Camera Matrix

I compute the image coordinates and P matrix using the formula at slide 20, and I just have to change the t and R matrix. As skew is zero I didn't include it in the formula.

```
function R = rotation(degree)
   R =  [cosd(degree) 0 sind(degree);
      0 1 0;
      -sind(degree) 0 cos(degree)]
end
K = [578.952814872337060 0 645.829043343746890;
   0 585.355122127640240  366.668100334605700;
   0 0 1]
X = [0 0 1 1];
t = [0 0 3];
r1 = rotation (0);
P = [K(1,1)*r1(1,:)+K(1,3)*r1(3,:) K(1,1)*t(1)+K(1,3)*t(3);
   K(2,2)*r1(2,:)+K(2,3)*r1(3,:) K(2,2)*t(2)+K(2,3)*t(3);
    r1(3,:) t(3)]
i = (P(1,:).*X)/(P(3,:).*X)    %X coordinate
j = (P(2,:).*X)/(P(3,:).*X)    %Y coordinate
```

Result:

1) Image(645.829,366.668)

```
P =

   1.0e+03 *

    0.5790         0    0.6458    1.9375
         0    0.5854    0.3667    1.1000
         0         0    0.0010    0.0030
```

2) Image(645.829,366.668)

```
P =

   1.0e+03 *

    0.5790         0    0.6458    3.2291
         0    0.5854    0.3667    1.8333
         0         0    0.0010    0.0050
```

3) Image(749.3835,558.2417)

```
P =

   1.0e+03 *

    0.3232         0    0.4616    2.2270
   -0.1254    0.5854    0.1496    1.6854
   -0.0003         0    0.0004    0.0030
```

4) Image (3496.8,558.2417)

```
P =

   1.0e+04 *

    0.0323         0    0.0462    1.0622
   -0.0125    0.0585    0.0150    0.1685
   -0.0000         0    0.0000    0.0003
```

We can see that the P matrix is composed of R for the first 3x3 matrix and concatenated with T column, you can see that if you change the T without changing the R the left columns stays the same and only the T column is changed.

$$\mathbf{P} = \left[\begin{array}{ccc|c} p_1 & p_2 & p_3 & p_4 \\ p_5 & p_6 & p_7 & p_8 \\ p_9 & p_{10} & p_{11} & p_{12} \end{array}\right]$$

Fundamental Matrix

Detecting surf features and getting the coordinate of them

```
image1 = rgb2gray(imread('image1.jpg'));
image2 = rgb2gray(imread('image2.jpg'));
points1 = detectSURFFeatures(image1);
points2 = detectSURFFeatures(image2);
[f1, vpts1] = extractFeatures(image1, points1.selectStrongest(100));
[f2, vpts2] = extractFeatures(image2, points2.selectStrongest(100));
indexPairs = matchFeatures(f1, f2) ;
matchedPoints1 = vpts1(indexPairs(:, 1));
matchedPoints2 = vpts2(indexPairs(:, 2));
x1 = matchedPoints1.Location;
x2 = matchedPoints2.Location;
```

Normalise the points

```
x1(:,3) = 1;
x2(:,3) = 1;
T = [2/height 0 -1;
   0 2/width -1;
   0 0 1];
x1 = T*x1';
x2 = T*x2';
```

Building A matrix

```
A = [x2(1,:)'.*x1(1,:)' x2(1,:)'.*x1(2,:)' x2(1,:)' ...
x2(2,:)'.*x1(1,:)' x2(2,:)'.*x1(2,:)' x2(2,:)' ...
x1(1,:)' x1(2,:)' ones(13,1) ];
```

Computing the F matrix

```
[U,D,V] = svd(A);


F = reshape(V(:,9),3,3)';
```

[U,D,V] = svd(F);

F = U*diag([D(1,1) D(2,2) 0])*V';

Denormalising the F matrix

F = T'*F*T;

F

Result:

```
F =

  3×3 single matrix

   -0.0000   -0.0000    0.0012
    0.0000    0.0000   -0.0024
   -0.0012    0.0013    0.1500
```

Calculate the epipole

epi1 = null(F)

epi2 = null(F')

epi1 = epi1/epi1(3)

epi2 = epi2/epi2(3)

Their locations are

First Image:

```
epi1 =

  3×1 single column vector

  282.1273
  145.9504
    1.0000
```

Second Image: