PROJECT 4 - Due Midnight April 21, 2016

This is by far the most complex project you have encountered. The assignment of points allows you to do section1 and pass, but section 2 is required for a good grade. Before anything else, you need to pick a partner, and work this project in pairs. Please send me, by email, the names of the two members of your group. **There will be no groups of one!** If you cannot find someone to team with, let me know and I will either find you a partner or create of group of 3.

I am not explicity requiring an "extra" section, however, if you add something that you think is particularly clever, add it and get points!

This is a project to find and use the *homography* which maps between points in two images. The coordinate reprsentation used is *homogeneous* coordinates. That is, let a point in image 1 be $\begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$, and suppose the corresponding point in image 2 is $\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix}$. Then the 3×3 matrix H will have the property that $\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$.

From $\begin{bmatrix} wx' \\ wy' \\ w \end{bmatrix}$, you can find the point on the focal plane by dividing by the third coordinate, w.

Since these are homogeneous representations of vectors, they represent projective operations.

1 Given manually determined correspondences, find H - 65 points

In this section, you will learn the basics of homographies, write and underlying code, and test it on real image, but using the human brain to solve the correspondence problem.

1. Take a camera out on the area between the EB buildings and take a picture of all or part of some building. Remember that the algorithm we are using is only valid for planar surfaces. Call that picture P. Standing in the same place, rotate through some moderate angle, like thirty degrees, and take another picture of the same wall. The amount of rotation doesn't really matter, but it is important that the two pictures have significant overlap, like 40%. Call the second picture P'.

2. Find as many correspondences as you can (the more you find, the better it will work... at least ten) between the two pictures. Obviously these must be in the overlapped area. Do this manually, That is, find a distinctive point in P (like the lower right

corner of a particular window). Find the same point in P'. Write down the two pairs of coordinates beside each other. The ten correspondences should be spread out over the overlap area as much as possible. Try to avoid having 3 in a perfect straight line.

3. Make a list of correspondences, of the form

x1	y1	x1'	y1'
x2	y2	x2'	y2'

   .
   .
   .

4. For each correspondence, $x_i, y_i \Longleftrightarrow x_i', y_i'$ construct this 2 x 8 matrix .

$$\begin{bmatrix} 0 & 0 & 0 & -x_i & -y_i & -1 & x_i y_i' & y_i y_i' \\ x_i & y_i & 1 & 0 & 0 & 0 & -x_i x_i' & -y_i x_i' \end{bmatrix}$$

5. Supposing that you found n correspondences, stack the 2x8 matrices to form a 2n x 8 matrix. Call that matrix A.

6. For each correspondence, construct the 2×1 vector

$$\begin{bmatrix} -y_i' \\ x_i' \end{bmatrix}$$

   Stack these vectors to product a $2n \times 1$ vector named **D**.

7. Construct the vector **h** using

$$\mathbf{h} = (A^T A)^{-1} A^T D$$

   I rewrite the previous equation explicity showing matrix/vector sizes:

$$\mathbf{h}_{8\times1} = (A^T_{8\times2n} A_{2n\times8})^{-1}_{8\times8} A^T_{8\times2n} D_{2n\times1}$$

   Note that this requires inversion of an 8 x 8 matrix.

8. You now have an 8 element vector

$$\mathbf{h} = \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \end{bmatrix},$$

Contstruct the 3×3 matrix

$$H = \begin{bmatrix} h_1 & h_2 & h_3 \\ h_4 & h_5 & h_6 \\ h_7 & h_8 & 1.0 \end{bmatrix}$$

9. The matrix H satisfies $\mathbf{x}' = H\mathbf{x}$. Thus, for any point $\mathbf{x}$ in P, $H\mathbf{x}$ will be the corresponding point in P1.

You may do these operations in Matlab, however, since in the next section, you will be writing code to use these operations, you may find it convenient to use the ifs matrix manipulation functions, (dmatrix(), ifsmatmult(), ifsinverse, dvector()).

Once you have H, you can copy an image into the coordinates of another image. So for every point in P, find its coordinates in P1 and copy the pixel from P to P1. Note that you will need to make P1 bigger since many of the points of P will fall outside of the original P1.

Note: you are very likely have the same problem with black dots here that you had in a previous homework. If this is a problem, instead of

copy from P to P1 using H (scan over the input image, figure out where it goes in the output)

do

copy from P1 to P using Hinverse (scan over the output imge, figure out where it came from in the input)

If your two image fit together reasonably well, congratuations!

2 Find the Correspondences Automatically, 35 points

1. Implement an interest point detector. You can use Harris-Laplace, SIFT, or you can invent one. Make sure it works reasonably well.
2. Using your interest detector to define the center of a neighborhood, construct the SKS or SIFT (or other) local area descriptor for the vicinity of each interest point.
3. Compare each local area descriptor with each other local area descriptor, unless you can figure out a clever way to reduce the complexity of this search. The result of the comparisons should be a list of correspondences.
4. Use the list of correspondences to compute an H matrix just like you did in the previous section.
5. Use the H matrix to stitch the images together as before.

The write up should be submitted through moodle.