Activity 11: Image Registration

Primary Goal: Learn how to perform feature-based image registration using control points.
Secondary Goal: Gain familiarity with image transformations.

*First download the **Activity11.zip** folder from our course website and unzip it.*

1.) Affine Transformations

Load a color photograph of yourself (or a favorite portrait) into an image matrix A. We are going to practice image transformations.

a.) Scaling

To make an image twice as large, we would double all x and y coordinates. Using matrix notation, we would multiply the homogeneous image coordinates (x,y,1) by the 3x3 matrix M shown below.

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} 2 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

We can type the 3x3 matrix M into Matlab as:

```
M = [2 0 0; 0 2 0; 0 0 1]
```

We can create an image transformation T from this matrix using the Matlab command maketform:

```
T = maketform ('affine', M);
```

We can apply this transformation T to an image A using the (now obsolete) function `imtransform`:

```
A_Transformed = imtransform(A,T);
```

View the image A_Transformed that results. If you would like to view the transformed image on top of the original image, the Matlab command `imshowpair` displays overlapping images.

```
imshowpair(A, A_Transformed);
```

Predict what the matrix M will do to the image.

$$M = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 0.5 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

b.) Rotation

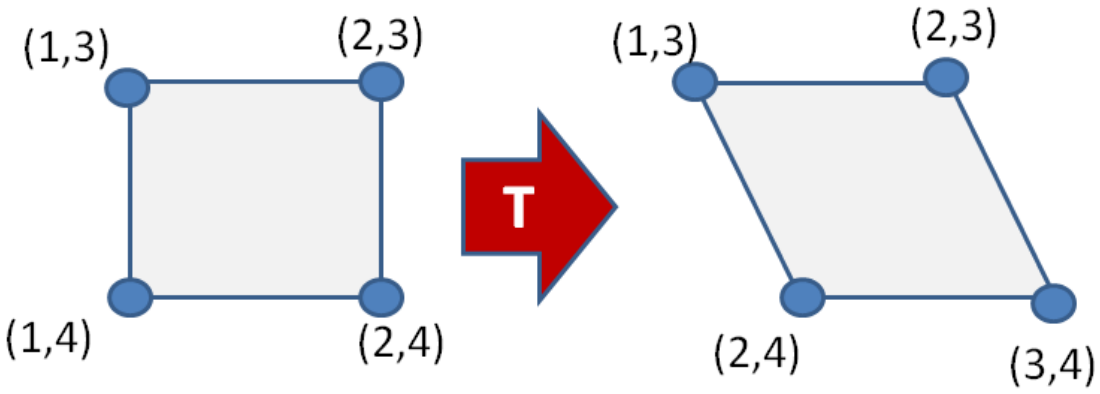
To rotate the (x,y) coordinates by an angle θ, we would use the 3x3 matrix M shown below.

$$\begin{bmatrix} x_2 \\ y_2 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

Create a rotation matrix M with angle $\theta = \pi/4$. Then create the transformation T and apply it to the original image A. View the overlapping results using the `imshowpair` command.

c.) Shear

Suppose we want to create a shear effect that maps a square to a slanted parallelogram. Note the y-coordinates are inverted, as is typical in images.



We can type the list of points for the square as:

```
P1 = [1,3; 2,3; 1,4; 2,4]
```

The corresponding points on the rhombus would be:

```
P2 = [1, 3; 2,3; 2,4; 3,4]
```

The Matlab command `cp2tform` *(Control Points To Transform)* creates the transformation that maps one set of points to another. Let's try this on our point sets. Note this is still an affine transformation, since parallel lines remain parallel.

```
T = cp2tform (P1, P2, 'affine');
```

2.) Image Stitching

The art of joining overlapping into one large panoramic image is called image stitching or panography. The two images in the Activity11.zip folder are photos of a common scene with a small amount of overlap. Let's load them into two images.

```
Left = imread('left.jpg');
Right = imread('right.jpg');
```

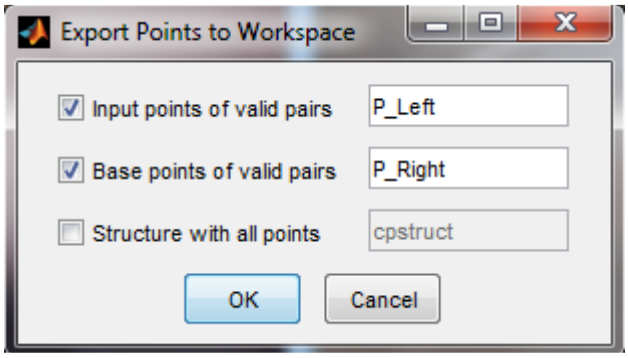
View these images. Note that Left shows the left half of a scene and Right shows the right half, with a small overlapping region common to both images. We want to find a transformation that maps Left to Right so we can stitch these images to make one large panoramic image.

First we need to identify common feature points in the two images. Matlab has a GUI called the **Control Point Selection Tool** which allows a user to click on corresponding points in two images.

```
cpselect (Left, Right);
```

Identify common points in the two images, such as corners or other well-defined features. Click on a point in the top Left zoomed image and the corresponding point in the top Right zoomed image. You can move the zoomed portion of the image by adjusting the window in the bottom pane. Select at least 4 pairs of points (8 mouse clicks total). Notice that Matlab labels the points as you go. Clicking on a previously clicked point allows you to move that point.

When you are done clicking points, save the points to the workspace by clicking the File menu and selecting "Export Points to Workspace". Give your points a name you will remember, such as P_Left and P_right. Note each of these variables is a Nx2 matrix of (x,y) points, where N is the number of pairs you selected.



The code below will display the points you clicked.

```
subplot(121); imagesc(Left); hold on;
plot(P_Left(:,1), P_Left(:,2), 'ro'); hold off;
subplot(122); imagesc(Right); hold on;
plot(P_Right(:,1), P_Right(:,2), 'ro'); hold off;
```

Next, we want to compute the affine transformation T that maps the points P_Left to the points P_Right. The Matlab command `cp2tform` maps one set of control points to another.

```
T = cp2tform (P_Left, P_Right, 'affine');
```

To view the result of applying a transformation to your Left image, you could type:

```
Left_Transformed = imtransform (Left, T);
imshow(Left_Transformed);
```

Unfortunately, this transformed image does not line up with the current image Right since they are not on a common coordinate system. You can verify this by displaying the overlapping images using imshowpair:

```
imshowpair (Left_Transformed, Right);
```

The tricky part is to place the transformed image and the original Right image on the same coordinate axes. The Matlab function `align` in the Activity11.zip folder does just that.

```
[Left_Transformed, Right_Transformed] = align (Left, Right, T);
```

View the transformed images together using `imshowpair`.

To make one image, we could average the images:

```
I = (Left_Transformed + Right_Transformed) / 2;
```

View this image. Note the parts where the images do not overlap get brighter since we are averaging with white (255,255,255) pixels.

Rather than averaging the images, we might also try taking the minimum value:

```
I = min (Left_Transformed, Right_Transformed);
```

This way, we will pick a color that is darker than the default white (255,255,255) background. View this image.

Try to think of how you might make a giant panoramic image from 3 or more images.