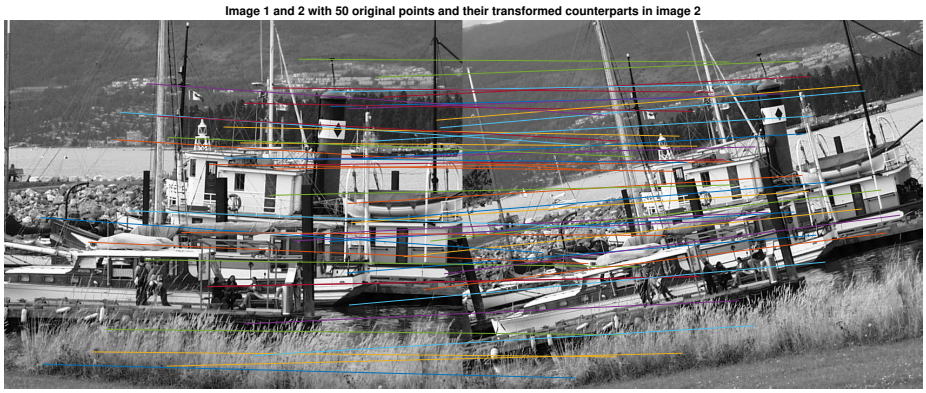

Figure 2: Correspondence in two image

4. Using the affine transform parameters, transform the locations of all T points in image 1. If the transform is correct, they should lie close to their counterparts in image 2. Plot the two images side by side with a line connecting the original T points image 1 and transformed T points over image 2 as shown in Fig 2.

   Recompute the matrix $\mathbf{A}$ over all points, and estimate the new $\mathbf{b}'$.

```
bprim = A*h
```

5. Find inliers: Count the numbers of inliers being defined as the number of transformed points from image 1 that lie within a radius of 10 pixels of their pair in image 2.

```
inliers = find(sqrt(sum((?).^2)) <threshold);
```

6. Refit the model and save the best model (inliers and the parameters) so far and repeat previous steps N times. How do you choose N? Hint : Let q be the probability of choosing an inlier each time a point is sampled, q is given by $q = \frac{\text{number\_inliers}}{\text{sample\_size}}$ when p points are needed for the model estimation, over i iterations. The probability that the algorithm never selects a set of p points of which all are inliers is given by $(1-q^p)^i$. This can be set to an acceptable error threshold (say 0.001).

Once the estimation of $\mathbf{h}$ is finished, transform image 1 using this final set of transformation parameters. If you display this image you should find that the pose of the object in the scene should correspond to its pose in image 2. To transform the image, use the build-in Matlab functions as follows

```
affine_transform = [h(1) h(2) h(5);...
                    h(3) h(4) h(6);...
                     0    0    1   ];
tform = maketform('affine', affine_transform');
image1_transformed = imtransform(image1, tform, ...
    'bicubic');
```

Do the same procedure for the second image, but then with the inverse transformation, as:

```
tform = maketform('affine', inv(affine_transform)' );
image2_transformed = imtransform(image2, tform, ...
    'bicubic');
```

The final images that you should plot for this exercise are depicted in Fig 3.

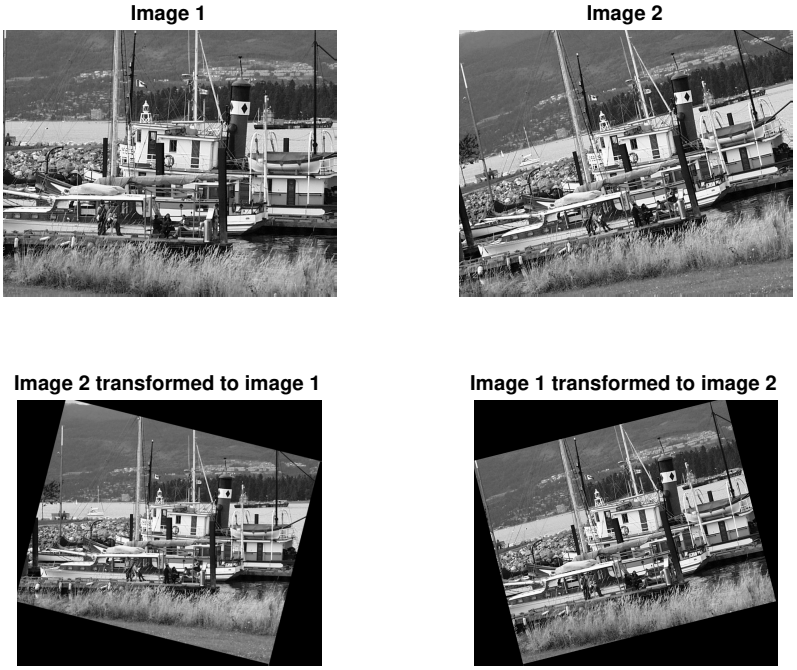

Figure 3: Transformed images

2 Image Stitching

In this assignment you will write a function that takes two images as input and stitch them together. You will work with supplied image *left.jpg* and *right.jpg*. The overall scheme can be summarized as follows:

1. Find the best transformation between input images, i.e., by taking the first image as original (no transform) and estimating the affine

transformation of the second image with respect to the first image. Use the image alignment function that you developed in the previous task.

2. Note that images do not have the same size. You need to think about the size of your final panoramic image. How do we determine that? Hint: calculate the transformed coordinate of corners of the *right.jpg*

3. Finally, combine the *left.jpg* with the transformed *right.jpg* in one image. The final output should look like Fig 4. Hint: Use *nanmean* function in Matlab if you need to compute the mean of a matrix with "NaN" values in it.

Figure 4: Stitched image