Load up the images and produce point matches. Use the zoom feature to produce better matches. Type help collect_point_matches for more info.

```
im1 = imread('shed1.jpg');
im2 = imread('shed2.jpg');
[x1, x2, tri_pts] = collect_point_matches(im1, im2)
```

Now make x1 & x2 into homogeneous vectors by adding columns of ones.

1) Implementing the Normalized 8-Point Algorithm

Now we have a set of point correspondences we are ready to calculate the fundamental matrix. As mentioned in Forsyth & Ponce (pg. 220), computing the fundamental matrix directly behaves rather poorly. Hartley showed this was primarily due to poor scaling, and rescaling improves things substantially. Rescaling involves:

Normalize each set of image points (left and right separately) so their mean is zero and their root mean squared distance from the origin is sqrt(2).

$$T_{norm} = \begin{bmatrix} \frac{\sqrt{2}}{R} & 0 & -\frac{\sqrt{2}}{R}\mu_x \\ 0 & \frac{\sqrt{2}}{R} & -\frac{\sqrt{2}}{R}\mu_y \\ 0 & 0 & 1 \end{bmatrix}$$

where $R = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left((x_i - \mu_x)^2 + (y_i - \mu_y)^2\right)}$

and $\mu_x = \frac{1}{N}\sum_{i=1}^{N} x_i \quad \mu_y = \frac{1}{N}\sum_{i=1}^{N} y_i$

Write a function determines the homogeneous transformation to perform this normalization on any set of 2D homogeneous points, so that

% x_norm = T_norm*X
%
% defines an x_norm with a mean of 0 and rms
% (root mean squared) distance from the origin
% of sqrt(2).

Normalize **x1 and x2.**

Now estimate the fundamental matrix F from normalized points.

We want to find the fundamental matrix **F** that relates the location of a point **x1** in image 1 with its location **x**2 in image 2.
Writing

$$x_2^T F x_1 = 0 \quad F = \begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$$

Rewrite as a dot product for each point:

$$\vec{q} \cdot \vec{f} = \begin{bmatrix} u_2 u_1 & u_2 v_1 & u_2 & v_2 u_1 & v_2 v_1 & v_2 & u_1 & v_1 & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix}$$

Form the above left-hand product q for each of the N matches and stack them into one large matrix A. Solve for f using [U,S,V]=svd (A), where f is the last column in V. Reshape f into F.

Enforce Singularity

The fundamental matrix is a 3x3 matrix of rank 2, therefore it has singular eigenvalues. It is unlikely your matrix obeys this. Enforce singularity by

```
[U,S,V] = svd(F);
S(3,3) = 0;
Fn = U*S*V'
```

Because we normalized the points, this Fn is not the right Fn. The correct Fn is given by:
Fn = **T_norm_2'*Fn*T_norm_1**

If things went well, points in one image will specify an epipolar line in the other. View them using: epipolar_viewer(Fn,im1, im2) and check how good your points are.

2) Stereo reconstruction

These cameras are calibrated. The intrinsic parameter matrices are in the file C1_C2.mat. Use these to compute the Essential matrix via:

$$C_2^t F C_1 = E$$

Recall E has the extrinsic parameters between the two cameras, and these can be extracted from:

$$E = USV^t$$

e.g. the svd of E, and the rotation matrix R and translation vectors are given by:

$$R = UWV^t \quad \text{or} \quad R = UW^tV^t \qquad \vec{t} = u_3 \quad \text{or} \quad \vec{t} = -u_3$$

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

which results in a four way ambiguity. Use these results to compute R and t.
Then the projection matrices can be written:
M1 = C1*[eye(3) [0 0 0]']
M2 = C2*[R t] (with four possibilities)

Now for each point, solve for its x,y,z coordinates using the method outlined in class. You will get a 4-D homogeneous vector. Divide all the coordinates by the 4^{th} to get the x,y,z coordinates.

Reproject the points to see which of the four possibilities is correct. In addition, the reprojection will tell you something about how good your reconstruction is.