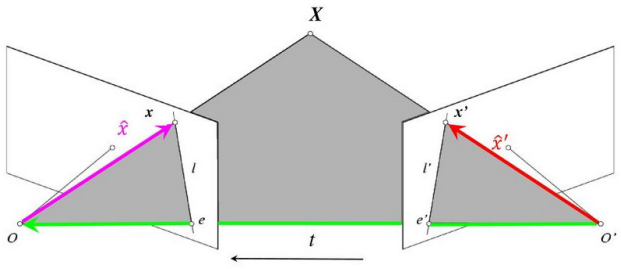
Computer Vision
Lab Exercise 5
Epipolar Geometry

This lab exercise introduces the concept of Epipolar geometry.



Where:

x, x' are points in the first and second image.

F is a 3×3 matrix called the Fundamental matrix.

F has 9 unknowns and 7 degrees of freedom because it has rank 2: $\det(F) = 0$.

We can determine F up to scale.

$F^T x' = l$ is the epipolar line associated with x'.

$Fx = l'$ is the epipolar line associated with x.

$F^T e' = 0$ and $Fe = 0$.

Assuming that the cameras can be approximated by the pinhole model, there are certain geometric relations between the 3D points and their corresponding 2D images. These relations can be used to impose constraints between image points. These constraints are useful because they reduce the correspondence problem to a 1D search along the Epipolar line.

1 Detecting Interest points and matching

First obtain feature points using the Harris & Hessian Affine detectors, which can be downloaded from `http://www.robots.ox.ac.uk/~vgg/research/affine/detectors.html`.

```
sudo ./extract_features -hesaff -i img.png
```

We will also provide the extracted features which you can use. If you have difficulties extracting them with the binaries from the above website you can use the provided features.
Once you have the extracted features and they are loaded, the *vl_feat* library is used to get matches

```
[feat1,desc1,¬,¬] = ...
    loadFeatures('TeddyBearPNG/obj02_001.png.haraff.sift');
[feat2,desc2,¬,¬] = ...
    loadFeatures('TeddyBearPNG/obj02_002.png.haraff.sift');

% Using vl_ubcmatch to match descriptors
disp('Matching Descriptors');
[matches, ¬] = vl_ubcmatch(desc1,desc2);
disp(strcat( int2str(size(matches,2)), ' matches found'));
```

Listing 1: From demo_lab5.m

2 Estimating the Fundamental matrix

If x_i and y_i denote x and y coordinates of the i^{th} point p_i respectively (the ' superscript indicates points in the second image) then,

$$\begin{bmatrix} x_i' & y_i' & 1 \end{bmatrix} \underbrace{\begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}}_{\mathrm{F}} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix} = 0, \tag{1}$$

Equation 1 can also be written as

$$\underbrace{\begin{bmatrix} x_1x'_1 & x_1y'_1 & x_1 & y_1x'_1 & y_1y'_1 & y_1 & x'_1 & y'_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ x_nx'_n & x_ny'_n & x_n & y_nx'_n & y_ny'_n & y_n & x'_n & y'_n & 1 \end{bmatrix}}_{\text{A}} \begin{bmatrix} f_{11} \\ f_{21} \\ f_{31} \\ f_{12} \\ f_{22} \\ f_{32} \\ f_{13} \\ f_{23} \\ f_{33} \end{bmatrix} = 0, \qquad (2)$$

where F denotes the fundamental matrix.

Note: Although F has 9 unknowns, we need only 8 correspondences in the matrix A, because we work with homogenous coordinates and F can be solved only up to scale (Homogenous coordinates are scale invariant).

2.1 Normalized Eight-point Algorithm with RANSAC

A careful normalization of the input data (the point correspondences) leads to improvement in the conditioning of the problem, and hence in the stability of the result.

2.1.1 Computing the fundamental matrix:

First pick 8 point correspondences randomly from the set of matches(can be accomplished with MATLAB's `randperm`).

```
% Randomly select P points from two sets of matched points
perm = ...
seed = ...
```

Listing 2: From estimateFundamentalMatrix.m

2.1.2 Normalization:

We want to apply a similarity transformation to the set of points p_i so that their mean is 0 and the average distance to the mean is $\sqrt{2}$.

Let $m_x = \frac{1}{n}\sum_{i=1}^{n} x_i, m_y = \frac{1}{n}\sum_{i=1}^{n} y_i, d = \frac{1}{n}\sum_{i=1}^{n}\sqrt{(x_i - m_x)^2 + (y_i - m_y)^2}$, and

$$T = \begin{bmatrix} \sqrt{2}/d & 0 & -m_x\sqrt{2}/d \\ 0 & \sqrt{2}/d & -m_y\sqrt{2}/d \\ 0 & 0 & 1 \end{bmatrix}$$

where (x_i) and (y_i) denote x and y coordinates of a point (p_i), respectively. Complete the function <normalize.m> by first subtracting the row wise mean from the points and also fill in values according to the equations given above.

```
% Compute d: scale all X so that the average distance to ...
    the mean is sqrt(2).
% Check the lab file for details
d = ...

% Compose the matrix T
T = ...
```

Listing 3: From normalize.m

2.1.3 Computing the Fundamental matrix

To compute the fundamental matrix, the following steps are necessary. (In the <estimateFundamentalMatrix.m> function):

- Construct a matrix A as shown in equation (2) from the normalized points. (In the <composeA.m> function).
- Fill in the <computeF.m> function. First we take the SVD of the constructed A matrix (A = UΣV'). The last column of the 'V' matrix is reshaped into a 3 x 3 matrix, to obtain the F matrix.

```
    % Solution for Af=0 using SVD
    [] = svd(A);
    f = V();
    F = ...
```

Listing 4: From computeF.m

– This 'F' matrix can be non-singular. To singularize, take the SVD of the newly estimated 'F' ($F = USV'$) and set the last element of the last row of S to zero and multiply the factors of the decomposition (U, S and V') again to get the new 'F'.

```
% Resolve the rank 2 constraint: det(F) ...
    =0 using SVD
[] = svd(F);
S(.. ,.. ) = 0;
F = ...
```

Listing 5: From computeF.m

– Denormalize the 'F' matrix by multiplying it with the transformation matrices determined with the <normalize.m> function: $F = T_2' F T_1$)

2.1.4 Find inliers

The next step is to find inliers by computing perpendicular errors between the points and the epipolar lines in each image. We want that both points x in the first image are close to their epipolar line $F^T x'$, and points in the second image x' are close to their epipolar line Fx. Sampson error accomplishes this, and it is defined as follows :

$$d_i = \frac{({x'}_i^T F x_i)^2}{(Fx_i)_1^2 + (Fx_i)_2^2 + (F^T x_i')_1^2 + (F^T x_i')_2^2}, \tag{3}$$

where x_i and ${x'}_i$ are the matches from the first images and the second image, respectively and $(Fx)_k^2$ is the square of the k^{th} entry of the vector Fx. If d_i is smaller than some threshold, the match is said to be an inlier (set threshold to 50 here).

```
% Calculate Sampson distance for each point
% Compute numerator and denominator at first
numer = ...
```

```
4       denom = ...
5       sd    = numer./denom;
6
7       % Return inliers for which sd is smaller than threshold
8       inliers = find(sd<threshold);
```

Listing 6: From computeInliers.m

2.1.5 Repeat and keep the best model

This process is repeated, recomputing and saving a new 'F' matrix if a larger amount of inliers are obtained.

```
1   % if number of inlier > the best so-far, use new F
2  if size(inliers,2)>bestcount
3      bestcount   = ...
4      bestF       = ...
5      bestinliers = ...
6  end
```

Listing 7: From estimateFundamentalMatrix.m

As you have done in the earlier lab, compute the iterations required for RANSAC.

```
1  % Calculate how many iterations we need by computing:
2  % i=log(eps)/log(1-q^p),
3  % where p=8 (the number of matches)
4  % q= #inliers/#total_pairs (the proportion of inliers over ...
       total pairs)
5  eps = 0.001;
6  N1  = ...
7  N   = ...
8  q   = ...
9
10 % To prevent special cases, always run at least a couple ...
       of times
11 iterations = max(miniter,ceil(... ));
```

Listing 8: From estimateFundamentalMatrix.m

3 Outputs

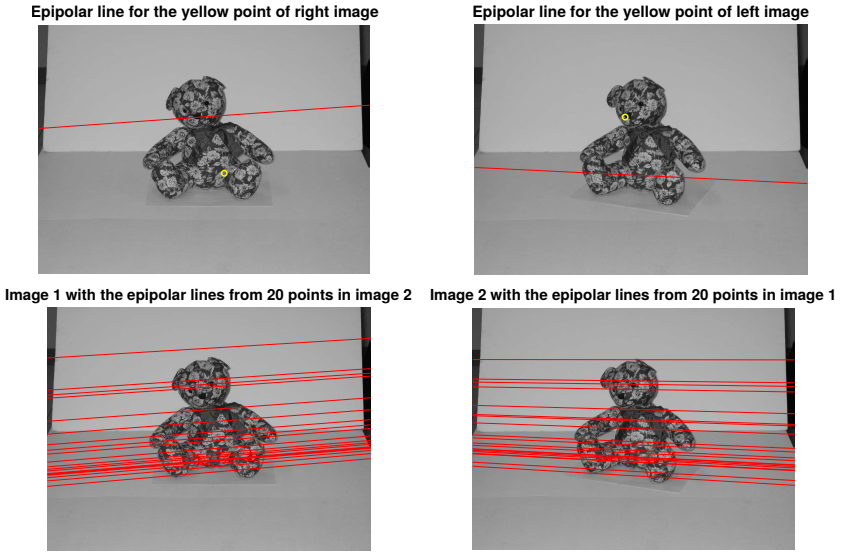


Figure 1: Feature matching and Epipolar lines for the teddy bear image example