Low-level Vision Tasks

Multiple-view Geometry and Projective Reconstruction

Kuk-Jin Yoon

Visual Intelligence Lab.
Department of Mechanical Engineering

KAIST

Two(or Multi)-view Geometry

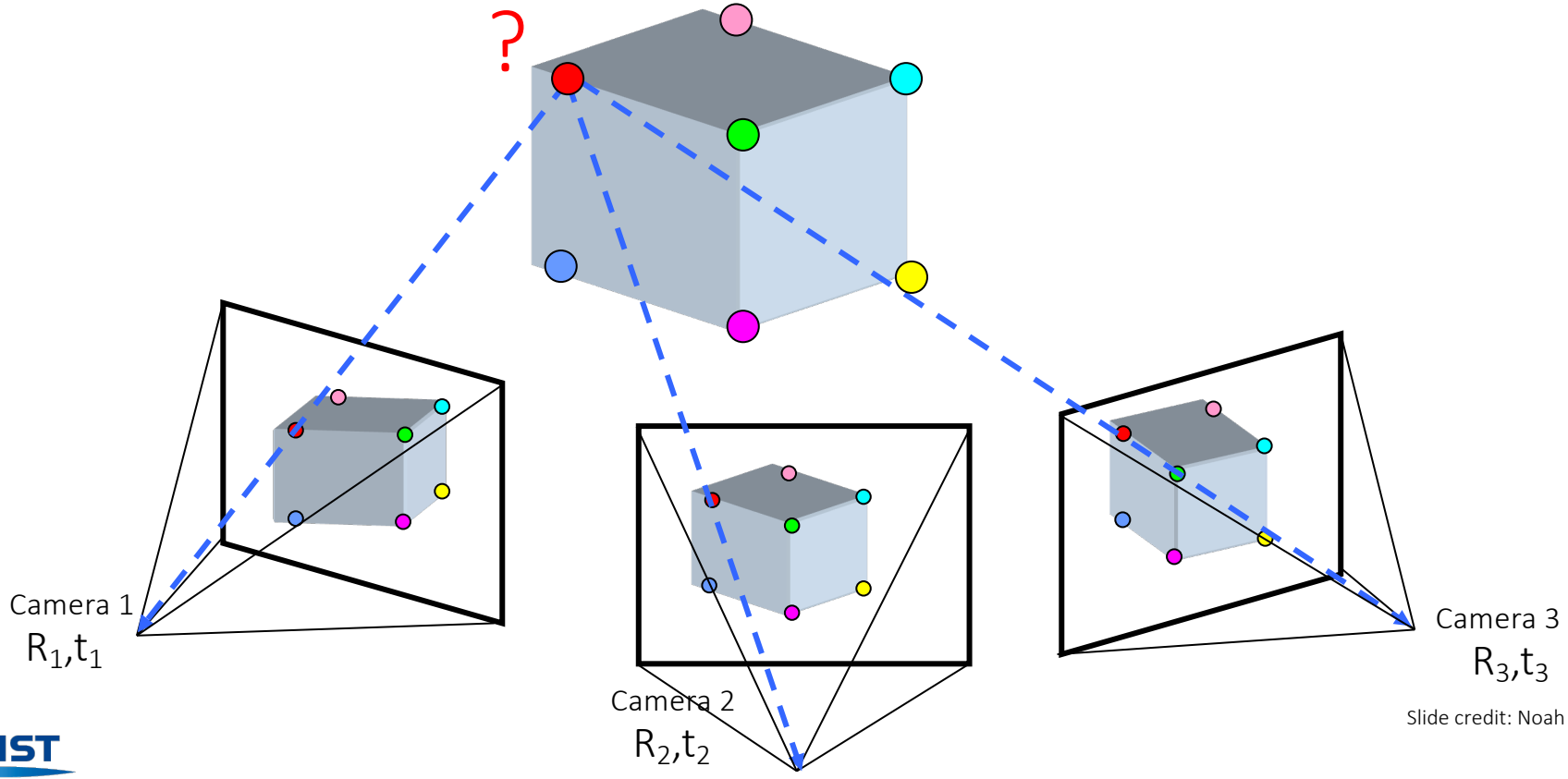
- Cameras P and P' such that

$$\mathbf{x} = \mathrm{P}\mathbf{X} \qquad \mathbf{x}' = \mathrm{P}'\mathbf{X}$$

- Baseline between the cameras is non-zero.
  - Scene geometry (structure)
    - Given projections of the same 3D point in two or more images, how do we compute the 3D coordinates of that point?
  - Correspondence (stereo matching)
    - Given a point in just one image, how does it constrain the position of the corresponding point in the second image?
  - Camera geometry (motion)
    - Given a set of corresponding points in two images, what are the cameras for the two views?

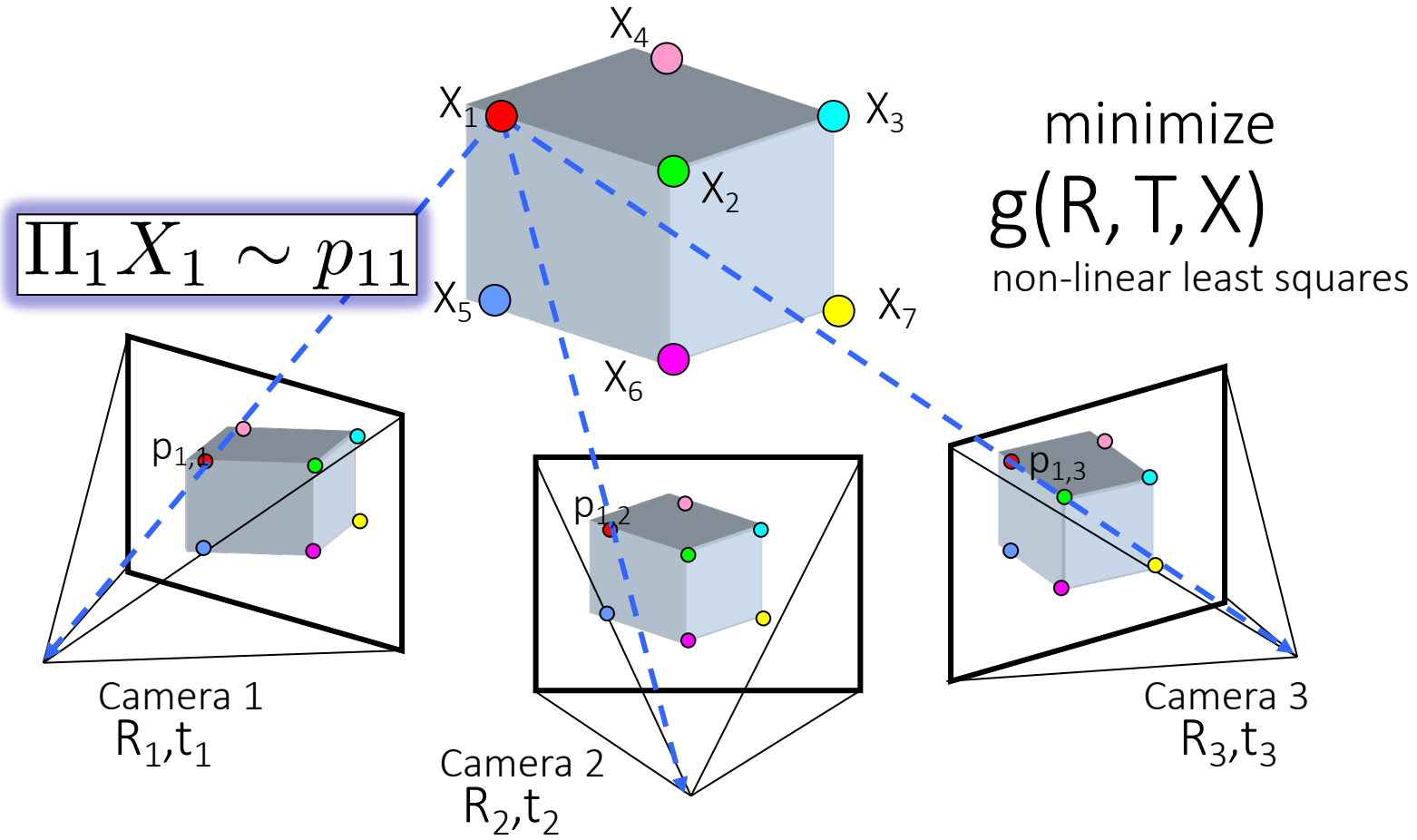
Multi-view Geometry Problems

- **Structure:** Given projections of the same 3D point in two or mo re images, compute the 3D coordinates of that point



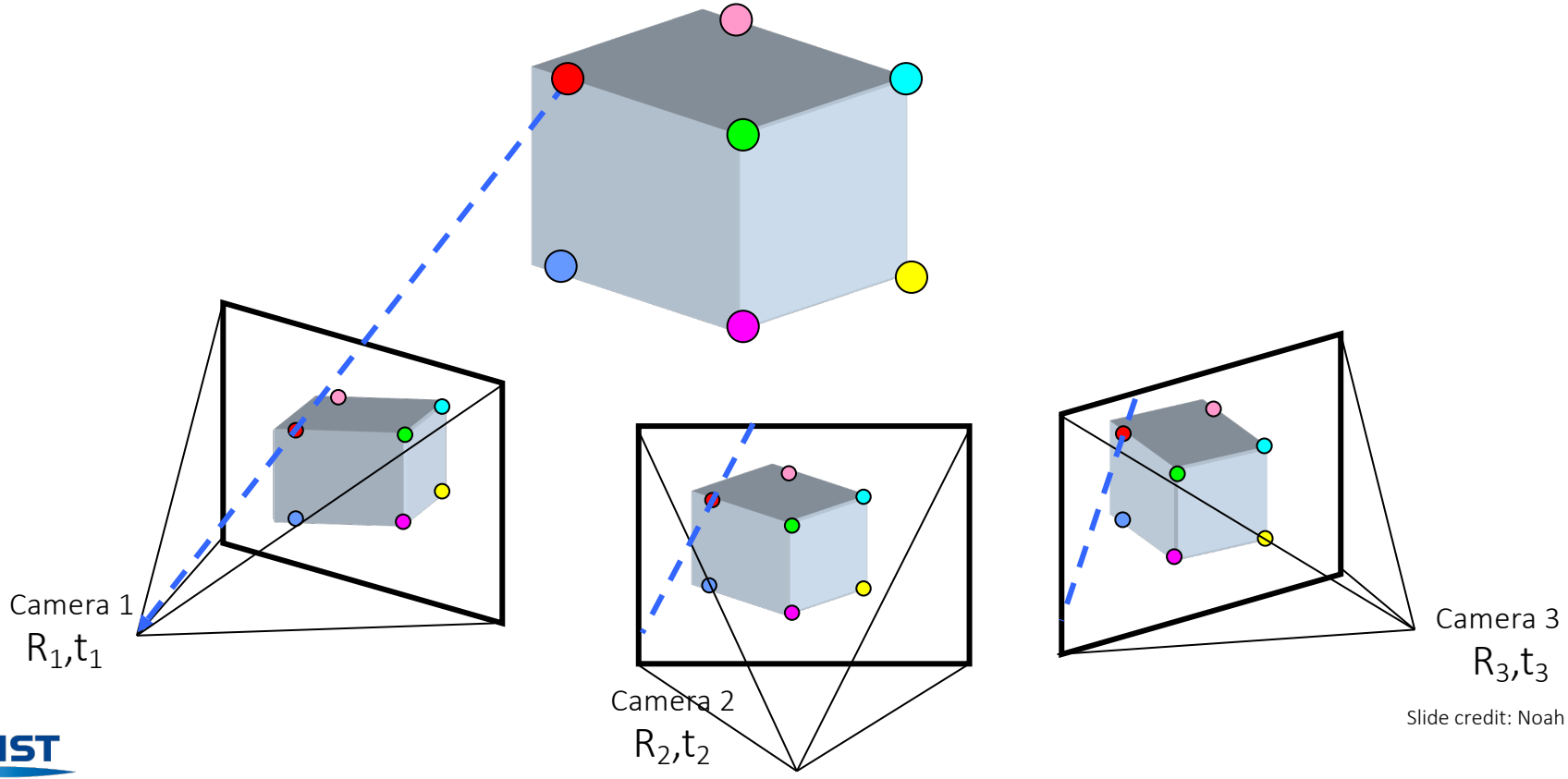
Slide credit: Noah Snavely

Multi-view Geometry Problems: Structure from Motion



Multi-view Geometry Problems

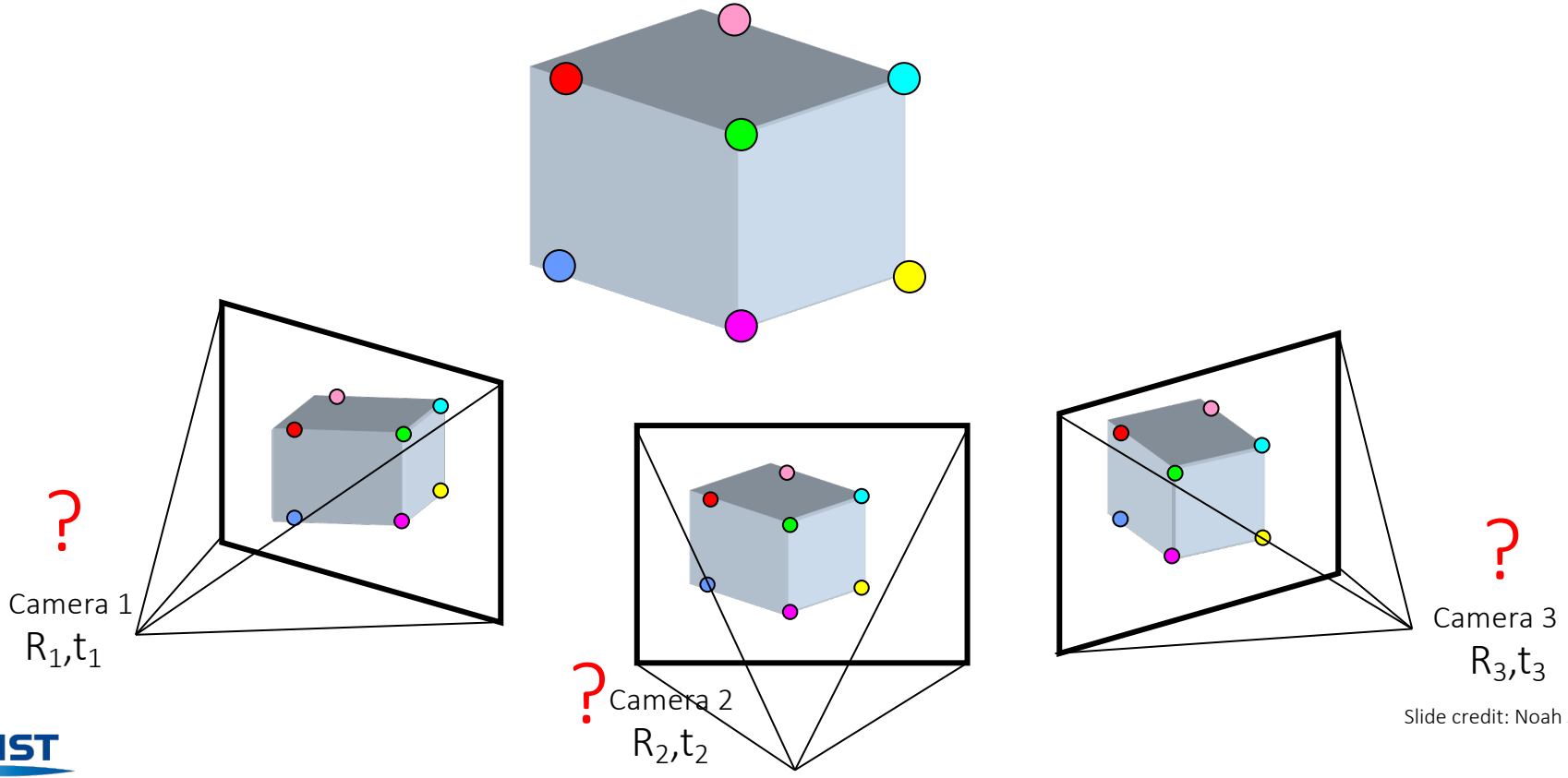
- **Stereo correspondence:** Given a point in one of the images, where could its corresponding points be in the other images?



Slide credit: Noah Snavely

Multi-view Geometry Problems

- **Motion:** Given a set of corresponding points in two or more im ages, compute the camera parameters



Slide credit: Noah Snavely

Two(or Multi)-view Geometry

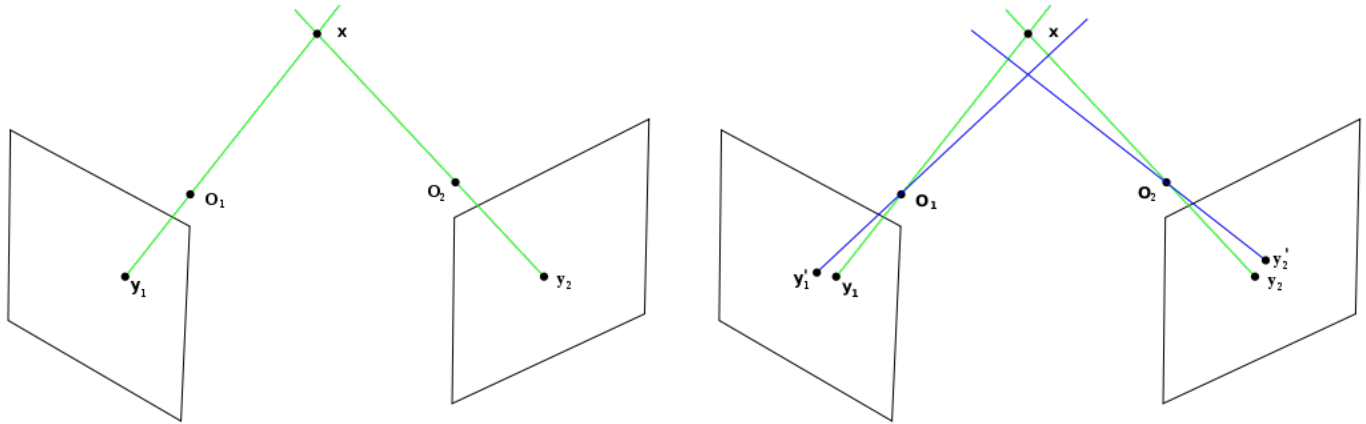
- Cameras P and P' such that

$$\mathbf{x} = \mathrm{P}\mathbf{X} \qquad \mathbf{x}' = \mathrm{P}'\mathbf{X}$$

- Baseline between the cameras is non-zero.
  - Scene geometry (structure)
    - Given projections of the same 3D point in two or more images, how do we compute the 3D coordinates of that point?
  - Correspondence (stereo matching)
    - Given a point in just one image, how does it constrain the position of the corresponding point in the second image?
  - Camera geometry (motion)
    - Given a set of corresponding points in two images, what are the cameras for the two views?

Triangulation

- Given projections of a 3D point in two or more images (with known camera matrices), find the coordinates of the point
  - Ideal case: two visual rays passing through two image points and optical centers intersect at one point in the scene.
  - Real case: two visual rays do not meet exactly because of noise and numerical errors.

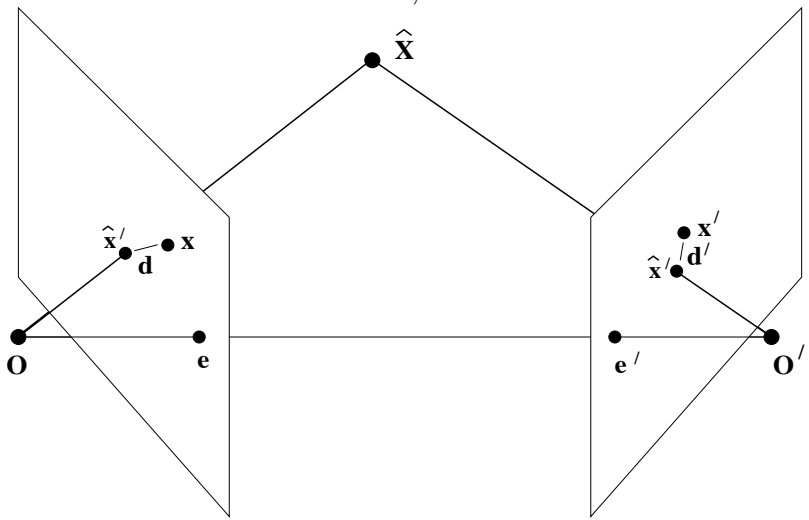


Triangulation

Triangulation :

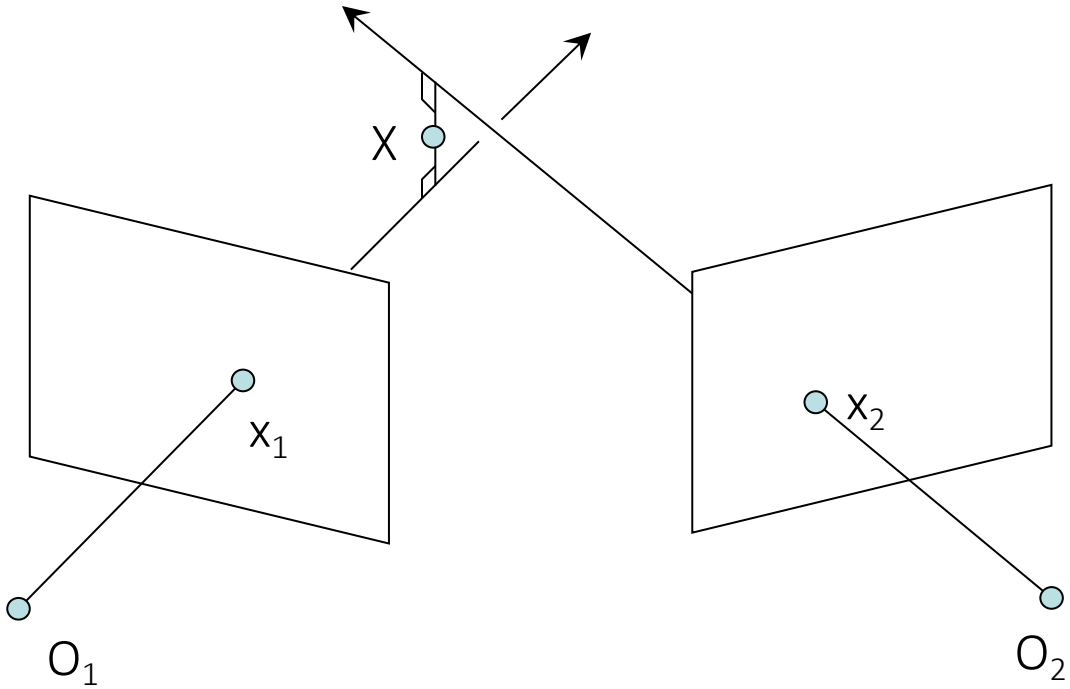
- Knowing P and P'
- Knowing $\mathbf{x}$ and $\mathbf{x}'$
- Compute x' such that

$$\mathbf{x} = \mathrm{P}\mathrm{x} \quad ; \quad \mathbf{x}' = \mathrm{P}'\mathrm{x}$$



Triangulation: Geometric Approach

- Find shortest segment connecting the two viewing rays and let X be the midpoint of that segment



Triangulation: Linear Approach

$$\lambda_1 \mathbf{x}_1 = \mathrm{P}_1 \mathrm{X} \qquad \mathbf{x}_1 \times \mathrm{P}_1 \mathrm{X} = 0 \qquad [\mathbf{x}_{1\times}]\mathrm{P}_1 \mathrm{X} = 0$$
$$\lambda_2 \mathbf{x}_2 = \mathrm{P}_2 \mathrm{X} \qquad \mathbf{x}_2 \times \mathrm{P}_2 \mathrm{X} = 0 \qquad [\mathbf{x}_{2\times}]\mathrm{P}_2 \mathrm{X} = 0$$

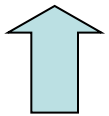
Cross product as matrix multiplication:

$$\mathbf{a} \times \mathbf{b} = \begin{bmatrix} 0 & -a_z & a_y \\ a_z & 0 & -a_x \\ -a_y & a_x & 0 \end{bmatrix} \begin{bmatrix} b_x \\ b_y \\ b_z \end{bmatrix} = [\mathbf{a}_\times]\mathbf{b}$$

Triangulation: Linear Approach

$$\lambda_1 \mathbf{x}_1 = \mathbf{P}_1 \mathbf{X} \qquad \mathbf{x}_1 \times \mathbf{P}_1 \mathbf{X} = 0 \qquad [\mathbf{x}_{1\times}]\mathbf{P}_1 \mathbf{X} = 0$$

$$\lambda_2 \mathbf{x}_2 = \mathbf{P}_2 \mathbf{X} \qquad \mathbf{x}_2 \times \mathbf{P}_2 \mathbf{X} = 0 \qquad [\mathbf{x}_{2\times}]\mathbf{P}_2 \mathbf{X} = 0$$



Two independent equations each in terms of three unknown entries of X

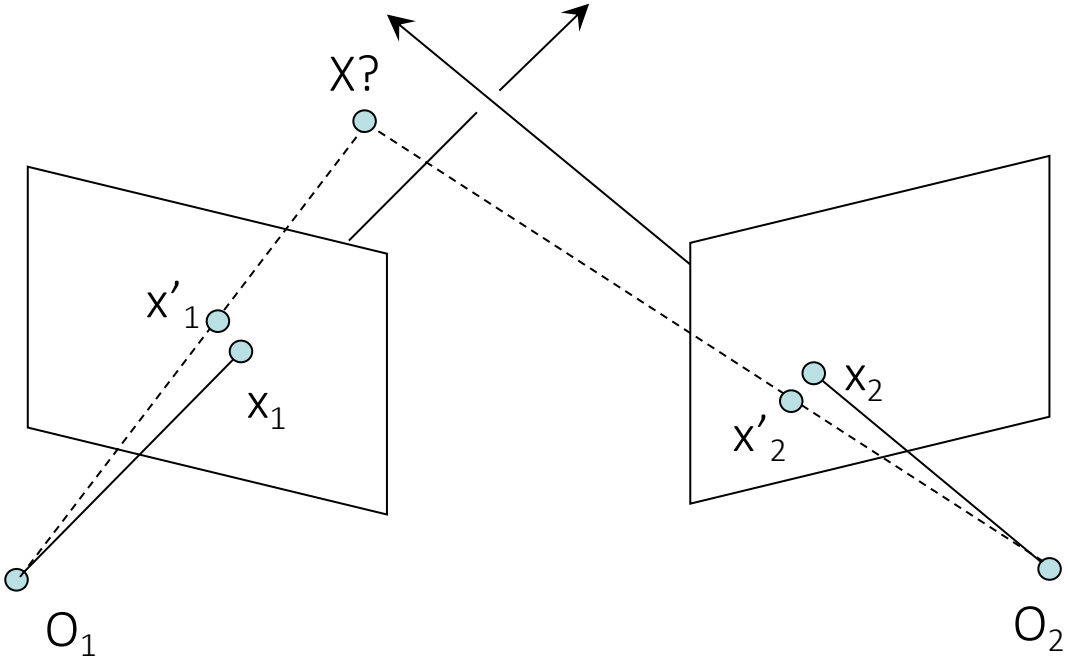
AX = 0

Eigen vector of (A^TA)

Triangulation: Nonlinear Approach

- Find X that minimizes

$$d^2(x_1, P_1X) + d^2(x_2, P_2X)$$



Two(or Multi)-view Geometry

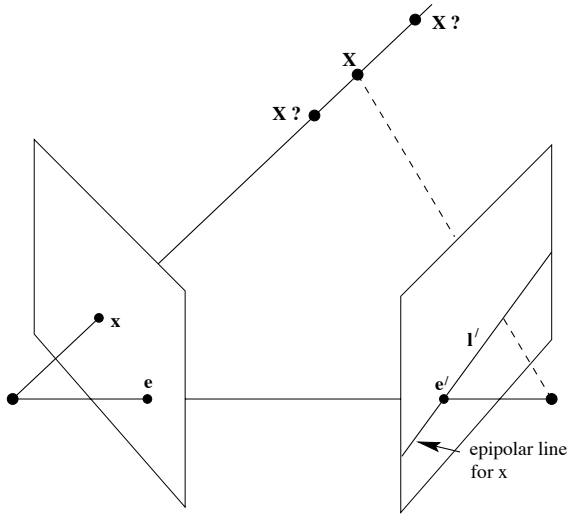
- Cameras P and P' such that

$$\mathbf{x} = \mathrm{P}\mathbf{X} \qquad \mathbf{x}' = \mathrm{P}'\mathbf{X}$$

- Baseline between the cameras is non-zero.
  - Scene geometry (structure)
    - Given projections of the same 3D point in two or more images, how do we compute the 3D coordinates of that point?
  - Correspondence (stereo matching)
    - Given a point in just one image, how does it constrain the position of the corresponding point in the second image?
  - Camera geometry (motion)
    - Given a set of corresponding points in two images, what are the cameras for the two views?

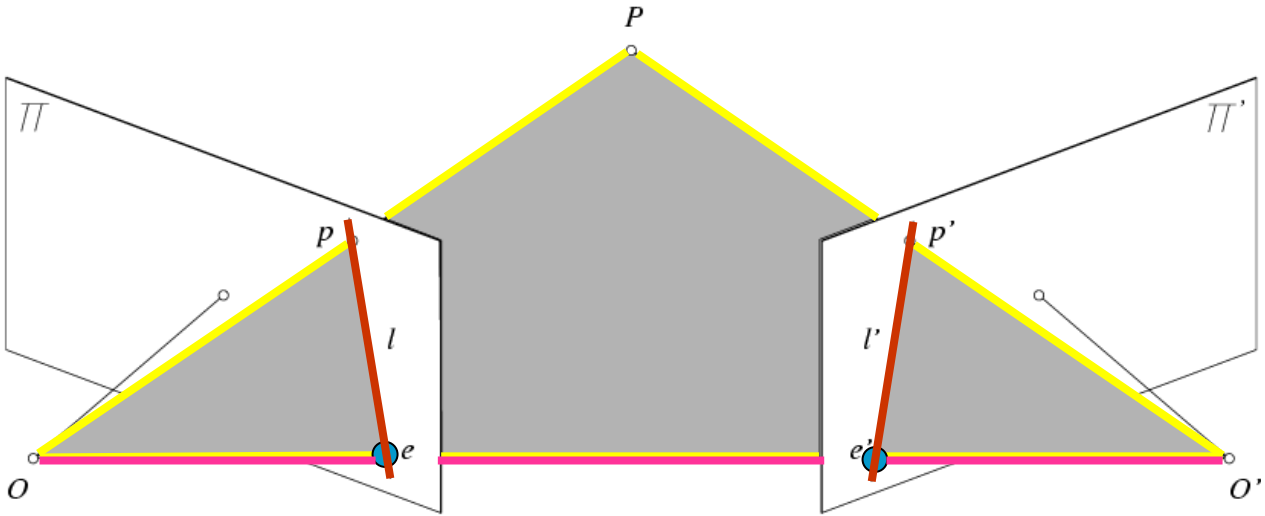
Correspondence Geometry

Given the image of a point in one view, what can we say about its position in another?



- A point in one image “generates” a line in the other image.
- This line is known as an epipolar line, and the geometry which gives rise to it is known as epipolar geometry.

Epipolar Geometry

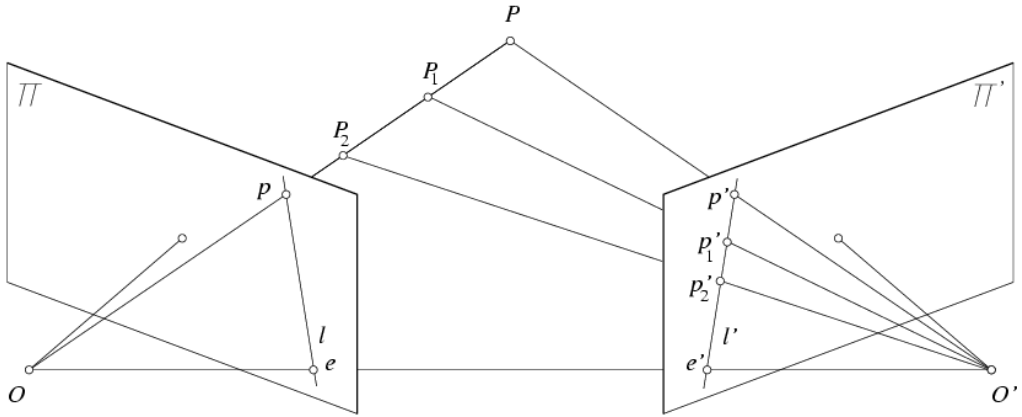


- Baseline – line connecting the two camera centers
- Epipolar Plane – plane containing baseline (1D family)
- Epipoles

= intersections of baseline with image planes
= projections of the other camera center
= vanishing points of camera motion direction

- Epipolar Lines - intersections of epipolar plane with image planes (always come in corresponding pairs)

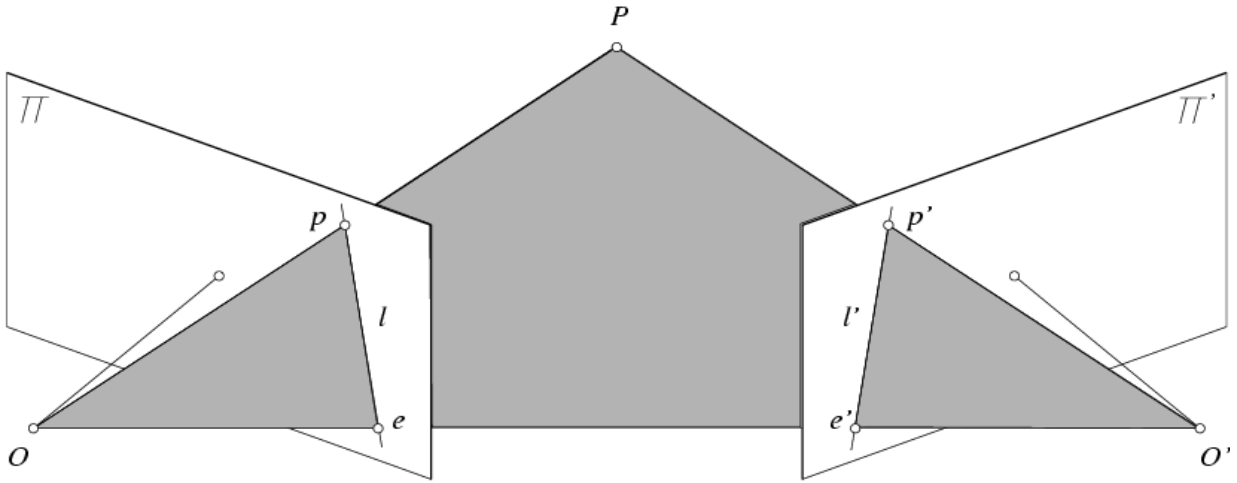
Epipolar Constraint



- If we observe a point p in one image, where can the corresponding point p' be in the other image?
  - Potential matches for *p* have to lie on the corresponding epipolar line *l'*.
  - Potential matches for *p'* have to lie on the corresponding epipolar line *l*.

[This part is from M. Pollefeys]

Epipolar Constraint: Calibrated Cameras



- Assume that the intrinsic and extrinsic parameters of the cameras are known.
- We can multiply the projection matrix of each camera (and the image points) by the inverse of the calibration matrix to get normalized image coordinates, and set the global coordinate system to the coordinate system of the first camera.

$$\hat{x} = K^{-1}x = X \qquad \hat{x}' = K'^{-1}x' = X'$$

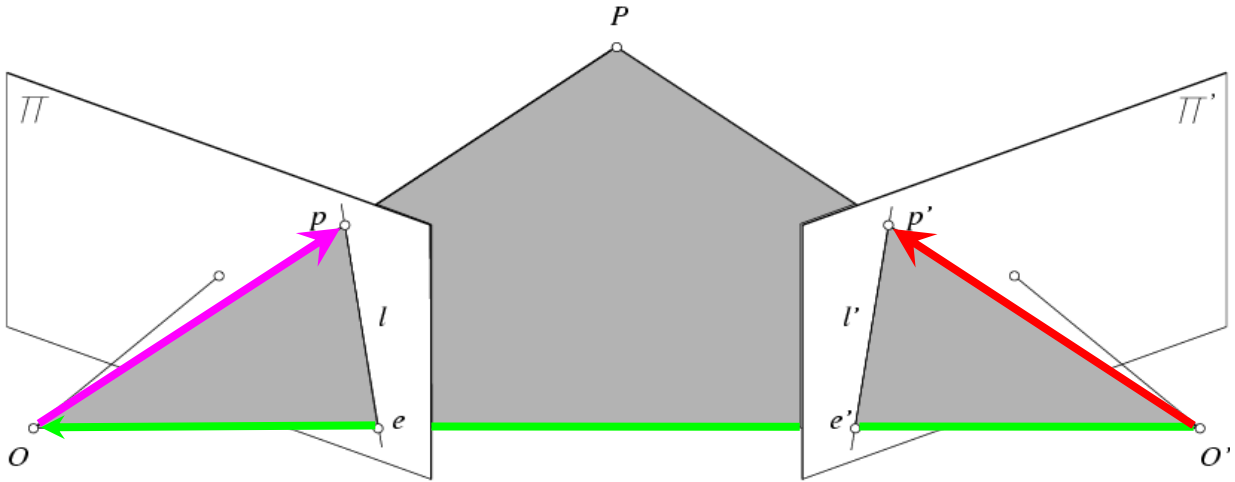
Homogeneous 2d point (3D ray towards X)

2D pixel coordinate (homogeneous)

3D scene point

3D scene point in 2nd camera's 3D coordinates

Epipolar Constraint: Calibrated Cameras

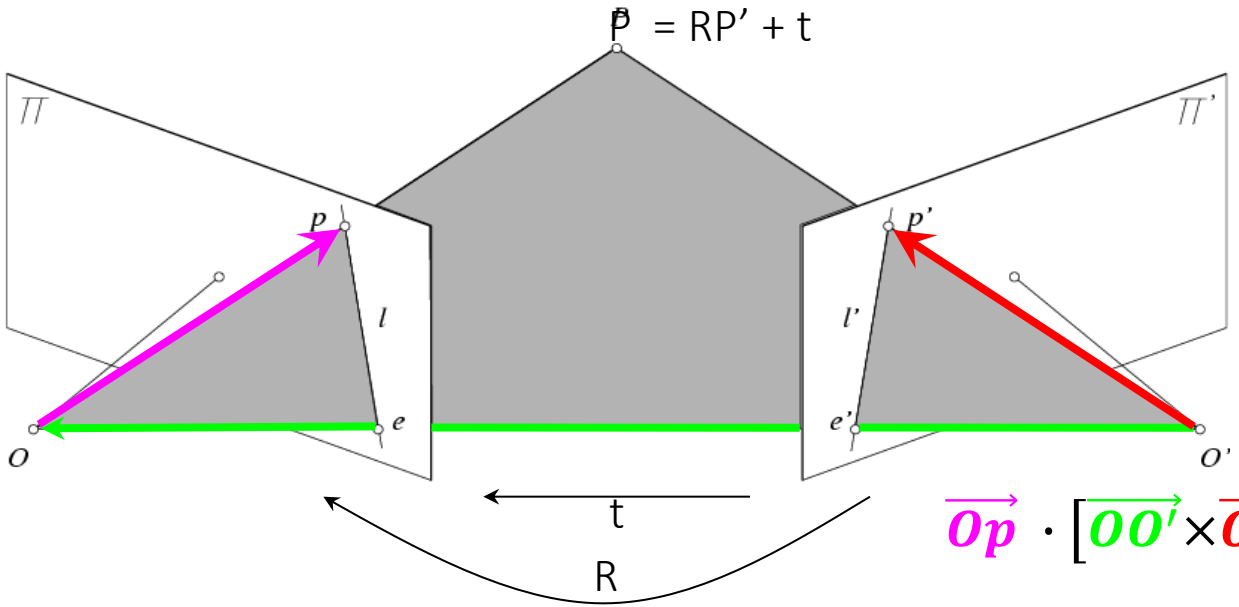


The three vectors $\overrightarrow{Op}$, $\overrightarrow{OO'}$, and $\overrightarrow{O'p'}$ are coplanar.

➡ One of them must lie in the plane spanned by the other two or

$$\overrightarrow{Op} \cdot \left[\overrightarrow{OO'} \times \overrightarrow{O'p'}\right] = 0$$

Epipolar Constraint: Calibrated Cameras



$$\overrightarrow{Op} \cdot \left[\overrightarrow{OO'} \times \overrightarrow{O'p'}\right] = 0$$

Camera matrix: [I|0]

$P = (u, v, w, 1)^T$

$p = (u, v, w)^T$

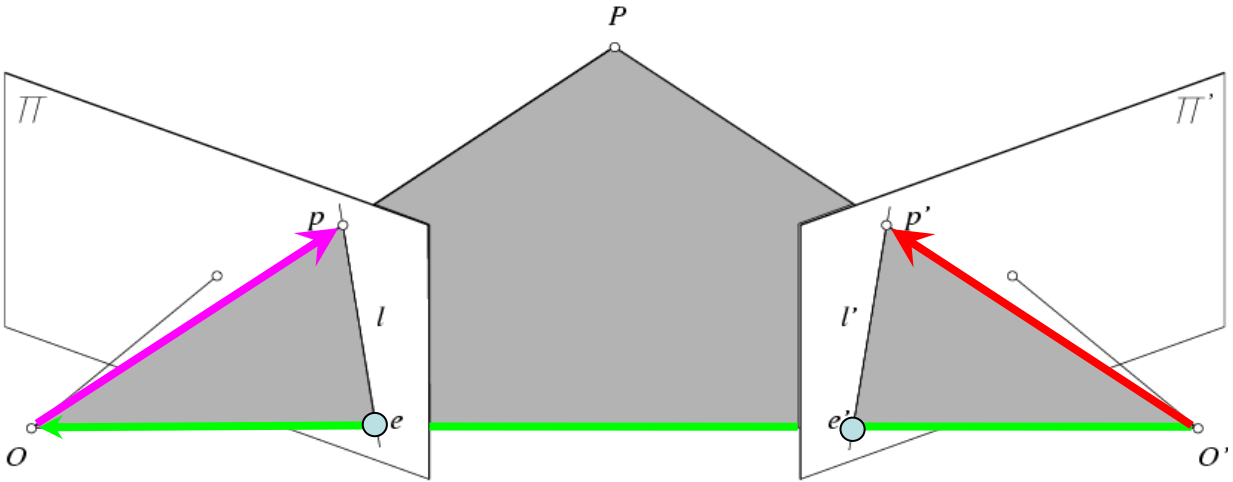
Camera matrix: $[R^T \mid -R^T t]$

- t is the coordinate vector of the translation $\overrightarrow{OO'}$
- R is the rotation matrix such that a free vector p' in the second coord. system has coordinates Rp' in the first one.*

The vectors p, t, and Rp' are coplanar.

*Refer Coordinate Free Geometry (CFG)

Epipolar Constraint: Calibrated Cameras



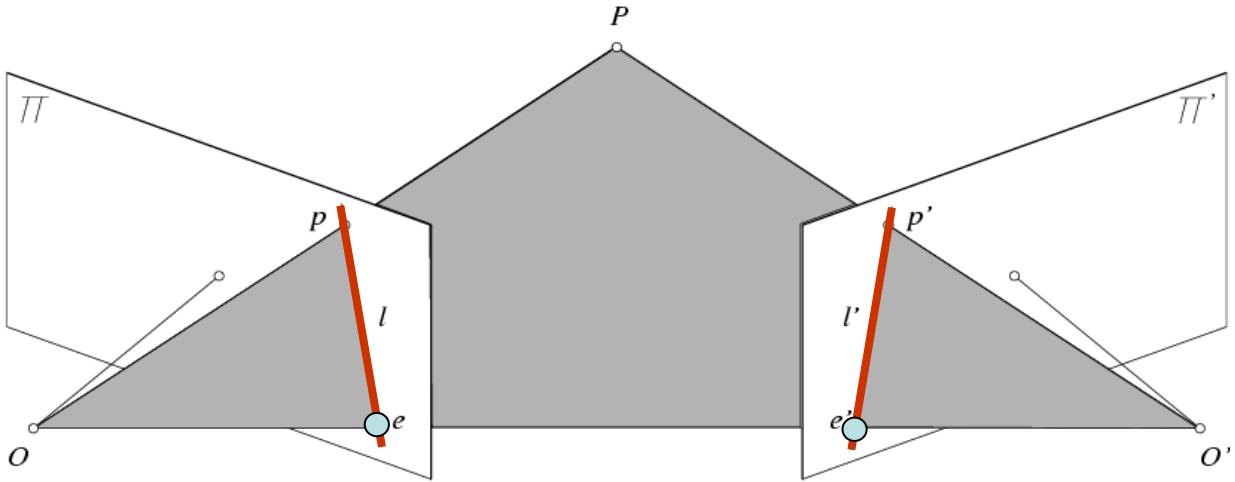
$$p \cdot [t \times (R p')] = 0 \quad \Rightarrow \quad p^T E p' = 0 \quad \text{with} \quad E = [t_\times] R$$

$$\overrightarrow{Op} \cdot [\overrightarrow{OO'} \times \overrightarrow{O'p'}] = 0$$

Essential Matrix
(Longuet-Higgins, 1981)

The vectors p, t, and Rp' are coplanar.

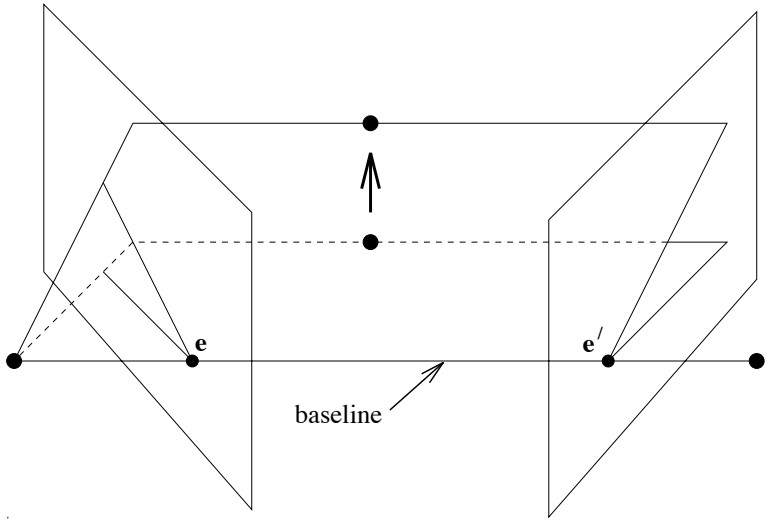
Epipolar Constraint: Calibrated Cameras



$$p \cdot [t \times (R\,p')] = 0 \quad \Rightarrow \quad p^T E\, p' = 0 \quad \text{with} \quad E = [t_\times]R$$

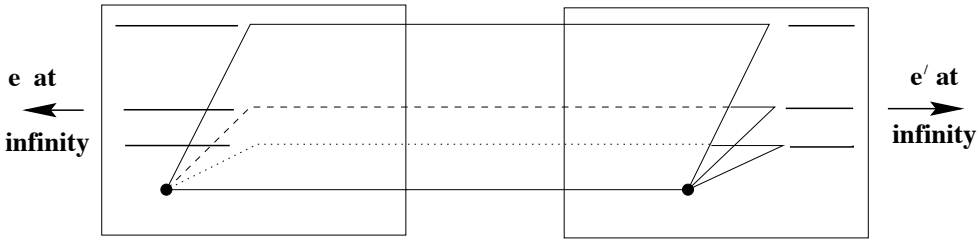
- $E\,p'$ is the epipolar line associated with p' ($l = E\,p'$).
- $E^T p$ is the epipolar line associated with p ($l' = E^T p$).
- $E\,e' = 0$ and $E^T e = 0$.
- E is singular (rank two).
- E has five degrees of freedom.

Epipolar pencil



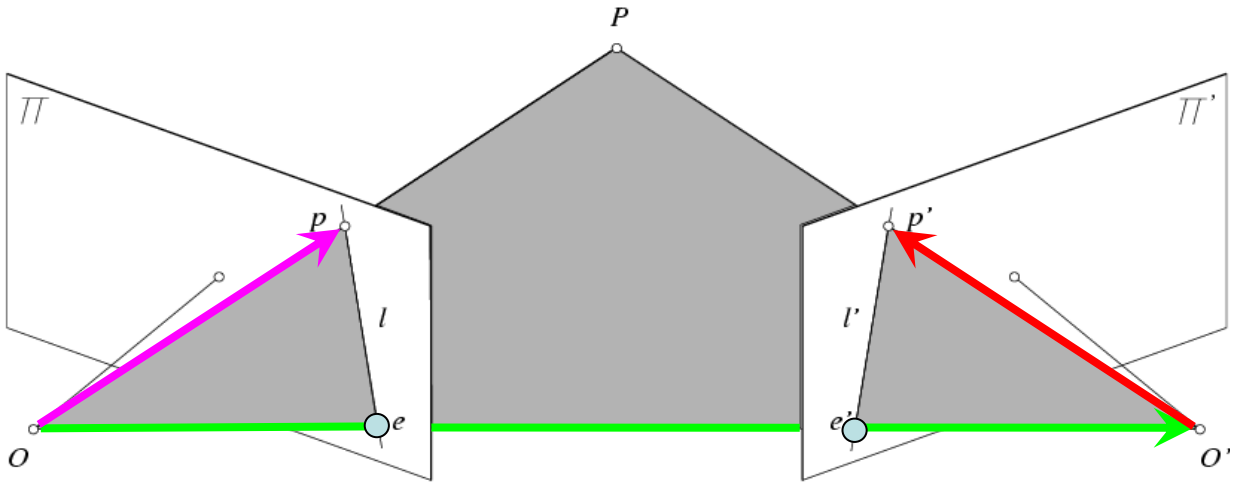
As the position of the 3D point $\mathbf{X}$ varies, the epipolar planes "rotate" about the baseline. This family of planes is known as an epipolar pencil. All epipolar lines intersect at the epipole.

Epipolar geometry example



Epipolar geometry depends only on the relative pose (position and orientation) and internal parameters of the two cameras, i.e. the position of the camera centres and image planes. It does not depend on structure (3D points external to the camera).

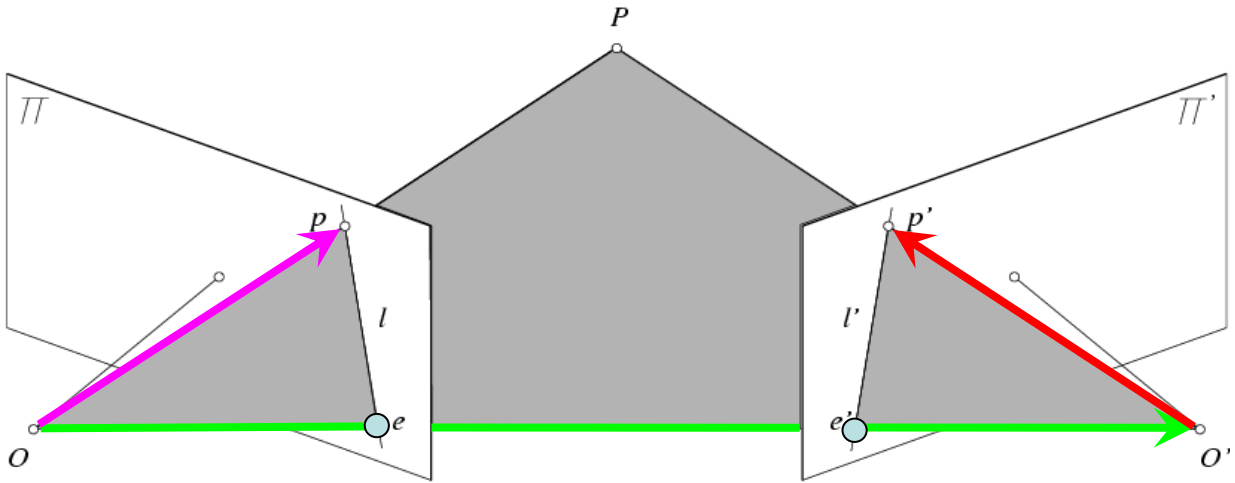
Epipolar Constraint: Uncalibrated Cameras



- The calibration matrices *K* and *K'* of the two cameras are unknown
- We can write the epipolar constraint in terms of *unknown* normalized coordinates:

$$p^T E p' = 0 \qquad \hat{p} = Kp \text{ and } \hat{p}' = Kp'$$

Epipolar Constraint: Uncalibrated Cameras

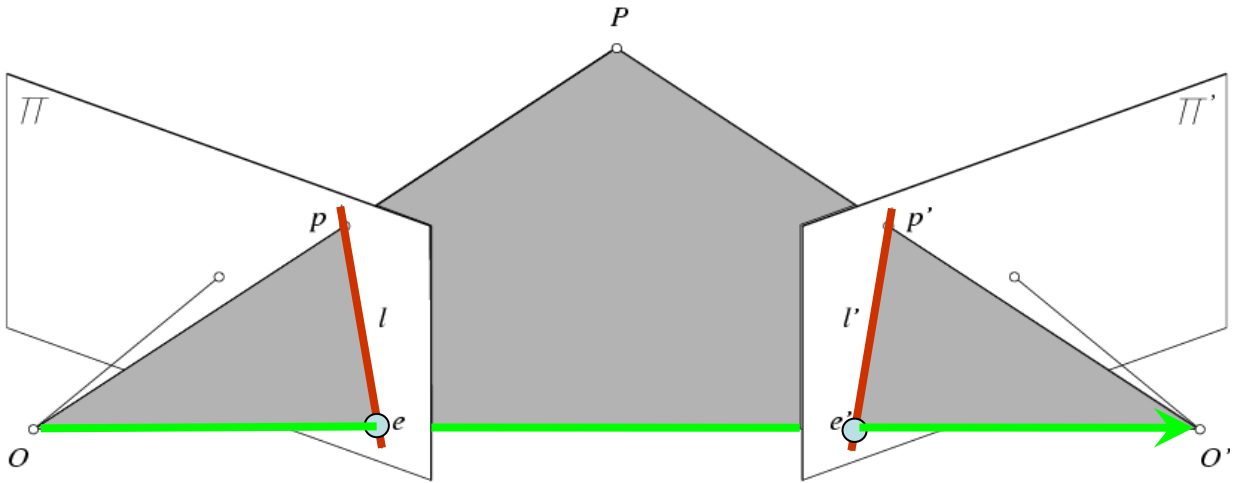


$$p^T E p' = 0 \Rightarrow \hat{p}^T F \hat{p}' = 0 \text{ with } F = K^{-T} E K'^{-1}$$

$$\hat{p} = Kp \,, \hat{p}' = Kp'$$

Fundamental Matrix
(Faugeras and Luong, 1992)

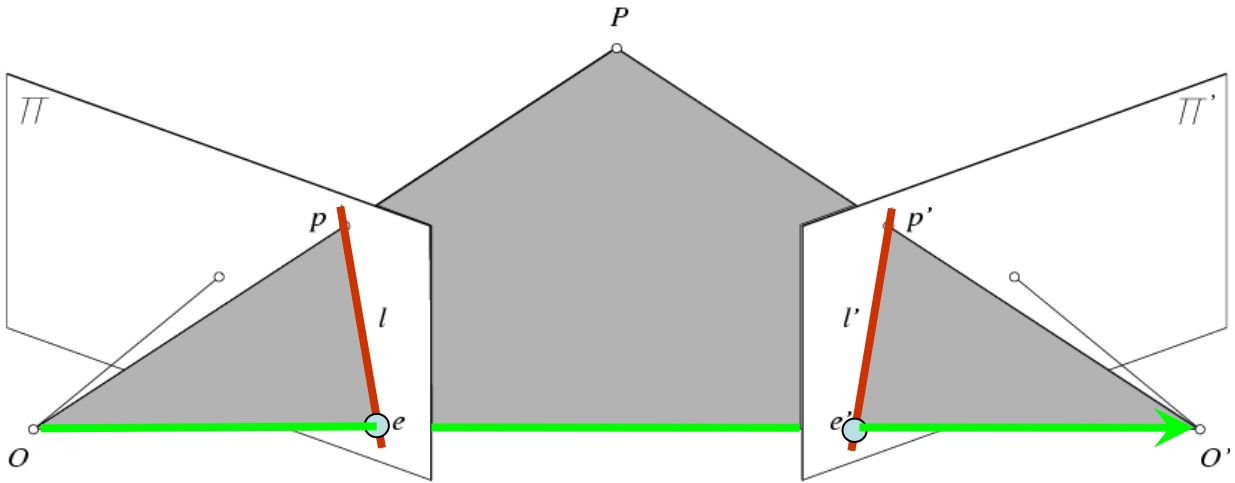
Epipolar Constraint: Uncalibrated Cameras



$$p^T E p' = 0 \Rightarrow \hat{p}^T F \hat{p}' = 0 \text{ with } F = K^{-T} E K'^{-1}$$

- $F\,\hat{p}'$ is the epipolar line associated with $\hat{p}'$ ($l = F\,\hat{p}'$).
- $F^T\,\hat{p}$ is the epipolar line associated with $\hat{p}$ ($l' = F^T\,\hat{p}$).
- $F\,e' = 0$ and $F^T e = 0$.
- F is singular (rank two).
- F has seven degrees of freedom.

Epipolar Constraint: Uncalibrated Cameras



$$p^T E p' = 0 \Rightarrow \hat{p}^T F \hat{p}' = 0 \text{ with } F = K^{-T} E K'^{-1}$$

- $F\,\hat{p}'$ is the epipolar line associated with $\hat{p}'$ ($l = F\,\hat{p}'$).
- $F^T\,\hat{p}$ is the epipolar line associated with $\hat{p}$ ($l' = F^T\,\hat{p}$).
- $F\,e' = 0$ and $F^T e = 0$.
- F is singular (rank two).
- F has seven degrees of freedom.

Computation of the Fundamental Matrix

Basic equations

Given a correspondence

$$\mathbf{x} \leftrightarrow \mathbf{x}'$$

The basic incidence relation is

$$\mathbf{x}'^{\top}\mathbf{F}\mathbf{x} = 0$$

May be written

$$x'xf_{11} + x'yf_{12} + x'f_{13} + y'xf_{21} + y'yf_{22} + y'f_{23} + xf_{31} + yf_{32} + f_{33} = 0 \quad .$$

Single point equation - Fundamental matrix

Gives an equation :

$$(x'x, x'y, x', y'x, y'y, y', x, y, 1) \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} = 0$$

where

$$\mathbf{f} = (f_{11}, f_{12}, f_{13}, f_{21}, f_{22}, f_{23}, f_{31}, f_{32}, f_{33})^\top$$

holds the entries of the Fundamental matrix

Total set of equations

$$\mathbf{Af} = \begin{bmatrix} x'_1x_1 & x'_1y_1 & x'_1 & y'_1x_1 & y'_1y_1 & y'_1 & x_1 & y_1 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \\ x'_nx_n & x'_ny_n & x'_n & y'_nx_n & y'_ny_n & y'_n & x_n & y_n & 1 \end{bmatrix} \begin{pmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{pmatrix} = \mathbf{0}$$

Solving the Equations

- Solution is determined up to scale only.
- Need 8 equations $\Rightarrow$ 8 points
- 8 points $\Rightarrow$ unique solution
- > 8 points $\Rightarrow$ least-squares solution.

Least-squares solution

(i) Form equations $\mathtt{A}\mathbf{f} = \mathbf{0}$.

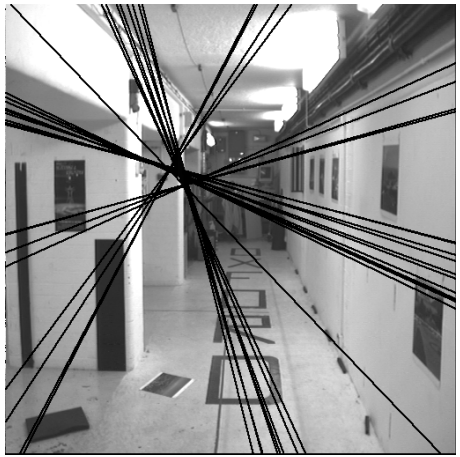
(ii) Take SVD : $\mathtt{A} = \mathtt{U}\mathtt{D}\mathtt{V}^{\top}$.

(iii) Solution is last column of $\mathtt{V}$ (corresp : smallest singular value)

(iv) Minimizes $||\mathtt{A}\mathbf{f}||$ subject to $||\mathbf{f}|| = 1$.

The singularity constraint

Fundamental matrix has rank 2 : $\det(\mathtt{F}) = 0$.



Left : Uncorrected $\mathtt{F}$ – epipolar lines are not coincident.

Right : Epipolar lines from corrected $\mathtt{F}$.

Computing F Matrix – Eight-point Algorithm

- Eight-point algorithm (Longuet-Higgins, 1981)
  - F should have rank 2
  - To enforce that F is of rank 2, F is replaced by F' that minimizes $\|\mathbf{F}-\mathbf{F}'\|$ subject to the rank constraint.
  - This is achieved by SVD. Let $\mathbf{F}=\mathbf{U}\Sigma\mathbf{V}^{\mathrm{T}}$, where

$$\Sigma=\begin{bmatrix}\sigma_1 & 0 & 0\\ 0 & \sigma_2 & 0\\ 0 & 0 & \sigma_3\end{bmatrix}, \text{ let } \quad \Sigma'=\begin{bmatrix}\sigma_1 & 0 & 0\\ 0 & \sigma_2 & 0\\ 0 & 0 & 0\end{bmatrix}$$

then $\mathbf{F}'=\mathbf{U}\Sigma'\mathbf{V}^{\mathrm{T}}$ is the solution.

Computing F Matrix – Eight-point Algorithm

```
% Build the constraint 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(npts,1) ];

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

% Extract fundamental matrix from the column of V
% corresponding to the smallest singular value.
F = reshape(V(:,9),3,3)';

% Enforce rank2 constraint
[U,D,V] = svd(F);
F = U*diag([D(1,1) D(2,2) 0])*V';
```

Computing F Matrix – Normalized Eight-Point Algorithm

- The Normalized Eight-Point Algorithm (Hartley, 1995)
  - Problem of eight-point algorithm
    - Poor numerical conditioning
    - Can be fixed by rescaling the data

$$\begin{bmatrix} u_1u_1' & v_1u_1' & u_1' & u_1v_1' & v_1v_1' & v_1' & u_1 & v_1 & 1 \\ u_2u_2' & v_2u_2' & u_2' & u_2v_2' & v_2v_2' & v_2' & u_2 & v_2 & 1 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ u_nu_n' & v_nu_n' & u_n' & u_nv_n' & v_nv_n' & v_n' & u_n & v_n & 1 \end{bmatrix} \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0$$

~10000 ~1000 ~100 ~10000 ~10000 ~100 ~100 ~100 1

0

Orders of magnitude difference
between column of data matrix
→ least-squares yields poor results

The normalized 8-point algorithm

Raw 8-point algorithm performs badly in presence of noise.

Normalization of data

- 8-point algorithm is sensitive to origin of coordinates and scale.
- Data must be translated and scaled to "canonical" coordinate frame.
- Normalizing transformation is applied to both images.
- Translate so centroid is at origin
- Scale so that RMS distance of points from origin is $\sqrt{2}$.
- "Average point" is $(1, 1, 1)^\top$.

Normalized 8-point algorithm

(i) **Normalization:** Transform the image coordinates :

$$\hat{\mathbf{x}}_i = \mathtt{T}\mathbf{x}_i$$
$$\hat{\mathbf{x}}'_i = \mathtt{T}'\mathbf{x}'_i$$

(ii) **Solution:** Compute $\mathtt{F}$ from the matches $\hat{\mathbf{x}}_i \leftrightarrow \hat{\mathbf{x}}'_i$

$$\hat{\mathbf{x}}_i'^{\top}\widehat{\mathtt{F}}\hat{\mathbf{x}}_i = 0$$

(iii) **Singularity constraint :** Find closest singular $\widehat{\mathtt{F}}'$ to $\widehat{\mathtt{F}}$.

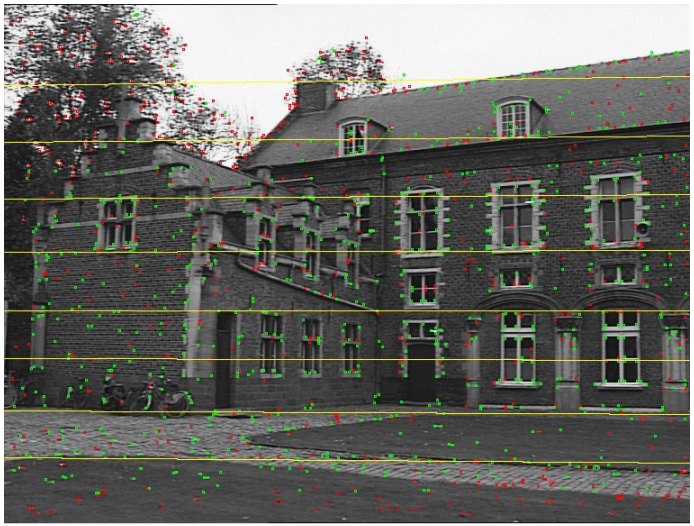
(iv) **Denormalization:** $\mathtt{F} = \mathtt{T}'^{\top}\widehat{\mathtt{F}}'\mathtt{T}$.

Computing F Matrix – Normalized Eight-Point Algorithm

```
[x1, T1] = normalise2dpts(x1);
[x2, T2] = normalise2dpts(x2);

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(npts,1) ]
   ;

[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';

% Denormalise
F = T2'*F*T1;
```

KAIST

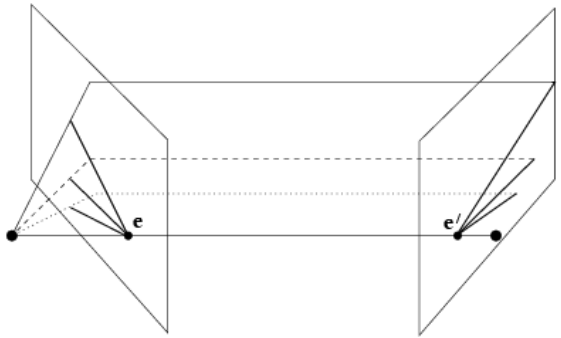
Examples of Epipolar Geometry



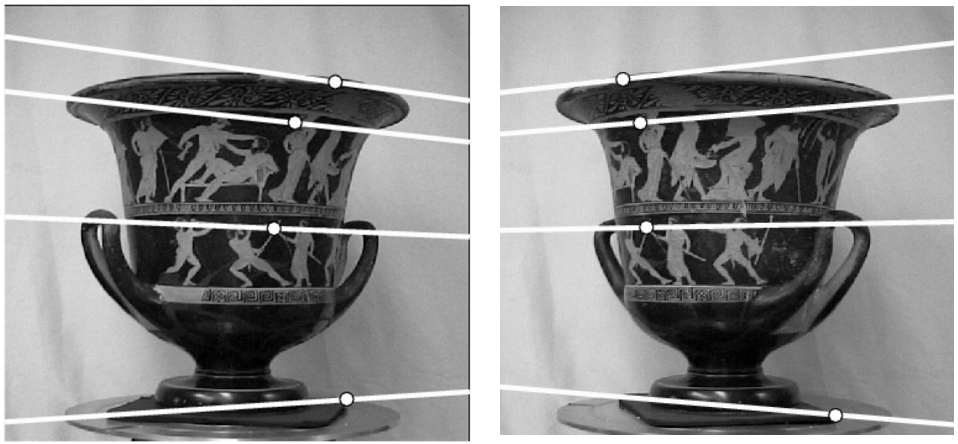
slides by Pollefeys

Examples of Epipolar Geometry

- Converging cameras



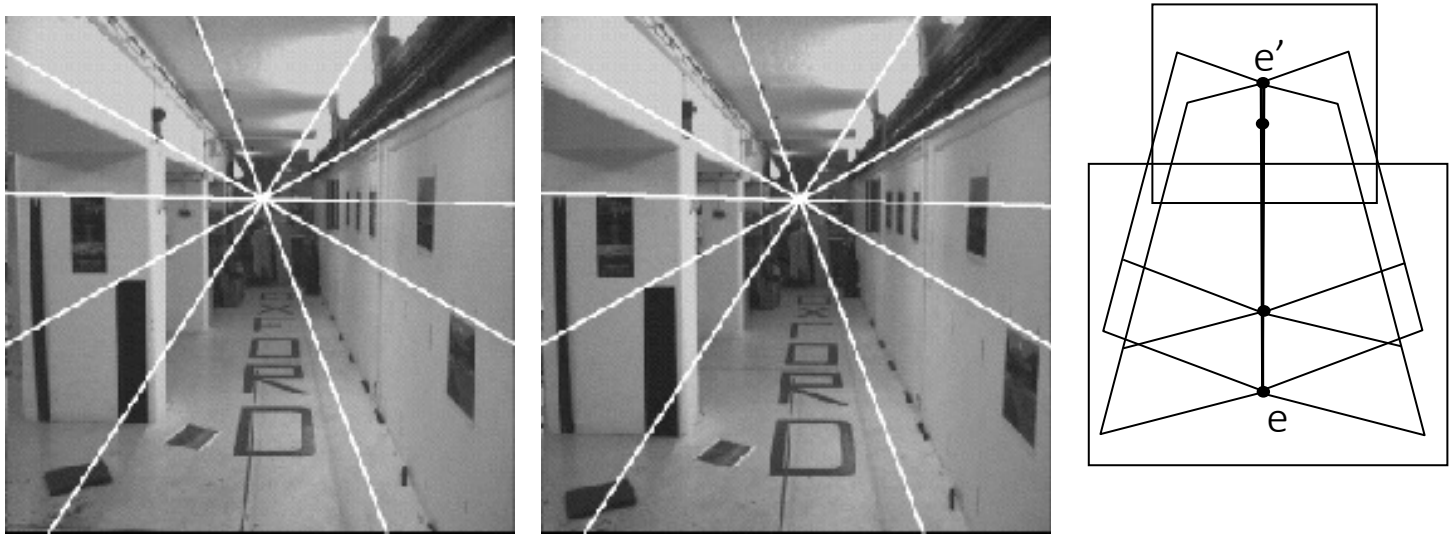
courtesy of Andrew Zisserman



slides by Pollefeys

Examples of Epipolar Geometry

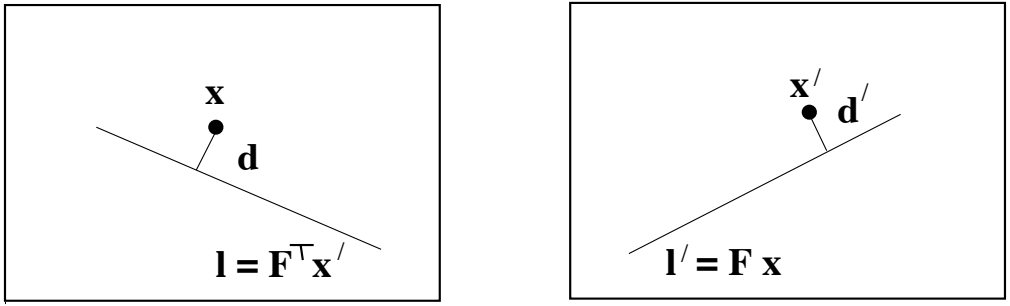
- Forward motion



Epipole has same coordinates in both images.
Points move along lines radiating from e: “Focus of expansion”

courtesy of Andrew Zisserman

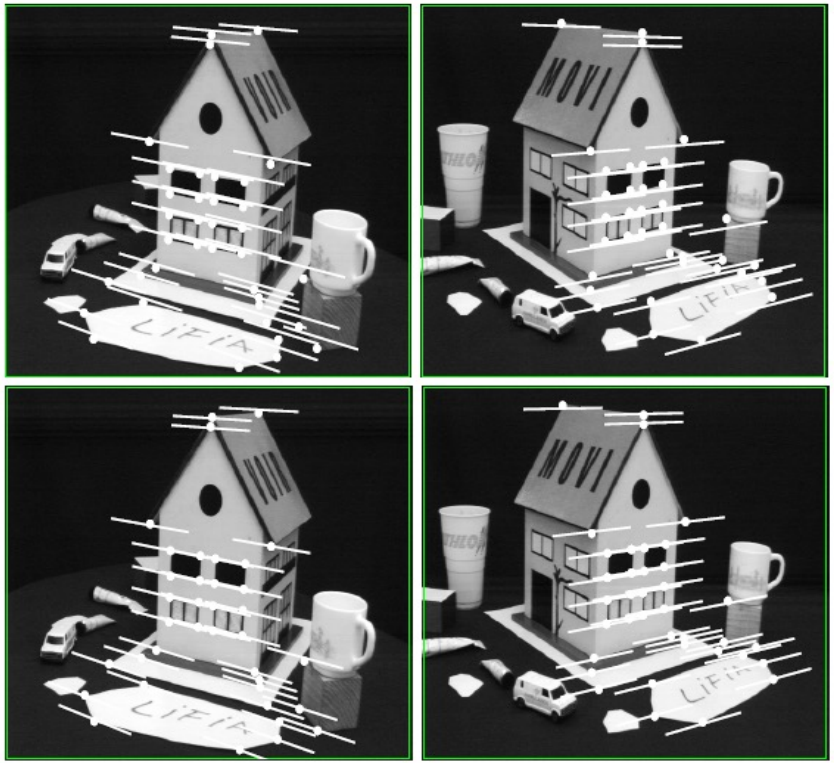
Epipolar distance



- Point correspondence $\mathbf{x}' \leftrightarrow \mathbf{x}$:
- Point $\mathbf{x} \mapsto$ epipolar line $\mathtt{F}\mathbf{x}$.
- Epipolar distance is distance of point $\mathbf{x}'$ to epipolar line $\mathtt{F}\mathbf{x}$.
- Write $\mathtt{F}\mathbf{x} = (\lambda, \mu, \nu)^\top$ and $\mathbf{x}' = (x', y', 1)^\top$.
- Distance is

$$d(\mathbf{x}', \mathtt{F}\mathbf{x}) = \mathbf{x}'^\top \mathtt{F}\mathbf{x}(\lambda^2 + \mu^2)^{-1/2}$$

Comparison of Estimation Algorithms



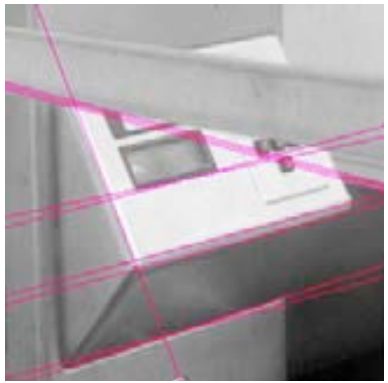
	8-point	Normalized 8-point	Nonlinear least squares
Av. Dist. 1	2.33 pixels	0.92 pixel	0.86 pixel
Av. Dist. 2	2.18 pixels	0.85 pixel	0.80 pixel

Fundamental Matrix Song

https://www.youtube.com/watch?v=DgGV3l82NTk

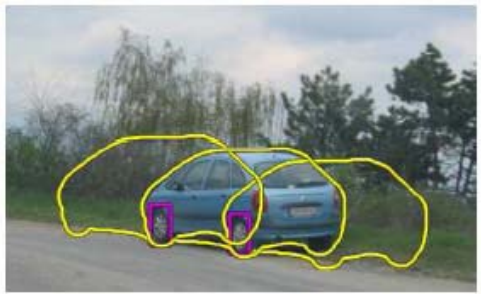
Fitting

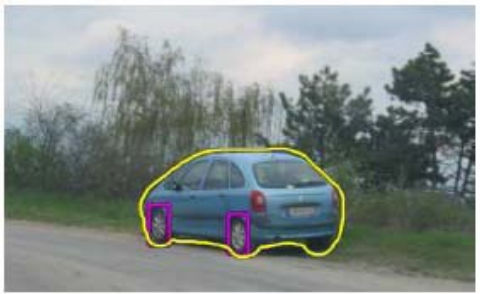
- Choose a parametric model to represent a set of features



simple model: lines

simple model: circles





complicated model: car

KAIST

Source: K. Grauman

Fitting: Issues

Case study: Line detection

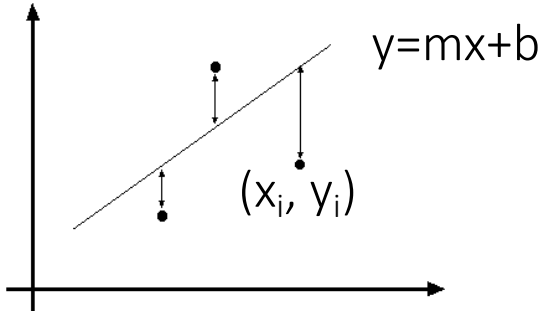
- **Noise** in the measured feature locations
- **Extraneous data:** clutter (outliers), multiple lines
- **Missing data:** occlusions

Fitting: Overview

- If we know which points belong to the line, how do we find the "optimal" line parameters?
  - Least squares
- What if there are outliers?
  - Robust fitting, RANSAC
- What if there are many lines?
  - Voting methods: RANSAC, Hough transform
- What if we're not even sure it's a line?
  - Model selection

Least Squares Line Fitting

- Data: $(x_1, y_1), \ldots, (x_n, y_n)$
- Line equation: $y_i = m x_i + b$
- Find (m, b) to minimize



$$E = \sum_{i=1}^{n} (y_i - m x_i - b)^2$$

$$Y = \begin{bmatrix} y_1 \\ \vdots \\ y_n \end{bmatrix} \qquad X = \begin{bmatrix} x_1 & 1 \\ \vdots & \vdots \\ x_n & 1 \end{bmatrix} \qquad B = \begin{bmatrix} m \\ b \end{bmatrix}$$

$$E = \|Y - XB\|^2 = (Y - XB)^T (Y - XB) = Y^T Y - 2(XB)^T Y + (XB)^T (XB)$$

$$\frac{dE}{dB} = 2X^T XB - 2X^T Y = 0$$

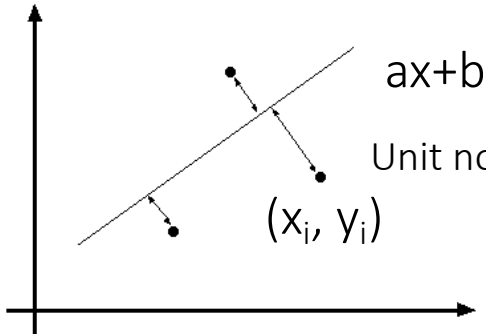
$$X^T XB = X^T Y$$ Normal equations: least squares solution to XB=Y

Problem with “Vertical” Least Squares

- Not rotation-invariant
- Fails completely for vertical lines

Total Least Squares

- Distance between point (x_i, y_i) and line $ax+by=d$ $(a^2+b^2=1)$: $|ax_i + by_i - d|$
- Find (a, b, d) to minimize the sum of squared perpendicular distances



$$E = \sum_{i=1}^{n} (ax_i + by_i - d)^2$$

$$\frac{\partial E}{\partial d} = \sum_{i=1}^{n} -2(ax_i + by_i - d) = 0$$

$$d = \frac{a}{n}\sum_{i=1}^{n} x_i + \frac{b}{n}\sum_{i=1}^{n} y_i = a\bar{x} + b\bar{y}$$

$$E = \sum_{i=1}^{n} (a(x_i - \bar{x}) + b(y_i - \bar{y}))^2 = \left\| \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix} \begin{bmatrix} a \\ b \end{bmatrix} \right\|^2 = (UN)^T (UN)$$

$$\frac{dE}{dN} = 2(U^T U)N = 0$$

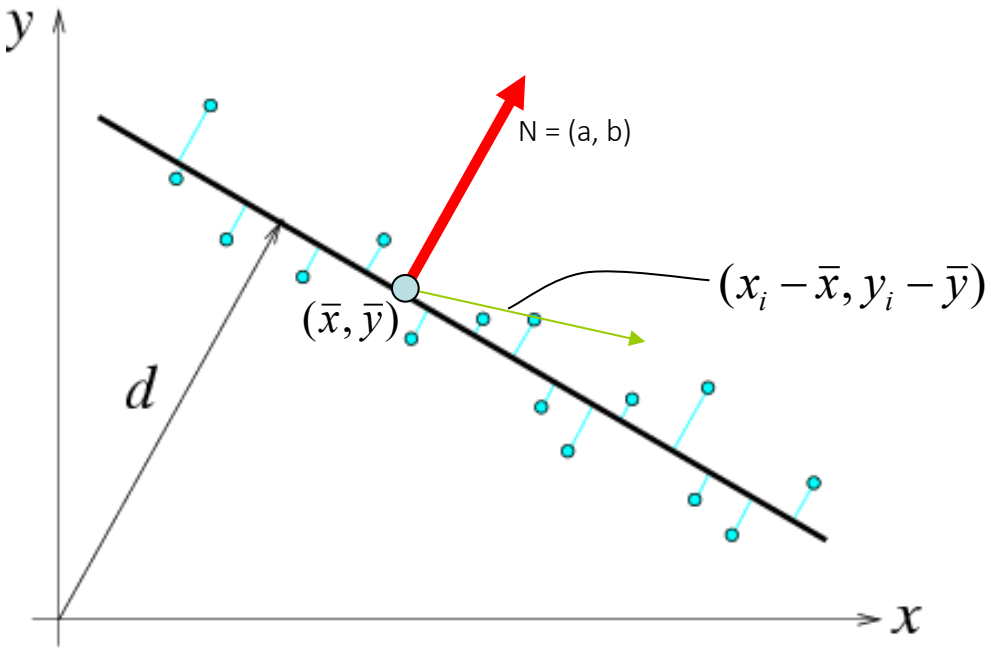
Solution to $(U^TU)N = 0$, subject to $||N||^2 = 1$: eigenvector of U^TU associated with the smallest eigenvalue (least squares solution to homogeneous linear system $UN = 0$)

Total Least Squares

$$U = \begin{bmatrix} x_1 - \bar{x} & y_1 - \bar{y} \\ \vdots & \vdots \\ x_n - \bar{x} & y_n - \bar{y} \end{bmatrix}$$

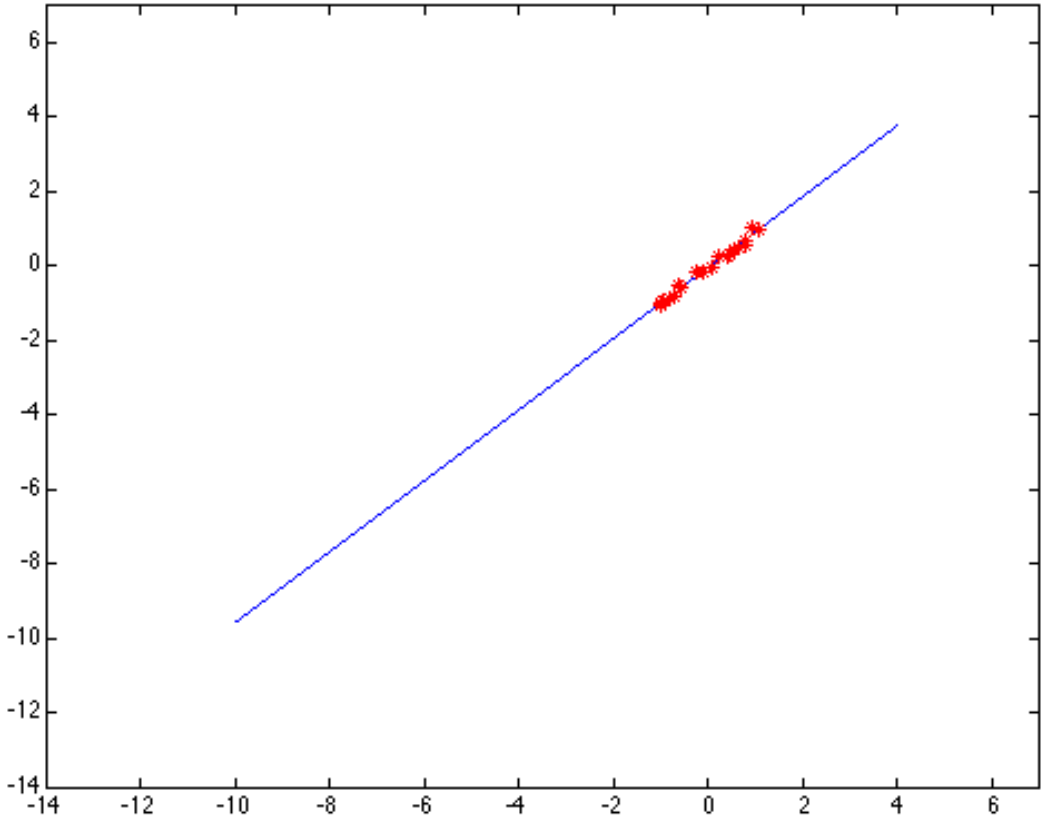
$$U^T U = \begin{bmatrix} \sum_{i=1}^{n} (x_i - \bar{x})^2 & \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) \\ \sum_{i=1}^{n} (x_i - \bar{x})(y_i - \bar{y}) & \sum_{i=1}^{n} (y_i - \bar{y})^2 \end{bmatrix}$$

second moment matrix



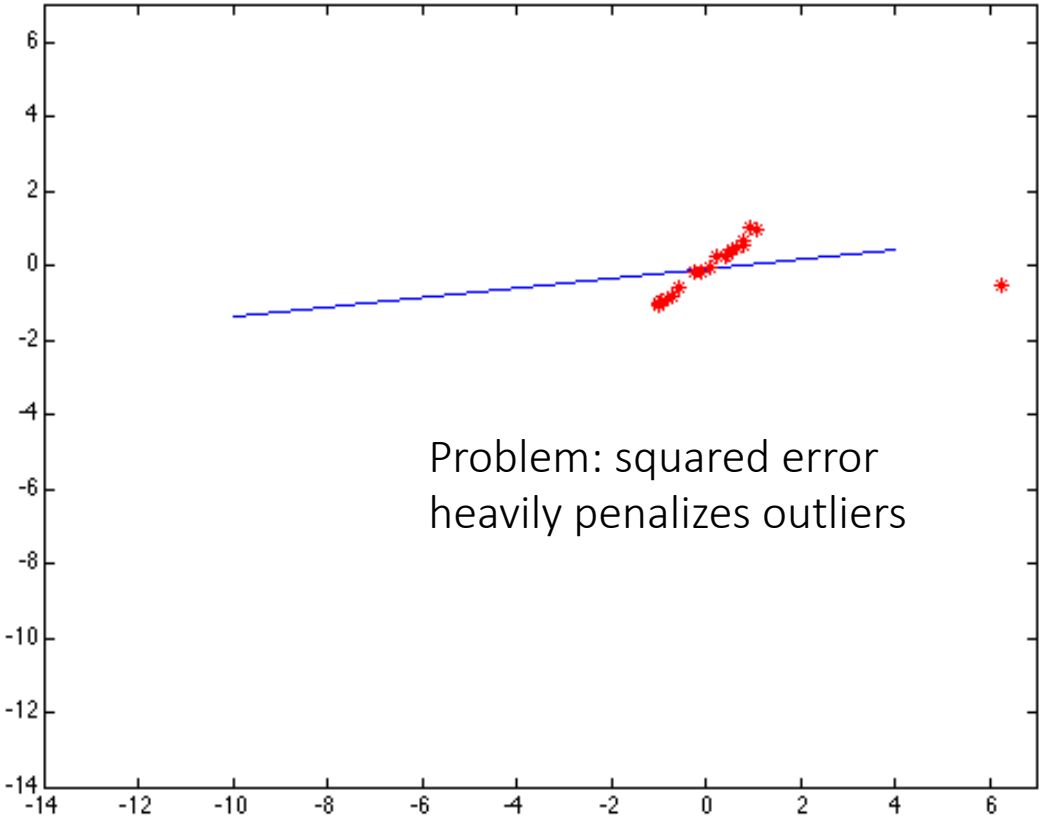
Least Squares: Robustness to Noise

- Least squares fit to the red points:



Least Squares: Robustness to Noise

- Least squares fit to the red points:



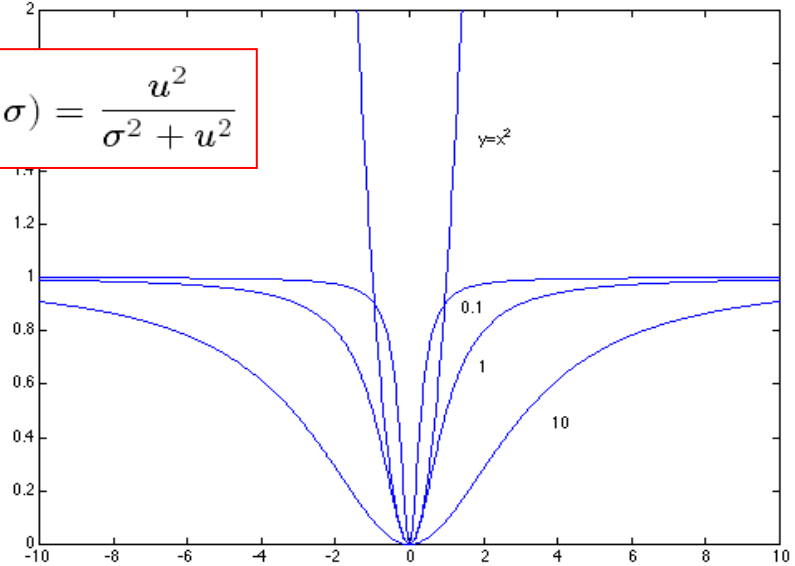
Robust Estimators

- General approach: find model parameters ϑ that minimize

$$\sum_i \rho\left(r_i\left(x_i, \theta\right); \sigma\right)$$

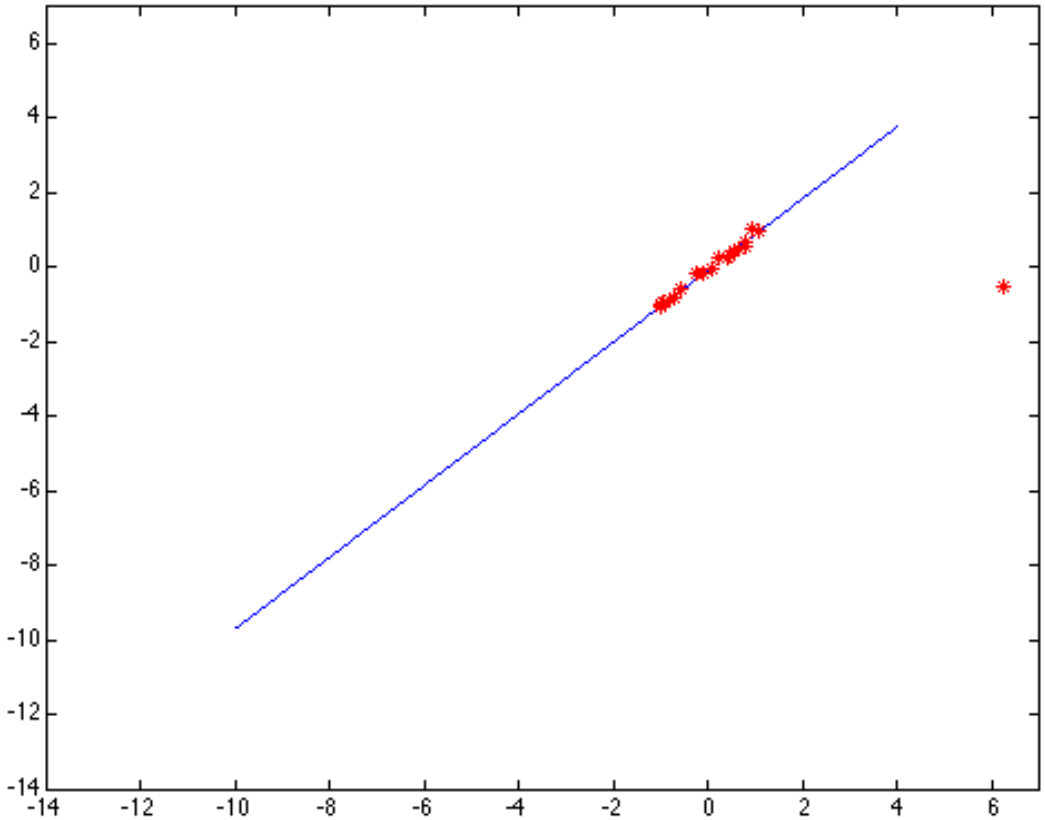
$r_i(x_i, \vartheta)$ – residual of i_{th} point w.r.t. model parameters ϑ
ρ – robust function with scale parameter σ

$$\rho(u;\sigma) = \frac{u^2}{\sigma^2 + u^2}$$



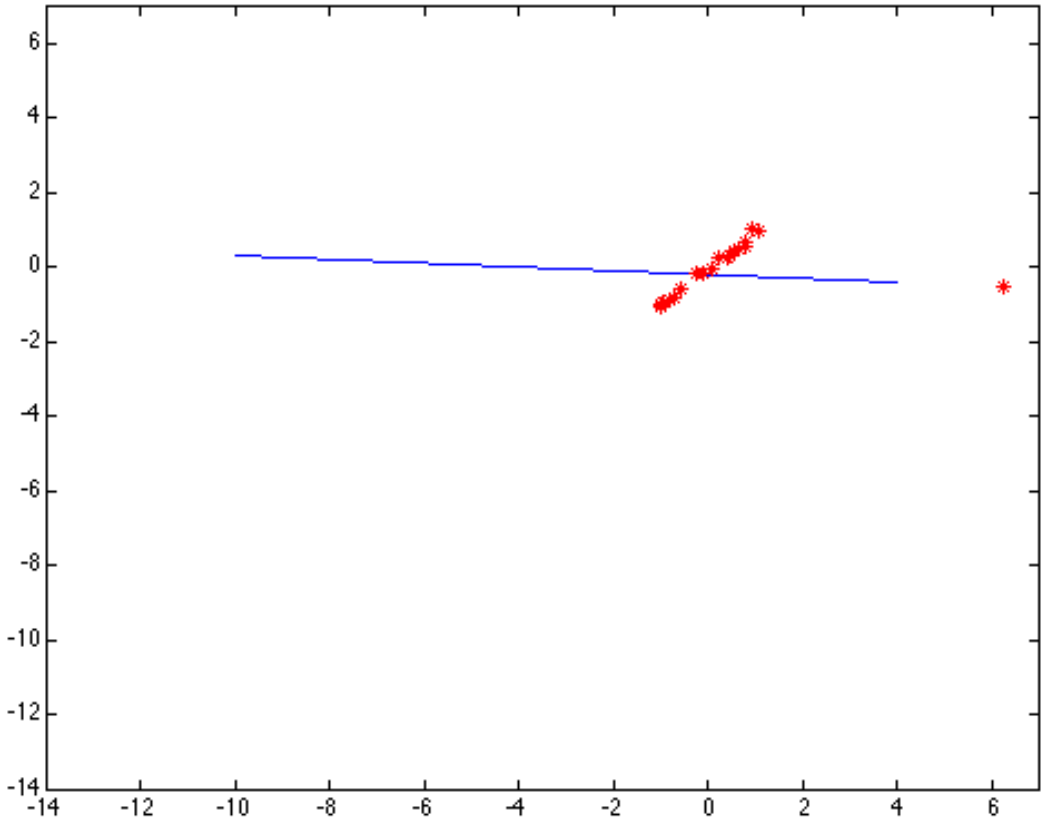
The robust function ρ behaves like squared distance for small values of the residual u but saturates for larger values of u

Choosing the Scale: Just Right



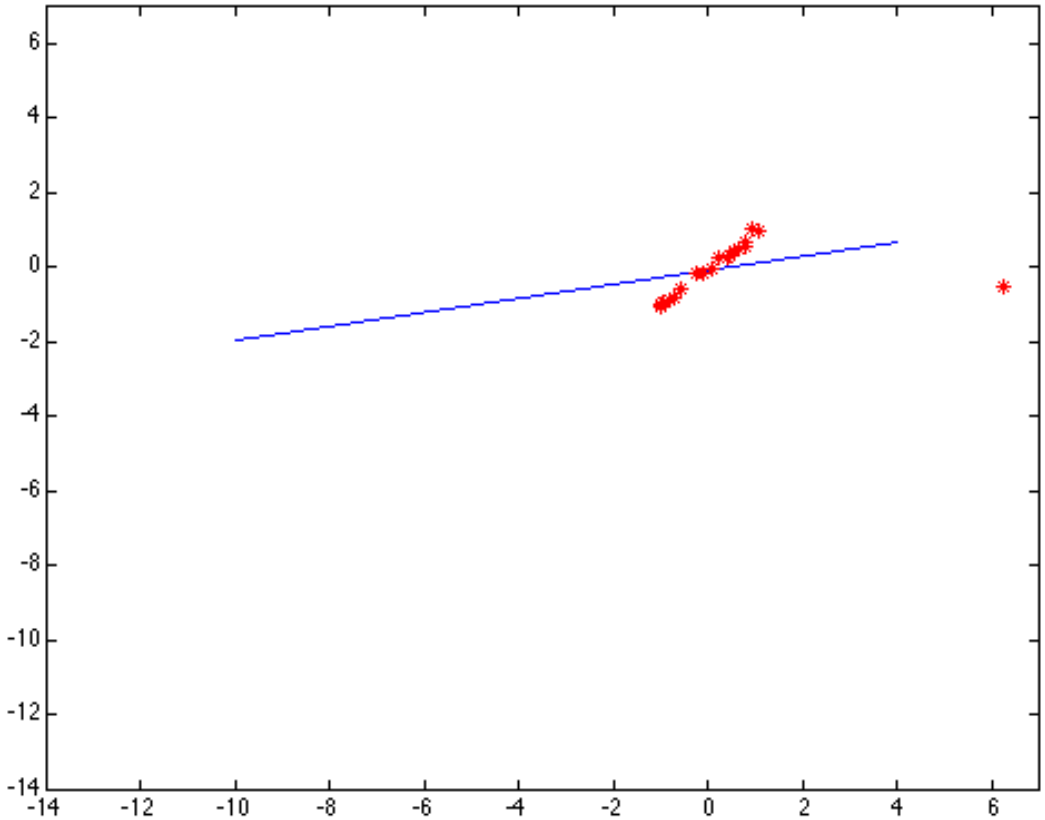
The effect of the outlier is minimized

Choosing the Scale: Too small



The error value is almost the same for every point and the fit is very poor

Choosing the Scale: Too large



Behaves much the same as least squares

Robust Estimation: Details

- Robust fitting is a nonlinear optimization problem that must be solved iteratively
- Least squares solution can be used for initialization
- Adaptive choice of scale: approx. 1.5 times median residual (F&P, Sec. 15.5.1)

RANSAC

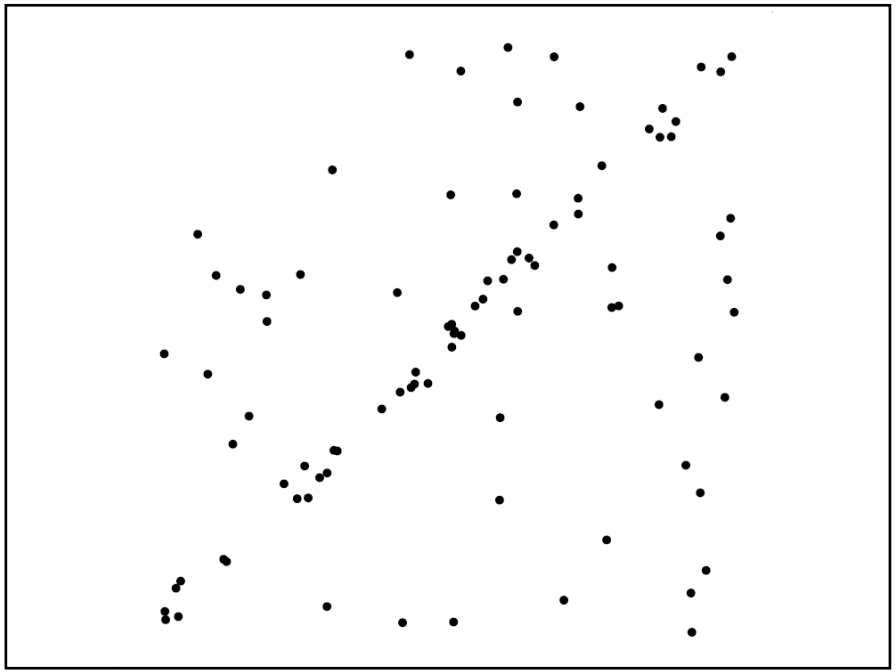
- Robust fitting can deal with a few outliers – what if we have very many?

- Random sample consensus (RANSAC):
  very general framework for model fitting in the presence of outliers

- Outline
  - Choose a small subset of points uniformly at random
  - Fit a model to that subset
  - Find all remaining points that are “close” to the model and reject the rest as outliers
  - Do this many times and choose the best model

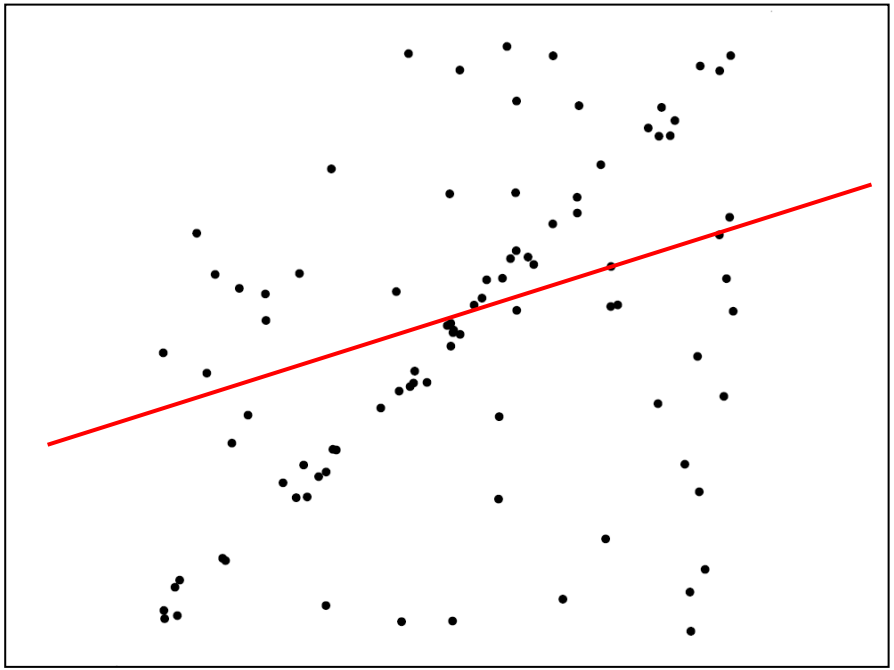
M. A. Fischler, R. C. Bolles. Random Sample Consensus: A Paradigm for Model Fitting with Applications to Image Analysis and Automated Cartography. Comm. of the ACM, Vol 24, pp 381-395, 1981.

RANSAC for Line Fitting Example



Source: R. Raguram

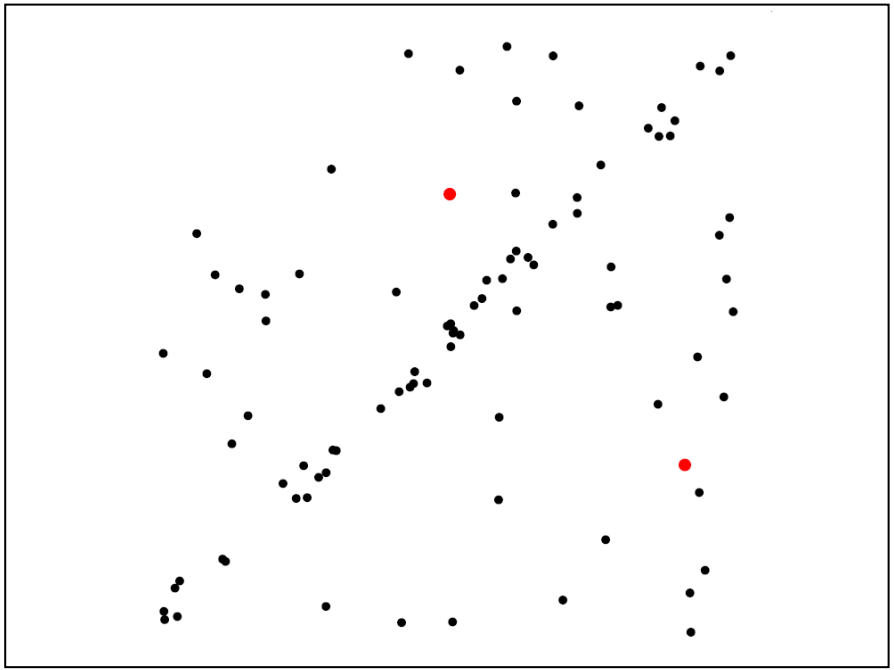
RANSAC for Line Fitting Example



Least-squares fit

Source: R. Raguram

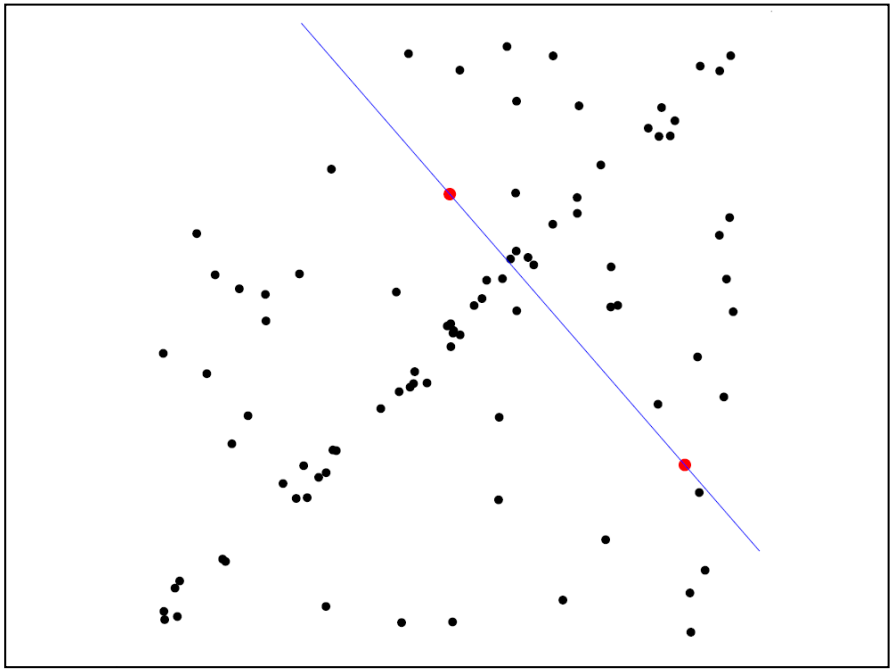
RANSAC for Line Fitting Example



1. Randomly select minimal subset of points

Source: R. Raguram

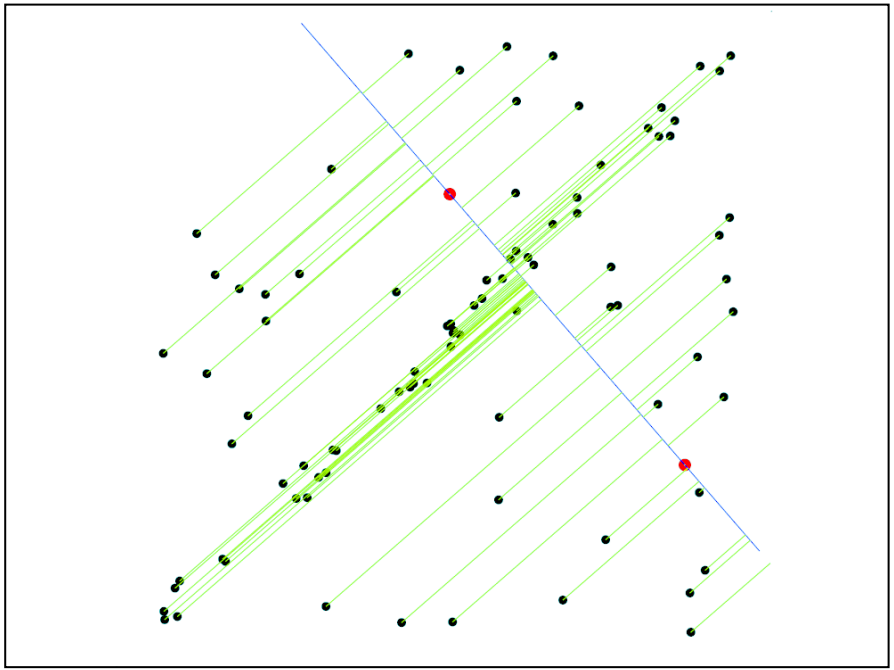
RANSAC for Line Fitting Example



1. Randomly select minimal subset of points
2. Hypothesize a model

Source: R. Raguram

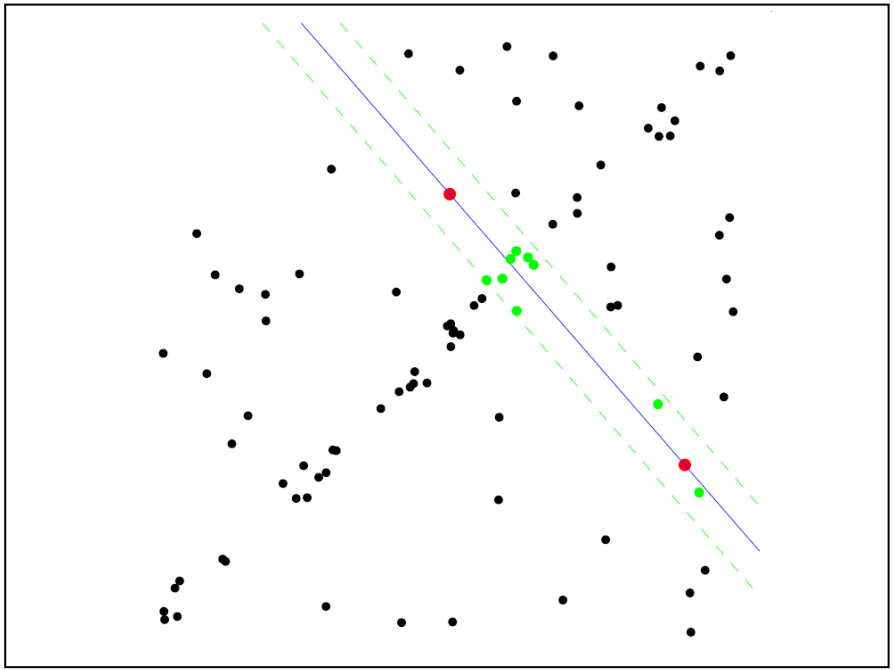
RANSAC for Line Fitting Example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function

Source: R. Raguram

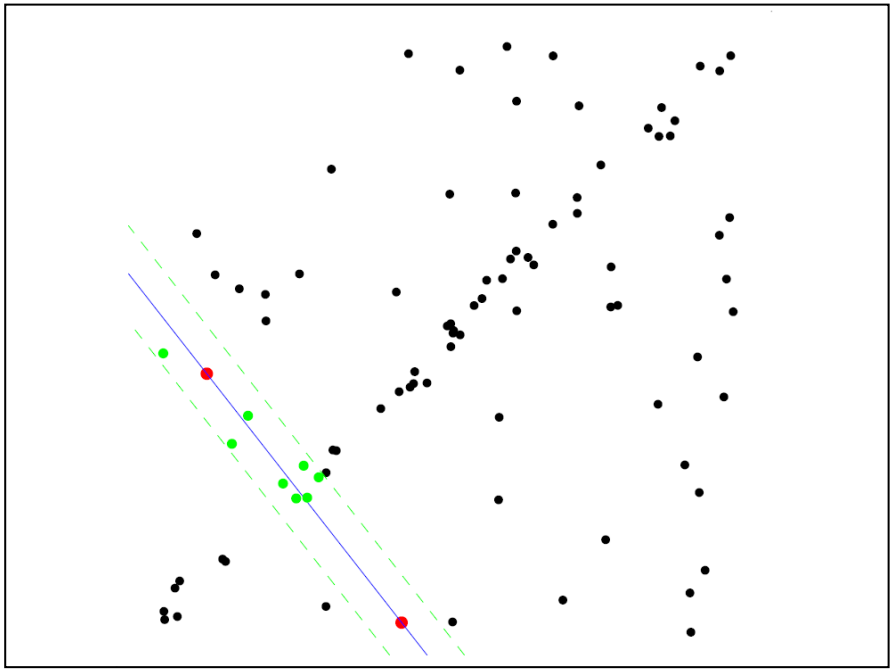
RANSAC for Line Fitting Example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model

Source: R. Raguram

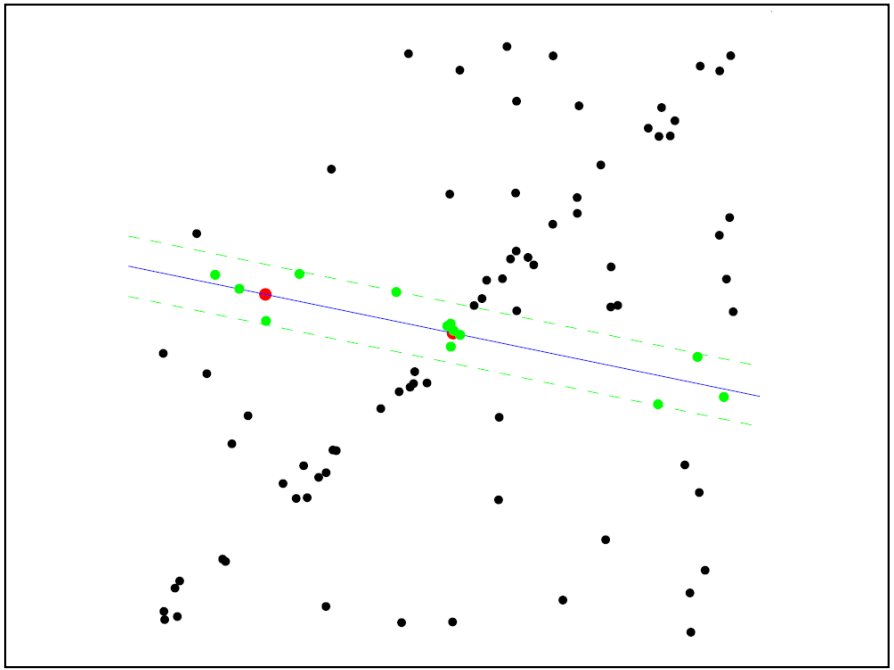
RANSAC for Line Fitting Example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize-and-verify loop

Source: R. Raguram

RANSAC for Line Fitting Example

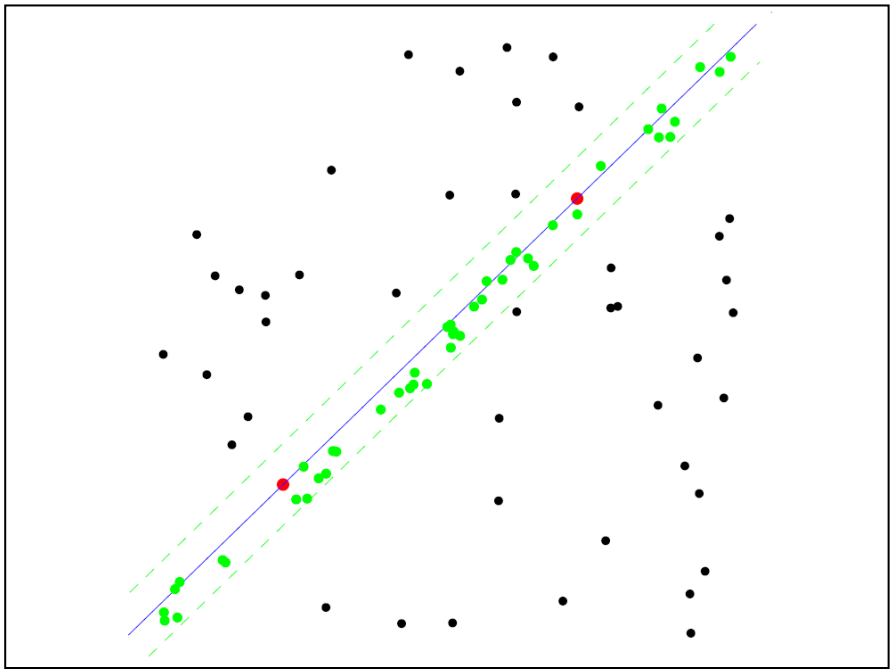


1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize-and-verify loop

Source: R. Raguram

RANSAC for Line Fitting Example

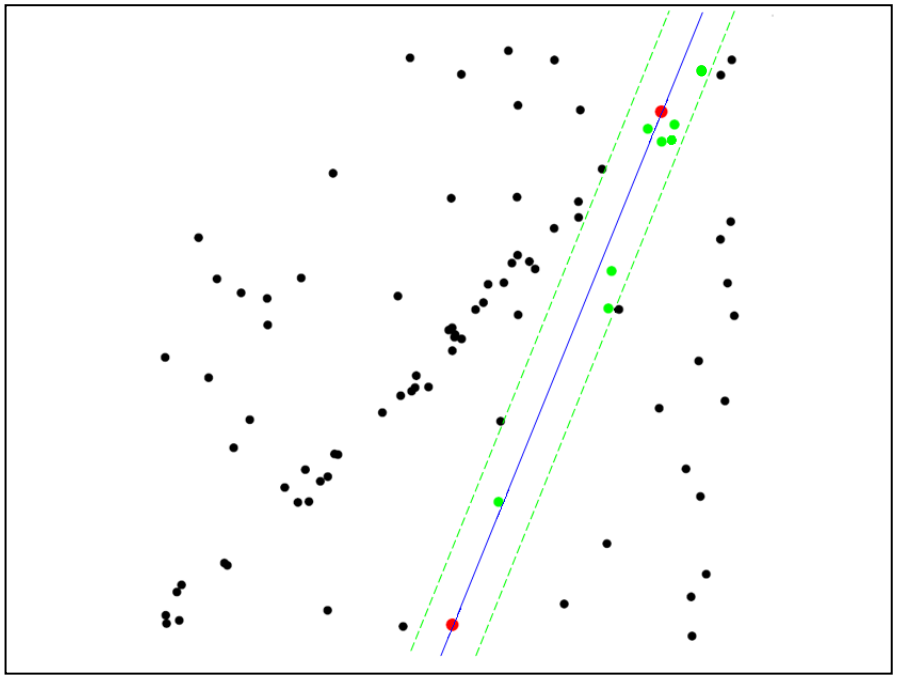
Uncontaminated sample



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize-and-verify loop

Source: R. Raguram

RANSAC for Line Fitting Example



1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize-and-verify loop

Source: R. Raguram

RANSAC for Line Fitting

- Repeat ***N*** times:
- Draw ***s*** points uniformly at random
- Fit line to these ***s*** points
- Find inliers to this line among the remaining points (i.e., points whose distance from the line is less than ***t***)
- If there are ***d*** or more inliers, accept the line and refit using all inliers

Choosing the Parameters

- Initial number of points *s*
  - Typically minimum number needed to fit the model
- Distance threshold *t*
  - Choose *t* so probability for inlier is *p* (e.g. 0.95)
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$
- Number of samples *N*
  - Choose *N* so that, with probability *p*, at least one random sample is free from outliers (e.g. *p*=0.99) (outlier ratio: *e*)

$$\left(1-\left(1-e\right)^s\right)^N = 1-p$$

$$N = \log\left(1-p\right) / \log\left(1-\left(1-e\right)^s\right)$$

	proportion of outliers *e*						
s	5%	10%	20%	25%	30%	40%	50%
2	2	3	5	6	7	11	17
3	3	4	7	9	11	19	35
4	3	5	9	13	17	34	72
5	4	6	12	17	26	57	146
6	4	7	16	24	37	97	293
7	4	8	20	33	54	163	588
8	5	9	26	44	78	272	1177

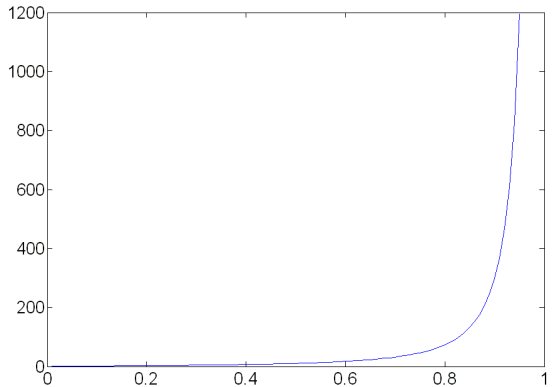
Source: M. Pollefeys

Choosing the Parameters

- Initial number of points *s*
  - Typically minimum number needed to fit the model
- Distance threshold *t*
  - Choose *t* so probability for inlier is *p* (e.g. 0.95)
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$
- Number of samples *N*
  - Choose *N* so that, with probability *p*, at least one random sample is free from outliers (e.g. *p*=0.99) (outlier ratio: *e*)

$$\left(1-\left(1-e\right)^s\right)^N = 1-p$$

$$N = \log\left(1-p\right) / \log\left(1-\left(1-e\right)^s\right)$$



Source: M. Pollefeys

Choosing the Parameters

- Initial number of points *s*
  - Typically minimum number needed to fit the model
- Distance threshold *t*
  - Choose *t* so probability for inlier is *p* (e.g. 0.95)
  - Zero-mean Gaussian noise with std. dev. σ: $t^2=3.84\sigma^2$
- Number of samples *N*
  - Choose *N* so that, with probability *p*, at least one random sample is free from outliers (e.g. *p*=0.99) (outlier ratio: *e*)
- Consensus set size *d*
  - Should match expected inlier ratio

Source: M. Pollefeys

Adaptively Determining the Number of Samples

- Inlier ratio *e* is often unknown a priori, so pick worst case, e.g. 50%, and adapt if more inliers are found, e.g. 80% would yield *e*=0.2

- Adaptive procedure:
  - *N*=∞, *sample_count* =0
  - While *N* >*sample_count*
    - Choose a sample and count the number of inliers
    - Set e = 1 – (number of inliers)/(total number of points)
    - Recompute *N* from *e:*

$$N = \log(1-p)/\log\left(1-(1-e)^s\right)$$

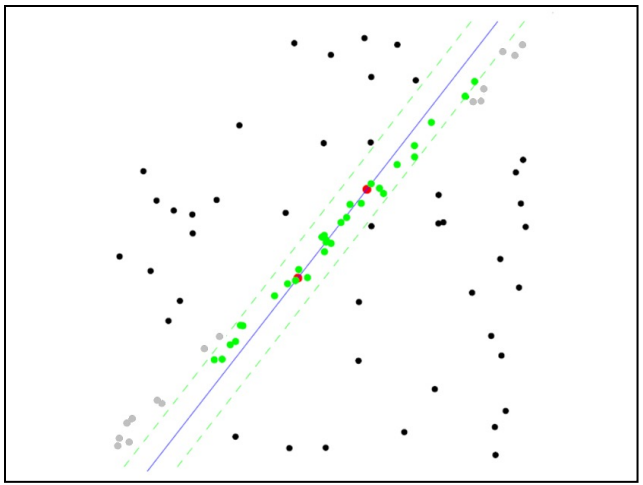
    - Increment the *sample_count* by 1

Source: M. Pollefeys

RANSAC Pros and Cons

- Pros
  - Simple and general
  - Applicable to many different problems
  - Often works well in practice

- Cons
  - Lots of parameters to tune
  - Doesn't work well for low inlier ratios (too many iterations, or can fail completely)
  - Can't always get a good initialization of the model based on the minimum number of samples



The RANSAC Song

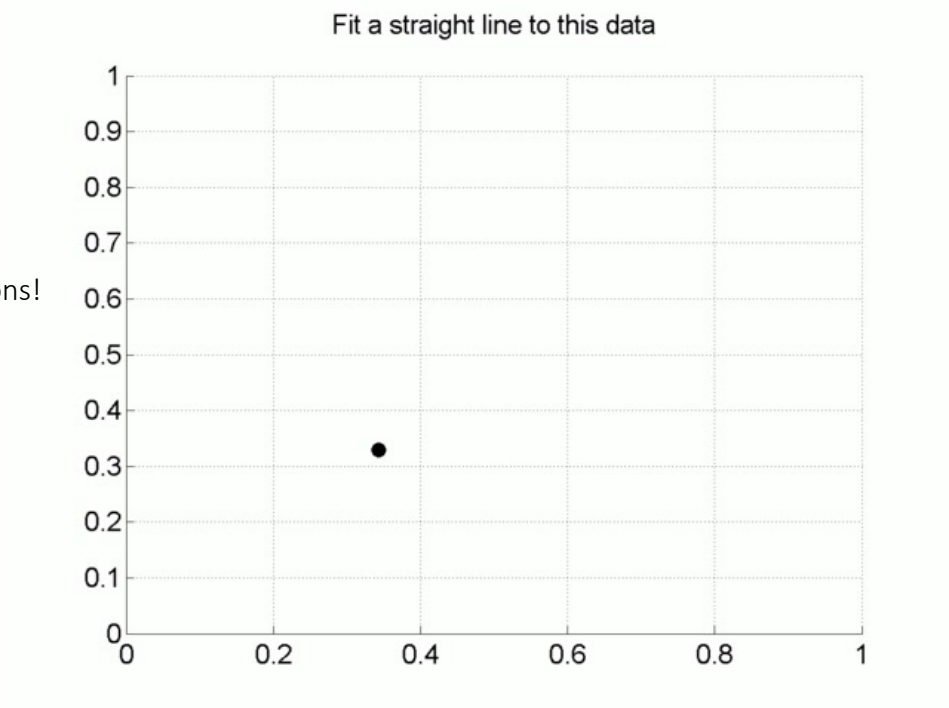
When you have outliers you may face much frustration
if you include them in a model fitting operation.
But if your model's fit to a sample set of minimal size,
the probability of the set being outlier-free will rise.
Brute force tests of all sets will cause computational constipation.

N random samples
will provide an example
of a fitted model uninfluenced by outliers. No need to test all combinations!

Each random trial should have its own unique sample set
and make sure that the sets you choose are not degenerate.
N, the number of sets, to choose is based on the probability
of a point being an outlier, and of finding a set that's outlier free.
Updating *N* as you go will minimise the time spent.

So if you gamble
that *N* samples are ample
to fit a model to your set of points, it's likely that you will win the bet.

Select the set that boasts
that its number of inliers is the most (you're almost there).
Fit a new model just to those inliers and discard the rest,
an estimated model for your data is now possessed!
This marks the end point of your model fitting quest.



Projective Reconstruction from 2 views

Two(or Multi)-view Geometry

- Cameras P and P' such that

$$\mathbf{x} = \mathrm{P}\mathbf{X} \qquad \mathbf{x}' = \mathrm{P}'\mathbf{X}$$

- Baseline between the cameras is non-zero.
  - Scene geometry (structure)
    - Given projections of the same 3D point in two or more images, how do we compute the 3D coordinates of that point?
  - Correspondence (stereo matching)
    - Given a point in just one image, how does it constrain the position of the corresponding point in the second image?
  - Camera geometry (motion)
    - Given a set of corresponding points in two images, what are the cameras for the two views?

Statement of the problem

Given

Corresponding points $\mathbf{x}_i \leftrightarrow \mathbf{x}'_i$ in two images.

Find

Cameras $\mathtt{P}$ and $\mathtt{P}'$ and 3D points x_i such that

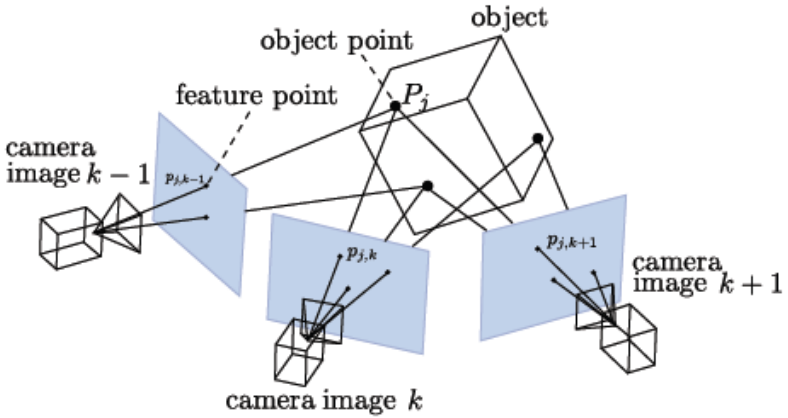
$$\mathbf{x}_i = \mathrm{P}\mathrm{x}_i \quad ; \quad \mathbf{x}'_i = \mathrm{P}\mathrm{x}_i$$

From Epipolar Geometry to Camera Calibration

- Estimating the fundamental matrix is known as “weak calibration”.
- If we know the calibration matrices of the two cameras, we can estimate the essential matrix: $E = K^T F K'$
- The essential matrix gives us the relative rotation and translation between the cameras, or their extrinsic parameters.

Two-frame Structure-from-Motion

- Structure from motion
  - Simultaneously recovering 3D structure and pose from correspondences



- Two-frame case
  - Compute fundamental matrix $\mathbf{F}$ between the two views
  - First camera matrix: **[I|0]**
  - Second camera matrix: **[R|t]**

$$z\mathbf{x} = [\mathbf{I}\,|\,\mathbf{0}]\mathbf{X}, \quad z'\mathbf{x}' = [\mathbf{R}\,|\,\mathbf{t}]\mathbf{X} \quad \Rightarrow \quad z'\mathbf{x}' = \mathbf{R}[\mathbf{I}\,|\,\mathbf{0}]\mathbf{X} + \mathbf{t} = z\mathbf{R}\mathbf{x} + \mathbf{t}$$

$$z'\mathbf{t}\times\mathbf{x}' = z\mathbf{t}\times\mathbf{R}\mathbf{x} \quad \mathbf{x}'\cdot(z'\mathbf{t}\times\mathbf{x}') = \mathbf{x}'\cdot(z\mathbf{t}\times\mathbf{R}\mathbf{x}) = 0$$

$$\mathbf{x}'^{\mathrm{T}}[\mathbf{t}_\times]\mathbf{R}\mathbf{x} = 0$$

$$\mathbf{F} = [\mathbf{t}_\times]\mathbf{R} \qquad \mathbf{t}\text{: epipole } (\mathbf{F}^{\mathrm{T}}\mathbf{t} = 0), \qquad \mathbf{R} = -[\mathbf{t}_\times]\mathbf{F}$$

Steps of projective reconstruction

Reconstruction takes place in the following steps :

- Compute the fundamental matrix $\mathtt{F}$ from point correspondences
- Factor the fundamental matrix as

$$\mathtt{F} = [\mathbf{t}]_{\times}\mathtt{M}$$

- The two camera matrices are

$$\mathtt{P} = [\mathtt{I} \mid 0] \text{ and } \mathtt{P}' = [\mathtt{M} \mid \mathbf{t}] \ .$$

- Compute the points x_i by triangulation

Non-uniqueness of factorization

- Factorization of the fundamental matrix is not unique.
- General formula : for varying $\mathbf{v}$ and λ

$$\mathtt{P} = [\mathtt{I} \mid 0] \;\; ; \;\; \mathtt{P}' = [\mathtt{M} + \mathbf{e}'\mathbf{v}^\top \mid \lambda \mathbf{e}']$$

- Difference factorizations give configurations varying by a projective transformation.
- 4-parameter family of solutions with $\mathtt{P} = [\mathtt{I} \mid 0]$.

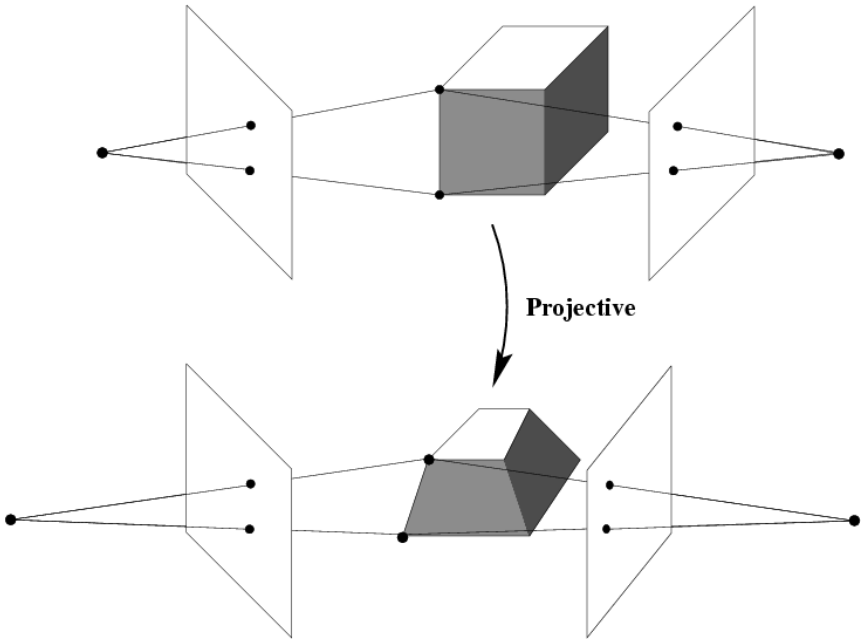
Ambiguity of Structure from Motion

$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\frac{1}{k}\mathbf{P}\right)(k\mathbf{X})$$

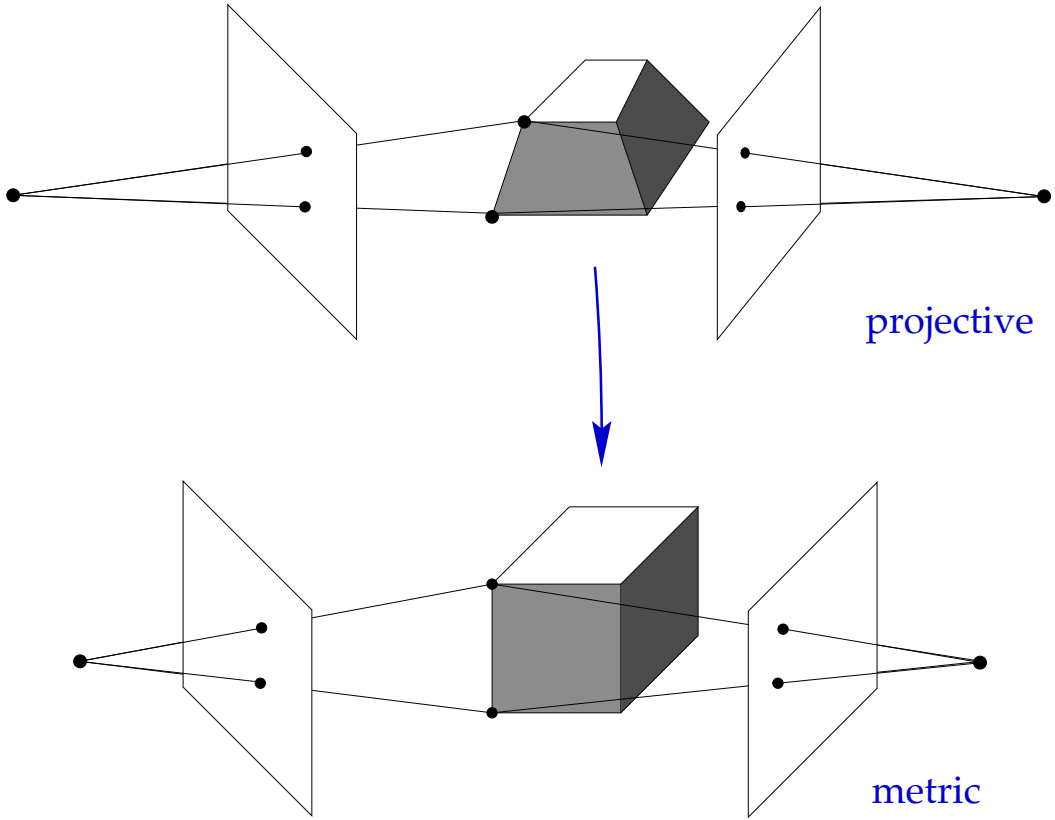
It is impossible to recover the absolute scale of the scene!

- More generally, for any transformation Q

$$\mathbf{x} = \mathbf{P}\mathbf{X} = \left(\mathbf{P}\mathbf{Q}^{-1}\right)(\mathbf{Q}\mathbf{X})$$

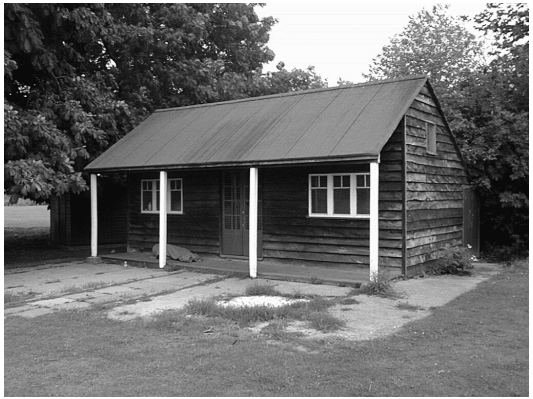


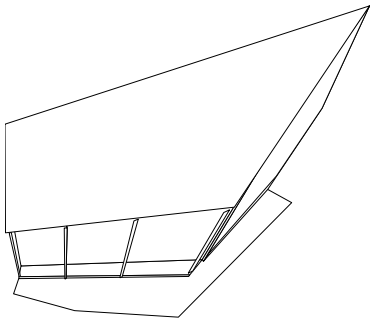
Metric Reconstruction

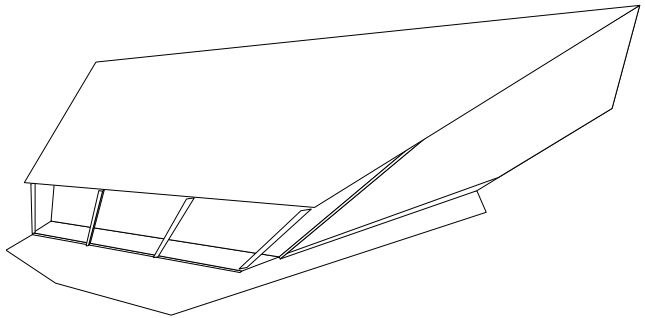


Correct: angles, length ratios.

Projective Reconstruction







Direct Metric Reconstruction

Use 5 or more 3D points with known Euclidean coordinates to determine H