Problem-set-2

In this practicing, you will implement an image stitcher that uses image warping and homographies to automatically create an image mosaic. We will focus on the case where we have two input images that should form the mosaic, where we warp one image into the plane of the second image and display the combined views. This problem will give some practice manipulating homogeneous coordinates, computing homography matrices, and performing image warps.

** There are some built-in Matlab functions that could do a lot of the work for this project. However, to get practice with the workings of the algorithms, we want you to write your own code. For example, you may not use any of these functions in your implementation: cp2tform, imtransform, tformarray, tformfwd, tforminv, imwarp, maketform. **

Here are the main components you will need to implement:

1. **Getting correspondences [10 points]**: First write code to get manually identified corresponding points from two views. Look at Matlab's `ginput` function for an easy way to collect mouse click positions. The results will be sensitive to the accuracy of the corresponding points; when providing clicks, choose distinctive points in the image that appear in both views.

2. **Computing the homography parameters [30 points]**: write a function that takes a set of corresponding image points and computes the associated 3 x 3 homography matrix H. This matrix transforms any point p_i in one view to its corresponding homogeneous coordinates in the second view, p'_i, such that $\lambda p_i = H p_i'$. Note that p_i and p_i' are both 3-vectors. The function should take a list of $n \geq 4$ pairs of corresponding points from the two views, where each point is specified with its 2d *image* coordinates.

   We can set up a solution using a system of linear equations $Ax = b$, where the 8 unknowns of H are stacked into an 8-vector x, the $2n$-vector b contains image points from one view, and the $2n \times 8$ matrix A is filled appropriately so that the full system gives us $\lambda p_i = H p_i'$. Let $H_{3\times 3} = 1$. Solve for the unknown homography matrix parameters.

   Verify that the homography matrix your function computes is correct by mapping the clicked image points from one view to the other, and displaying them on top of each respective image (`imshow`, followed by `hold on` and `plot`). Be sure to handle homogenous and non-homogenous coordinates correctly.

3. **Warping between image planes [30 points]:** write a function that can take the recovered homography matrix and an image, and return a new image that is the warp of the input image using H. Since the transformed coordinates will typically be sub-pixel values, you will need to sample the pixel values from nearby pixels. For color images, warp each RGB channel separately and then stack together to form the output.

   To avoid holes in the output, use an inverse warp. Warp the points from the source image into the reference frame of the destination, and compute the bounding box in that new reference frame. Then sample all points in that destination bounding box from the proper coordinates in the source image. Note that transforming all the points will generate an image of a different shape / dimensions than the original input.

   [Useful Matlab functions: `round`, `interp2`, `meshgrid`, `isnan`].

As an example for the syntax of `interp2`, consider this form: `ZI = INTERP2(Z,XI,YI).` If we call `output = interp2(double(input), x, y);`

where `x` and `y` together specify a single position we want to inverse warp from, then we get the interpolated color for that single position sampled from `input`. If we call `output(:,:,i) = interp2(double(input(:,:,i)), X, Y);`

where `X` and `Y` are 2D *matrices* of coordinates (`X` listing the x positions, `Y` listing the y positions), then we get a corresponding matrix of interpolated colors for *all* those positions at once.

4. **Create the output mosaic: [5 points]** once we have the source image warped into the destination image's frame of reference, we can create a merged image showing the mosaic. Create a new image large enough to hold both (registered) views; overlay one view onto the other, simply leaving it black wherever no data is available. Don't worry about artifacts that result at the boundaries.

After writing and debugging your system:

5. **[5 pts]** Apply your system to the provided pair of images (uttower1,2), and display the output mosaic.

6. **[5 pts]** Show one additional example of a mosaic you create using images that you have taken.

7. **[15 pts]** Warp one image into a "frame" region in the second image. To do this, let the points from the one view be the corners of the image you want to insert in the frame, and the let the corresponding points in the second view be the clicked points of the frame (rectangle) into which the first image should be warped. Use this idea to replace one surface in an image with an image of something else. For example -- overwrite a billboard with a picture of your dog, or project a drawing from one image onto the street in another image, or replace a portrait on the wall with someone else's face, or paste a Powerpoint slide onto a movie screen, ...