CS5670 Computer Vision -- Assignment 2

Lilin Wang (lw555@cornell.edu)
Xiaoyang Ma (xm74@cornell.edu)

Task 1: Color quantization with k-means

1. Assuming your goal is to find interest points that are consistent under changes in the image, or noise, which of the operators will you choose?
   I choose DoG (Difference of Gaussians). Because gradient is very sensitive to noise. In order to deal with noisy images, it's better to use DoG.

2. Use SIFT to find interest points and descriptors to all the provided input images.
   getDescriptorKP(*img, file_name*)
   img: input image file name
   file_name: output image file name

Input Image: StopSign1.jpg

Input Image: StopSign2.jpg

Input Image: StopSign3.jpg

Input Image: StopSign4.jpg

3. Write a function "findMatches" that gets as input two sets of descriptors and their locations in the image (one set per image).
   findMatches(*kp1, des1, kp2, des2*)
   kp1: interest points for image1
   des1: descriptors for image1
   kp2: interest points for image2
   des2: descriptors for image2

4. Write a function "showMatches" that gets as input two images, and the list (or array) of matches from the previous item. The function will generate an image where the matches are displayed. See an example bellow of a good visualization – it connects matching pixels by highlighted lines. Please generate these images for the following image pairs: StopSign1-StopSign2, StopSign1-StopSign3, StopSign1-StopSign4
   showMatches(*img1, img2, matches, file_name*)
   img1, img2: input image name
   matches: matched coordinates of img1 and img2
   file_name: prefix for output image name

StopSign1- StopSign2

StopSign1- StopSign3

StopSign1- StopSign4

5. Write a function “affineMatches” that takes a list of matches between two images, uses RANSAC to find an affine transformation between the pair of images and rejects the outlier matches. You can implement RANSAC yourself, or download code for it. The function should return the computed affine transformation and the inlier matches. Generate again the visualizations of the image matches, this time using only inliers.

affineMatches(*matches, is_homograph*=0)
matches: matched coordinates of img1 and img2
is_homograph: generating the affine transformation when is_homograph=0, while generating homograph transformation when is_homograph=1

Did RANSAC do what you expected it to do?
Yes. I expect it to generate more geometrically accurate matched coordinates, and it does.

How many matches did you have before and after RANSAC?

Input Image: StopSign1- StopSign2.jpg Before RANSAC Number of Matches: 27

Input Image: StopSign1-StopSign2.jpg
After RANSAC
Number of Matches: 18

Input Image: StopSign1-StopSign3.jpg
Before RANSAC
Number of Matches: 30

Input Image: StopSign1-StopSign3.jpg
After RANSAC
Number of Matches: 21

Input Image: StopSign1-StopSign4.jpg
Before RANSAC
Number of Matches: 36

Input Image: StopSign1-StopSign4.jpg
After RANSAC
Number of Matches: 19

Are all the remaining matches consistent geometrically?
No. But most of matches consistent geometrically. Because there is an error tolerance at RANSAC, it iterates several times until the error is small enough. So theoretically speaking, not all the remaining matches necessarily consistent geometrically.

6. You will write a function "alignImages" that gets as input a pair of images and the affine transformation between them. The function will apply the transformation to one of the images (warp) and will "merge" the warped image with the other image. Merging will be done as follows: Take the red and green channels from one image and the blue channel from the other image. Use them to generate a new image.

alignImages(*img1*, *img2*, *transformation*, *file_name*)
img1, img2: input two images
transformation: transformation matrix
file_name: prefix for output image

Input Image: StopSign1- StopSign2.jpg

Input Image: StopSign1- StopSign3.jpg

Input Image: StopSign1- StopSign4.jpg

How can you tell if the transformation you got is accurate?
Directly, we can tell the accuracy from the merged image. If the two input images overlap comparatively well in the merged image, the transformation we got is comparatively accurate.

Did you get satisfactory results?
Yes. Although the two input images do not perfectly overlap, we get a reasonable result after we adjust the

parameters in RANSAC, such as maximum error, minimum good matches number and iteration number.

When does it succeed and when does it fail?
When there is too much noise in the image, such as the StopSign2.jpg, it might fail. From StopSign2.jpg, we can see that most interest points locate in the background rather than the stop sign. Those noises in the background make it hard to find right matches.
When there is few noise in the image and the object in the image is clear, such as the StopSign4.jpg , the main background is clean sky, it's more likely to succeed.

7. Come up with a quantitative measure to assess the success of the alignment. The measure you propose and its results should make sense and explain well your results. Discuss the quantitative results you got.

measureAlignment(*matches, transformation*)
matches: matched coordinates of input images
transformation: transformation matrix

I calculate the average of distance of every matched pair of coordinates. Distance = abs(2.0 - calculated.x/ original.x – calculated.y / original.y), where the (calculated.x, calculated.y) is the coordinate calculated from transformation matrix and image1's matched coordinates, and the (original.x, original.y) is image2's matched coordinate.

8. Repeat Items 5 and 6, this time for a homography. Compare the quantitative results of with affine transformations and homographies. Can you explain the results?

Haffine_from_points(*ft, tp*) is to calculate the affine transformation of two matrixs.
H_from_points(*ft, tp*) is to calculate the homography transformation of two matrixs.

Input Image: StopSign1-StopSign2.jpg Affine transformation	Input Image: StopSign1-StopSign2.jpg homography
Transformation matrix is: [[9.3581361e-01 5.8176318e-02 3.2122550e+02] [8.8695524e-02 1.1934018e+00 6.6344095e+01] [0.0000000e+00 0.0000000e+00 1.0000000e+00]] The number of inlier matches is 18 Average distance of alignments is: 0.0113743845637	Transformation matrix is: [[7.3408190e-01 1.1137584e-01 3.3774098e+02] [-5.2581908e-02 1.0848166e+00 9.4564019e+01] [-4.1182149e-04 4.1801084e-04 1.0000000e+00]] The number of inlier matches is 17 Average distance of alignments is: 0.0110259982399
	

Input Image: StopSign1-StopSign3.jpg Affine transformation	Input Image: StopSign1-StopSign3.jpg homography
Transformation matrix is: [[1.4629606e+00 -5.6520190e-02 1.2143512e+03] [7.9009322e-02 1.58532029e+00 1.0026557e+03] [0.0000000e+00 0.0000000e+00 1.0000000e+00]] The number of inlier matches is 21 Average distance of alignments is: 0.00262826986492	Transformation matrix is: [[2.0231657e+00 -1.6788835e-01 1.1711935e+03] [5.2559696e-01 1.4657265e+00 9.7403199e+02] [6.6555199e-05 -4.1684920e-05 1.0000000e+00]] The number of inlier matches is 21 Average distance of alignments is: 0.00262826986492
	
Input Image: StopSign1- StopSign4.jpg Affine transformation	Input Image: StopSign1-StopSign4.jpg homography
Transformation matrix is: [[2.5855301e+00 -9.4154134e-02 1.4354716e+02] [2.3583272e-01 2.6547298e+00 6.2781008e+01] [0.0000000e+00 0.0000000e+00 1.0000000e+00]] The number of inlier matches is 19 Average distance of alignments is: 0.00789528191382	Transformation matrix is: [[2.7383899e+00 -7.4699720e-03 1.45914912e+02] [3.7402192e-01 2.4856815e+00 8.8438825e+01] [-1.2743759e-04 6.3564243e-04 1.0000000e+00]] The number of inlier matches is 15 Average distance of alignments is: 0.0073686293909
	

From above results, we can tell that the results from homography may not be more accurate than the results from affine transformation. Although there are 9 valid parameters in homography matrix, while there are only 6 valid parameters in affine transformation, homography may not be more accurate.

An affine transformation is a special type of general homography. Affine transformation is a linear transformation, while homography may be a non-linear transformation. Homography also includes image distortion and change of distance scale.

So in some cases, homography can lead to overfitting, while affine transformation can be more stable. That's why the result of homography is not more accurate than affine transformation sometimes.

Task 2: Epipolar Geometry and Camera Calibration

Part1:

1. Assume that the left camera lies at the origin of the world coordinate system:

1) The right camera coordinate is rotated by 60° along y-axis based on the world coordinate. The rotation matrix R is computed as follow:

$$\begin{pmatrix} cos(-60°) & 0 & sin(-60°) \\ 0 & 1 & 0 \\ -sin(-60°) & 0 & cos(-60°) \end{pmatrix} = \begin{pmatrix} \frac{1}{2} & 0 & -\frac{\sqrt{3}}{2} \\ 0 & 1 & 0 \\ \frac{\sqrt{3}}{2} & 0 & \frac{1}{2} \end{pmatrix}$$

2) The coordinate of the right camera in the world coordinate system is $\tilde{C} = \left(-\frac{\sqrt{3}}{2}d, 0, \frac{d}{2}\right)^T$. The translation vector t is computed with the equation $t = -R\tilde{C}$ which the result is $t = \left(\frac{\sqrt{3}}{2}d, 0, \frac{d}{2}\right)^T$. Therefore the matrix T is:

$$\begin{pmatrix} 0 & -\frac{d}{2} & 0 \\ \frac{d}{2} & 0 & -\frac{\sqrt{3}}{2}d \\ 0 & \frac{\sqrt{3}}{2}d & 0 \end{pmatrix}$$

3) As each camera has a focal length of unity, K_L and K_R are both identity matrices.

4) The fundamental matrix is:

$$F = {K_R}^{-T} T_X R {K_L}^{-1} = \begin{pmatrix} 0 & -\frac{d}{2} & 0 \\ -\frac{d}{2} & 0 & -\frac{\sqrt{3}}{2}d \\ 0 & \frac{\sqrt{3}}{2}d & 0 \end{pmatrix}$$

2.

The corresponding epipolar line in the right image is:

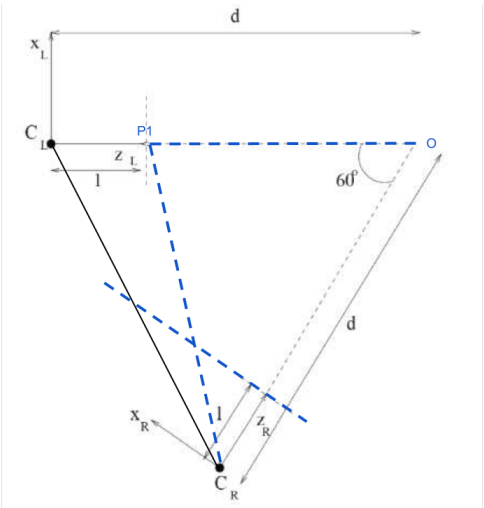
$$l_R = Fx = \begin{pmatrix} 0 & -\frac{d}{2} & 0 \\ -\frac{d}{2} & 0 & -\frac{\sqrt{3}}{2}d \\ 0 & \frac{\sqrt{3}}{2}d & 0 \end{pmatrix} \begin{pmatrix} 1 \\ 1 \\ 1 \end{pmatrix} = \begin{pmatrix} -\frac{d}{2} \\ -\frac{1+\sqrt{3}}{2}d \\ \frac{\sqrt{3}}{2}d \end{pmatrix}$$

The epipolar of the left camera corresponding to the right image plan is $(\sqrt{3}, 0)$. We can also know the normal vector of the corresponding epipolar line from l_R is $(-\frac{d}{2}, -\frac{1+\sqrt{3}}{2}d)$. Therefore the epipolar line is:

$$-\frac{d}{2}(x - \sqrt{3}) - \frac{1+\sqrt{3}}{2}dy = 0$$

3.

The correspondences to the left image point (x, y) = (0, 0) is the radial P_1O showed in the following figure.



The points in radial P_1O correspond to the right image are what we want. We can use $p = [R|r]P1$ to calculate the corresponding points and we have already known matrix R in question1.

The corresponding points of P_1 and O on the right image are $(\frac{\sqrt{3}(d-1)}{d+1}, 0)^T$ and $(0,0)^T$. Therefore the corresponding radial is:

$$\begin{cases} x \leq \dfrac{\sqrt{3}(d-1)}{d+1} \\ y = 0 \end{cases}$$

4.

As the manipulation is done on the left camera, let's set the right camera lies at the origin of the world coordinate system. Still use R and T_x to be the rotation and translation matrix of the left camera. Suppose x_1 and x_2 are the points that respectively on the left image plane and the right image plane projected by a sample point X. We can know that $Fx_1$and $F^T x_2$ are the epipolar lines. We want these two lines to be parallel. Since each image points satisfies ${x_2}^T F x_1 = 0$, we have ${x_2}^T F^T x_2 = 0, \forall x_2$, which shows that F is skew-symmetric. Since $F = T_X R$, where T_X is skew-symmetric and R is orthogonal, we can derive that R and T_x should satisfy the following:

$$(T_x R)^T = -T_x R$$

Part2:

For this part of the assignment, the goal was to compute the 3 by 4 projection matrix P of a pinhole camera given 10 world points and their corresponding image projections. Once P is computed, it is decomposed to its intrinsic and extrinsic parameters. The input files used were the *HW2_world.txt* file, which contain the (X, Y, Z) values of each work points, and the *HW2_image.txt* file, which contains the (x, y) image points of each corresponding world point.

1.

Firstly, since P is a homogenous matrix, the world points and the image points need to be converted to homogenous points by adding a row of 1 for both sets of data. This makes the image points 3d coordinates and the world points 4d coordinates. The computed P matrix can be seen below.

$$P = \begin{bmatrix} -0.127000127 & -0.254000254 & -0.381000381 & -0.508000508 \\ -0.508000508 & -0.381000381 & -0.254000254 & -0.127000127 \\ -0.127000127 & 5.41233725e-16 & -0.127000127 & -6.66133815e-16 \end{bmatrix}$$

The re-projection of the points is very close to the original image. We computed the difference of the original points and the projected points using $\Delta d = \sqrt{\Delta x^2 + \Delta y^2}$ and the output is as follows:

Δd*:*
6.466036496704424e-15, 7.549516567451064e-15, 4.528839093602940 6e-15, 5.17892563931115e-15, 4.528839093602940 6e-15, 1.1374233532693354e-14, 1.3322676295501878e-14, 7.160723346098895e-15, 1.6968624882578708e-14, 6.466036496704424e-15

2.
Since $PC = 0$, C lies in the null space of P. Therefore C can be computed by getting the SVD of P and picking the vector corresponding to that null space. The C parameters can be seen below.