8 Point Algorithm

Essential and Fundamental Matrices

- Calibrated Cameras – Essential Matrix
- Un-Calibrated Cameras – Fundamental matrix
- There are five parameters in the essential matrix—three for rotation and two for the direction of translation (scale is not set)—along with two other constraints. The two additional constraints on the essential matrix are: (1) the determinant is 0 because it is rank-deficient (a 3-by-3 matrix of rank 2); and (2) its two nonzero singular values are equal because the matrix $\hat{T}$ is skewsymmetric and R is a rotation matrix.
- The fundamental matrix F is just like the essential matrix E, except that F operates in image pixel coordinates whereas E operates in physical coordinates. The fundamental matrix F is of rank 2. The fundamental matrix has seven parameters, two for each epipole and three for the homography that relates the two image planes (the scale aspect is missing from the usual four parameters).

8-point Linear Algorithm

Given a number of corresponding points, we could use the epipolar constraints to try to recover camera pose (R, T).

$$E = \hat{T}R = \begin{bmatrix} e_{11} & e_{12} & e_{13} \\ e_{21} & e_{22} & e_{23} \\ e_{31} & e_{32} & e_{33} \end{bmatrix} \in R^{3\times 3}$$

8-point Linear Algorithm

$$p_r^T E p_l = \begin{bmatrix} x_r & y_r & z_r \end{bmatrix} E \begin{bmatrix} x_l \\ y_l \\ z_l \end{bmatrix}$$

$$= x_l x_r e_{11} + x_l y_r e_{12} + z_l z_r e_{13} + y_l x_r e_{21} + y_l y_r e_{22} + y_l z_r e_{23}$$
$$+ z_l x_r e_{31} + z_l y_r e_{32} + z_l z_r e_{33}$$

Stacking Kronecker product

$$= a^T E^s = 0$$

$$a = \begin{bmatrix} x_l x_r, x_l y_r, z_l z_r, y_l x_r, y_l y_r, y_l z_r, z_l x_r, z_l y_r, z_l z_r \end{bmatrix}$$

$$E^s = \begin{bmatrix} e_{11}, e_{12}, e_{13}, e_{21}, e_{22}, e_{23}, e_{31}, e_{32}, e_{33} \end{bmatrix}^T$$

8-point Linear Algorithm

$$a^T E^s = 0$$

$$a = [x_l x_r, x_l y_r, z_l z_r, y_l x_r, y_l y_r, y_l z_r, z_l x_r, z_l y_r, z_l z_r]$$

$$E^s = [e_{11}, e_{12}, e_{13}, e_{21}, e_{22}, e_{23}, e_{31}, e_{32}, e_{33}]^T$$

Given a set of corresponding image points $(p_l^j, p_r^j), j = 1,2,3 \cdots n$

define a matrix

$$\chi \in R^{n \times 9}$$

$$\chi = [a^1, a^1, \cdots a^n]^T$$

$$\chi E^s = 0$$

8-point Linear Algorithm

$$\chi E^s = 0$$

For the solution to be unique (up to a scale factor) the rank of X must be exactly 8 (excluding E^S=0). Since there may be errors in correspondences, there may only be a least-squares solution, which is the eigenvector of the X^TX that corresponds to its smallest eigenvalue. If the rank of X is smaller than 8, then there are multiple solutions (corresponding points lie on a plane).

8-point Linear Algorithm

$$\chi E^s = 0$$

Compute SVD of $X=UDV^T$ and the ninth column of V (corresponding to the smallest eigenvalue) is the solution for E^S.

Rearrange the nine elements of E^S to a 3x3 matrix. This 3x3 matrix E will be in general not in the essential space.

Compute SVD of $E = UDV^T$ and $D=\{\sigma_1, \sigma_1, \sigma_1\}$

Replace D by $\Sigma=\{1, 1, 0\}$ and computer the new E that is in (normalized) essential space

Solving R and T from E

Essential Matrix $E = \hat{T}R \in R^{3\times 3}$

A nonzero matrix E is an essential matrix if and only if E has a singular value decomposition (SVD)$E=UDV^T$ with

$D=\mathrm{diag}\{\sigma, \sigma, 0\}$

There exist exactly four solutions (2 with positive Z)

$$If\ a\ given\ matrix\ E \in R^{3\times 3}\ has\ SVD\ E = UDV^T\ then$$

$$(\hat{T}_1, R_1) = (UR_z(+\frac{\pi}{2})DU^T, UR_z^T(+\frac{\pi}{2})V^T)$$

$$(\hat{T}_2, R_2) = (UR_z(-\frac{\pi}{2})DU^T, UR_z^T(-\frac{\pi}{2})V^T)$$

R_z: Rotation matrix about Z axis

8-point Linear Algorithm

Essential Matrix $E = \hat{T}R \in R^{3\times 3}$

Compute SVD of the new $E=UDV^T$

Calculate R and T as

$$R = UR_z^T(\pm\frac{\pi}{2})V^T$$

$$\hat{T} = UR_z(\pm\frac{\pi}{2})\Sigma U^T$$

$$R_Z^T(\pm\frac{\pi}{2}) = \begin{bmatrix} 0 & \pm 1 & 0 \\ \mp 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Normalized

$$\Sigma = \begin{bmatrix} \sigma & 0 & 0 \\ 0 & \sigma & 0 \\ 0 & 0 & 0 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

The unknown scale factor can be calculated if object size is known.

$$\hat{T} = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ T_y & T_x & 0 \end{bmatrix}$$

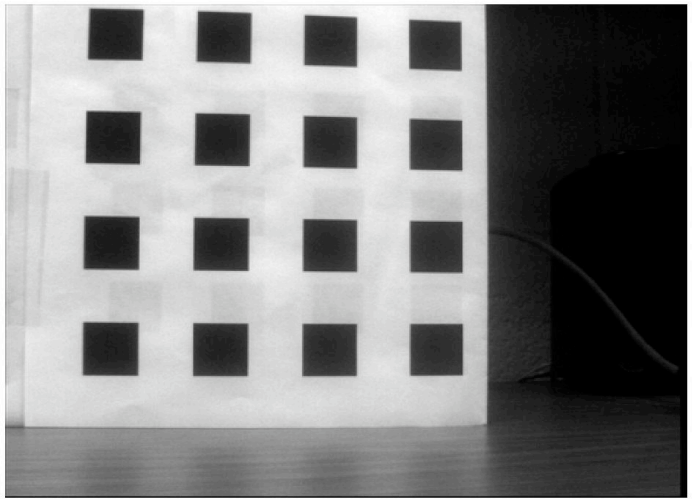
Limitations

There is no guarantee that the estimated pose (R,T) is as close as possible to the true solution.

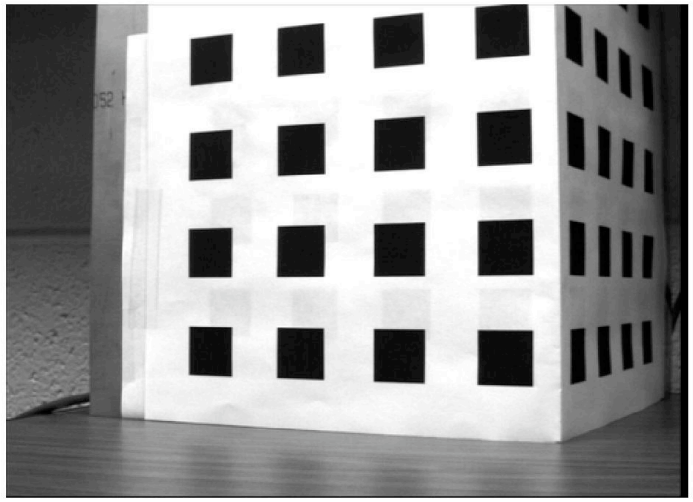
Even if we were to accept such an (R,T), a noisy image pair would not necessary give rise a consistent 3-D reconstruction.

Fundamental Matrix

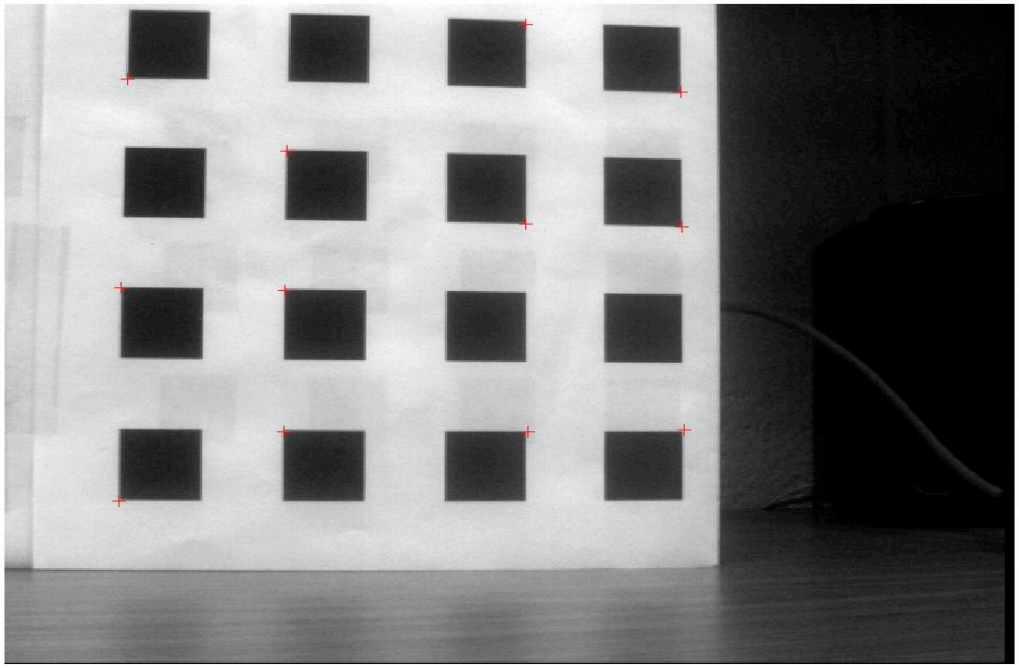
Left Image



Right Image



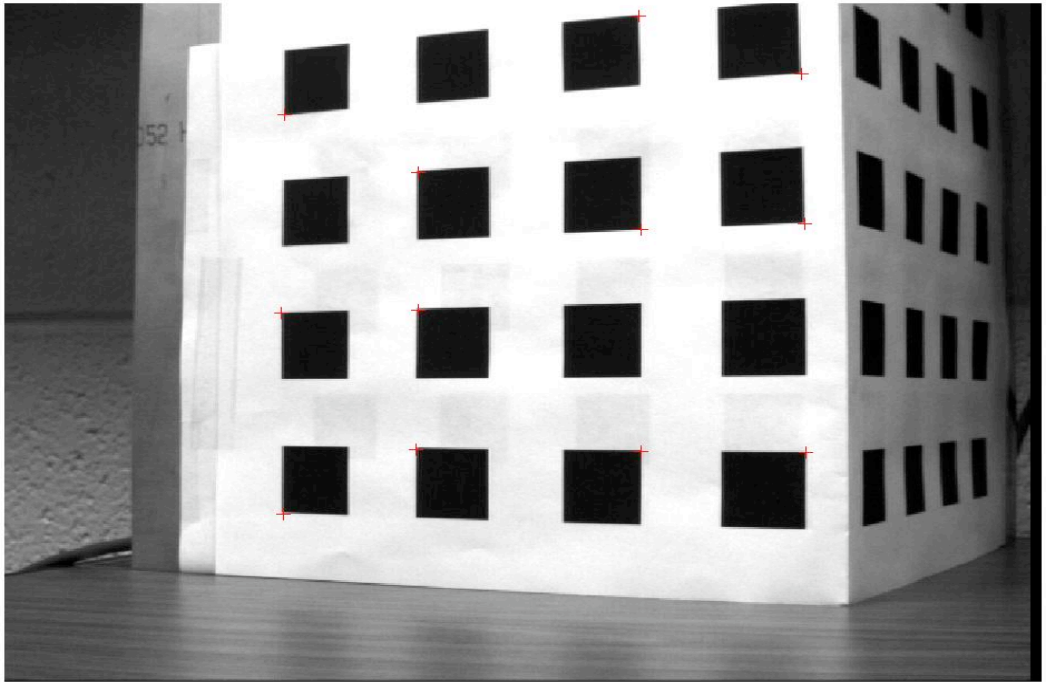
Data from Left Image



$$\overline{P}_l$$

79	57
75	214
74	374
181	111
180	216
179	322
334	16
334	166
335	322
433	67
434	168
435	321

Pick Points in Right Image



$$\overline{P}_r$$

175	82
173	227
174	374
258	124
258	225
257	327
396	10
397	166
397	328
497	52
499	162
500	329

Fundamental Matrix

$$\bar{p}_r^T F \, \bar{p}_l = 0$$

$$\begin{bmatrix} x_r & y_r & 1 \end{bmatrix} \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix} \begin{bmatrix} x_l \\ y_l \\ 1 \end{bmatrix} = 0$$

$$\begin{bmatrix} x_r & y_r & 1 \end{bmatrix} \begin{bmatrix} F_{11}x_l + F_{12}y_l + F_{13} \\ F_{21}x_l + F_{22}y_l + F_{23} \\ F_{31}x_l + F_{32}y_l + F_{33} \end{bmatrix} = 0$$

$$(F_{11}x_l + F_{12}y_l + F_{13})x_r + (F_{21}x_l + F_{22}y_l + F_{23})y_r + F_{31}x_l + F_{32}y_l + F_{33} = 0$$

$$\begin{bmatrix} x_l x_r & y_l x_r & x_r & x_l y_r & y_l y_r & y_r & x_l & y_l & 1 \end{bmatrix} \begin{bmatrix} F_{11} & \bullet & \bullet & F_{33} \end{bmatrix}^T = 0$$

$$AF = 0$$

$A = UDV^T$ F is the components of V corresponding to the least singular value of A

Calculate F = UDV^T and set the smallest singular value to zero. Reconstruct F with the new D.

Constructing Matrix A

$$\overline{p}_r^T F \, \overline{p}_l = 0$$

1. Convert pixel coordinates into homogeneous coordinates
2. Each row of A will be:

$$\begin{bmatrix} x_l x_r & y_l x_r & x_r & x_l y_r & y_l y_r & y_r & x_l & y_l & 1 \end{bmatrix}$$

Singular Value Decomposition

Perform SVD on A to get the eigenvector associated with the smallest singular value, i.e., the Null Space for A.

In Matlab:
[U, S, V] = svd(A)

Select the column of V that is associated with the least (or zero) singular value in S.
This column is the nontrivial solution that is unique up to a scale factor.

Calculate Fundamental Matrix F

a. Rearrange the column (solution for F) to make a 3-by-3 matrix.
b. Perform SVD for F
   In Matlab, [U, S, V] = svd(F)
c. Examine the smallest singular value, if it's not zero, set it to zero

$$F = USV^T$$

It's a new S. S(3,3) = 0

d. Multiply USV to get better/final F
   In Matlab,

7-point or 8-point Algorithm

- The 7-point algorithm uses exactly seven points, and it uses the fact that the matrix F must be of rank 2 to fully constrain the matrix. The advantage of this constraint is that F is then always exactly of rank 2 and so cannot have one very small eigenvalue that is not quite 0. The disadvantage is that this constraint is not absolutely unique and so three different matrices might be returned (the returned matrix must be a 9-by-3 matrix, so that all three returns can be accommodated).
- The 8-point algorithm just solves F as a linear system of equations. If more than eight points are provided then a linear least-squares error is minimized across all points.

Problems and Solutions

F is often ill-conditioned, which means small variations in the data points (x, y coordinates) selected will completely mess up the calculation for F.

The problem with both the 7-point and 8-point algorithms is that they are extremely sensitive to outliers (even if we have many more than eight points in the 8-point algorithm).

They can fail if the points supplied provide less than the required amount of information, such as when one point appears more than once or when multiple points are collinear or coplanar with too many other points.

How to solve or minimize this problem?

1. Perform data normalization and
2. RANdom Sample Consensus (RANSAC)
3. Least Median Squares (LMedS)

Data Points Normalization

$$\overline{P}_r^T F \overline{P}_l = 0$$

$$(H_r \overline{P}_r)^T \overline{F} (H_l \overline{P}_l) = 0$$

$$\hat{P}_r = H_r \overline{P}_r$$

$$\hat{P}_l = H_l \overline{P}_l$$

Use an affine matrix H to normalize ("balance") the data points

$$\hat{P}_r^T \overline{F} \hat{P}_l = 0$$

If we use normalized points $\hat{p}_l \ and \ \hat{p}_r$

and send them through the 8 point algorithm, we can calculate for $\overline{F}$ first, and then recover $F = (H_r)^T \overline{F} H_l$

What are $\hat{p}_l$ and $\hat{p}_r$?

$$\text{centroid}: \quad \bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i \quad \text{and} \quad \bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i$$

$$\text{average distance to the centroid}: \quad \bar{d} = \frac{\sum_i \sqrt{(x_i - \bar{x})^2 + (y_i - \bar{y})^2}}{n}$$

$$\text{normalized points } \hat{p}_l^i \text{ or } \hat{p}_r^i = \begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ 1 \end{bmatrix} = \begin{bmatrix} (x_i - \bar{x})/\bar{d} \\ (y_i - \bar{y})/\bar{d} \\ 1 \end{bmatrix}$$

$$\hat{p}_l^i = H_l \bar{P}_i^l \quad \text{and} \quad \hat{p}_r^i = H_r \bar{P}_i^r$$

$$\begin{bmatrix} \hat{x}_i \\ \hat{y}_i \\ 1 \end{bmatrix} = \begin{bmatrix} (x_i - \bar{x})/\bar{d} \\ (y_i - \bar{y})/\bar{d} \\ 1 \end{bmatrix} = \begin{bmatrix} 1/\bar{d} & 0 & -\bar{x}/\bar{d} \\ 0 & 1/\bar{d} & -\bar{y}/\bar{d} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_i \\ y_i \\ 1 \end{bmatrix}$$

Use left points and right points to calculate H_l and H_r

OpenCV Functions

Mat findFundamentalMat(const Mat& points1, const Mat& points2,
vector<uchar>& status, int method=FM RANSAC,
double param1=3., double param2=0.99);
Mat findFundamentalMat(const Mat& points1, const Mat& points2,
int method=FM RANSAC, double param1=3., double
param2=0.99);

Method : **CV_FM_7POINT**
CV_FM_8POINT
CV_FM_RANSAC
CV_FM_LMEDS

This OpenCV functions does the point normalization internally.

Epipoles

$$\bar{p}_r^T F \, \bar{e}_l = 0 \qquad \text{for every } \bar{p}_r$$

$$F \, \bar{e}_l = 0 \quad \text{and} \quad (\bar{e}_r)^T F = 0$$

Accurate epipole location helps to

- *refine the location of corresponding epipolar lines*
- *check the geometry consistency of the entire construction*
- *simplify stereo geometry*
- *recover 3-D structure in the case of uncalibrated stereo*

$$F = UDV^T$$

Epipoles are associated with the Null Space of F (right epipole) and F transpose (left epipole).

e_l is the column of V corresponding to the null singular value

e_r is the column of U corresponding to the null singular value

Pick the third column of U, that's the homogeneous coordinates for the right epipole

Pick the third column of V, that's the homogeneous coordinates for the left epipole

We can calculate for pixel coordinates.

Fundamental Matrix and Epipoles

$$F = \begin{bmatrix} 0.0000 & -0.0000 & 0.0062 \\ 0.0000 & 0.00001 & -0.0068 \\ -0.0061 & 0.0088 & -0.4538 \end{bmatrix}$$

$$e_l = \begin{bmatrix} 307.9824 & 267.3172 \end{bmatrix}$$

$$e_r = \begin{bmatrix} 372..2064 & 272.3900 \end{bmatrix}$$

Epipolar line

$$\bar{u}_r = F\,\bar{p}_l$$

Use one $\bar{p}_l$ to calculate one $\bar{u}_r$, the entries in it are the coefficients for the line.

$$\bar{u}_r = \begin{bmatrix} a \\ b \\ c \end{bmatrix}$$

$$ax + by + c = 0$$

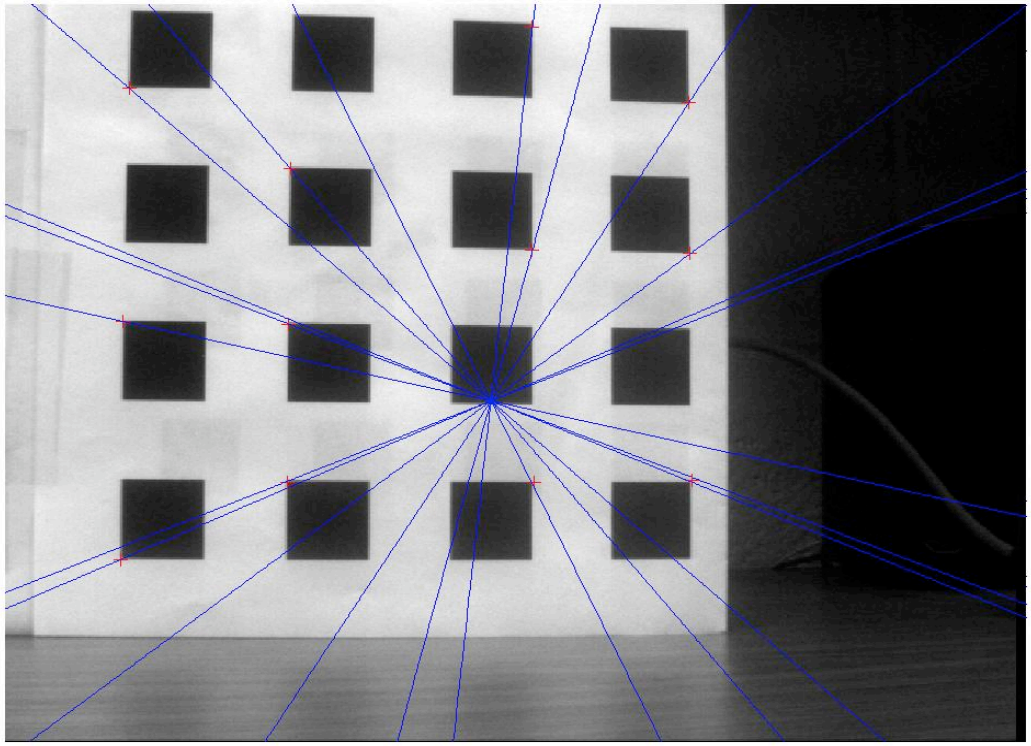
is the epipolar line.

OpenCV Function

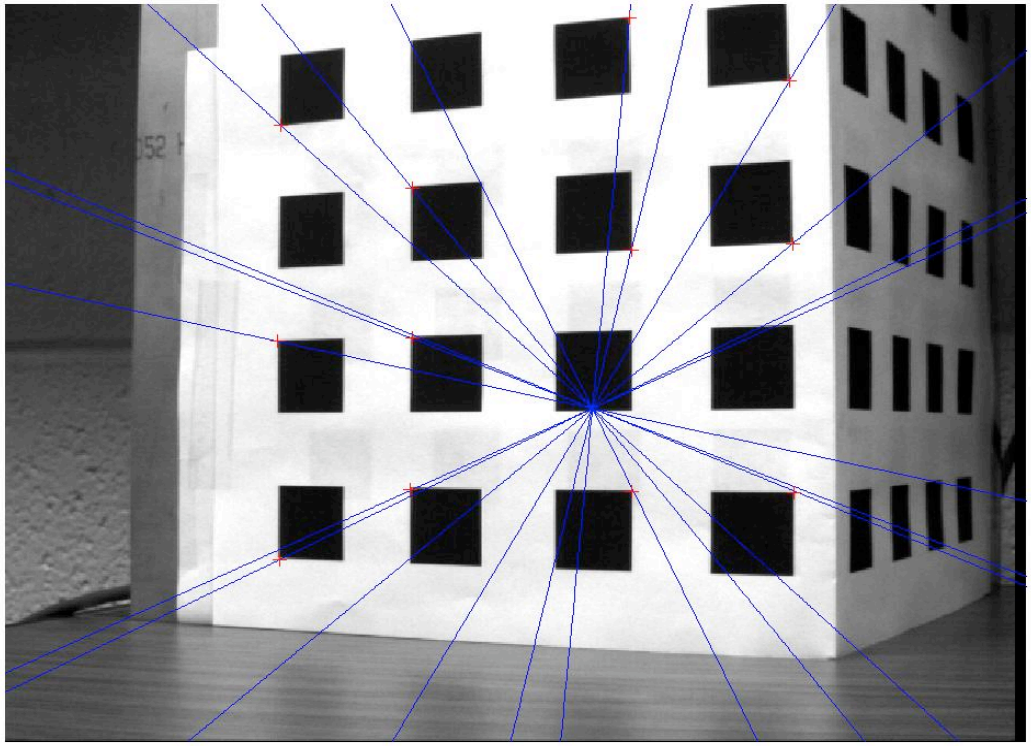
void computeCorrespondEpilines(const Mat& points, int whichImage, const Mat& F, vector<Vec3f>& lines);

For points in one image of stereo pair, compute the epipolar lines in the other image.

Epipolar Lines and Epipole (Left)



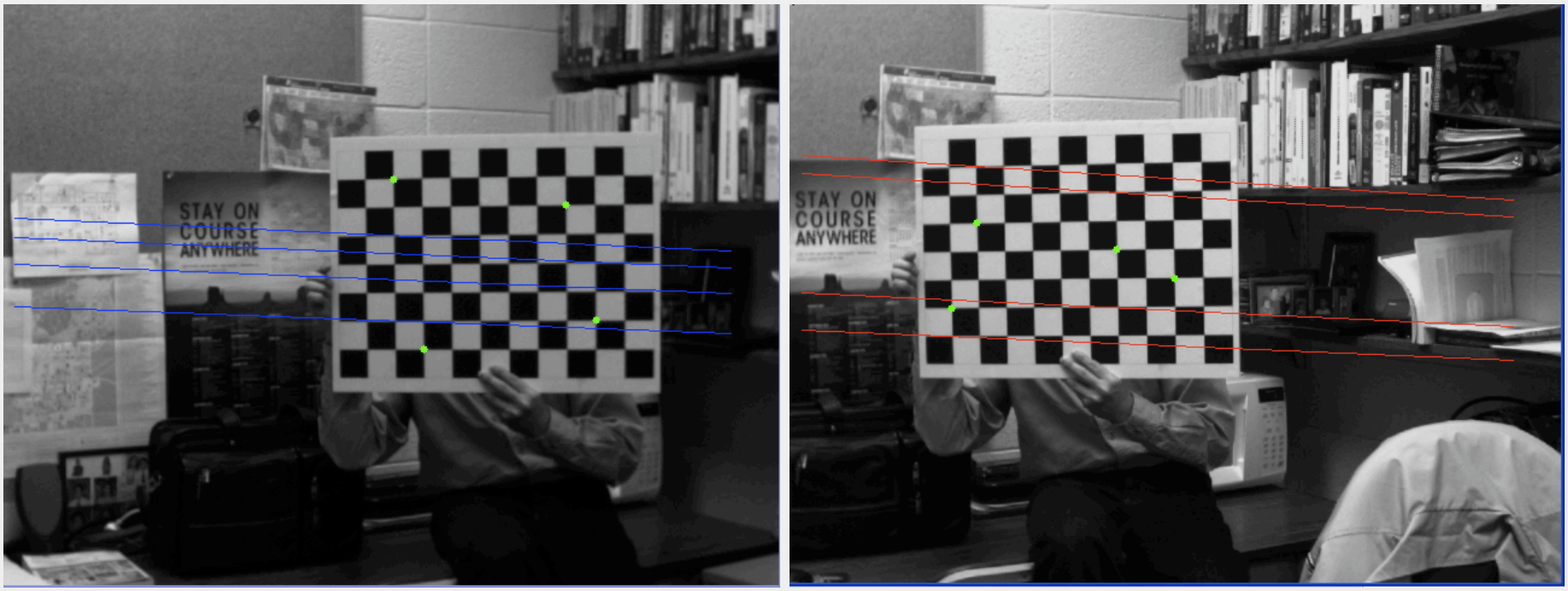
Epipolar Lines and Epipole (Right)



Near Canonical Configuration

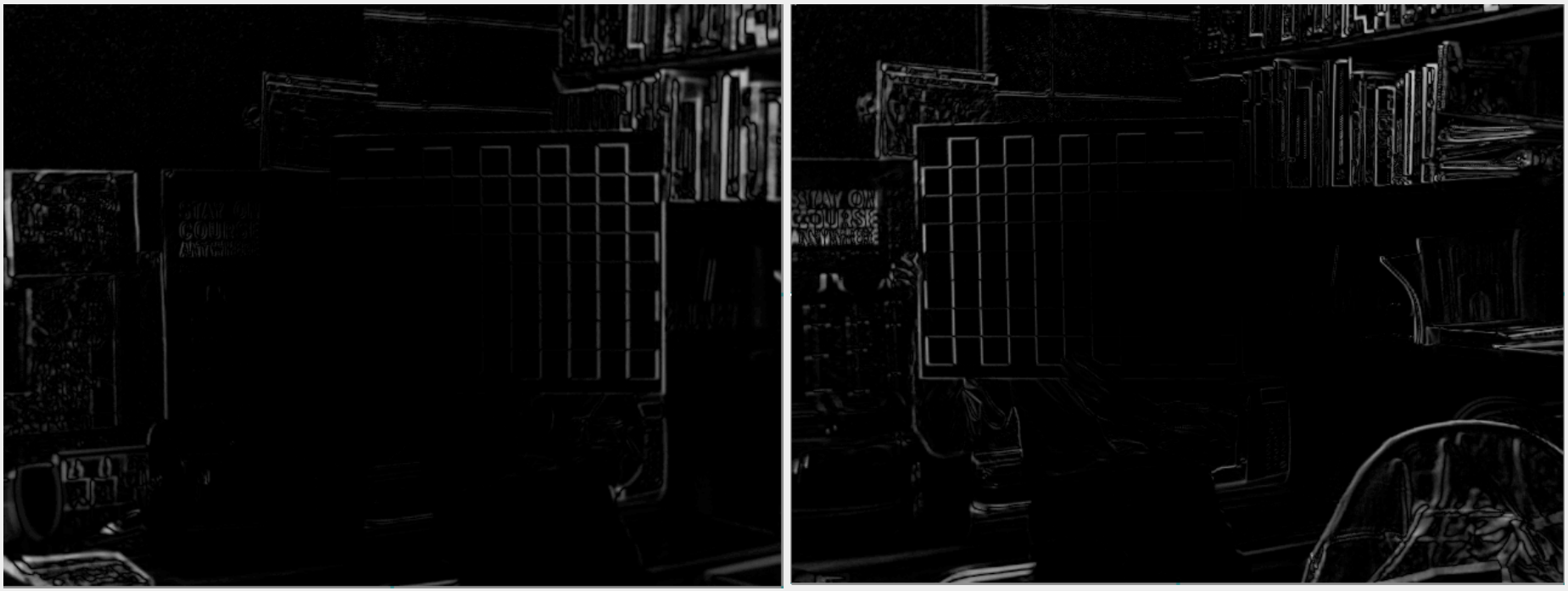
Lens distortion not corrected

- Epipolar lines are not parallel (very close though)
- Epipoles are somewhere far away from image planes but not infinity.



Lens Distortion Correction

- Absolute difference between the original image and the undistorted image.



Near Canonical Configuration

Lens distortion corrected

- Epipolar lines are not parallel (very close though)
- Epipoles are somewhere far away from image planes but not infinity.