Camera Calibration

2017 Fall

National Cheng Kung University

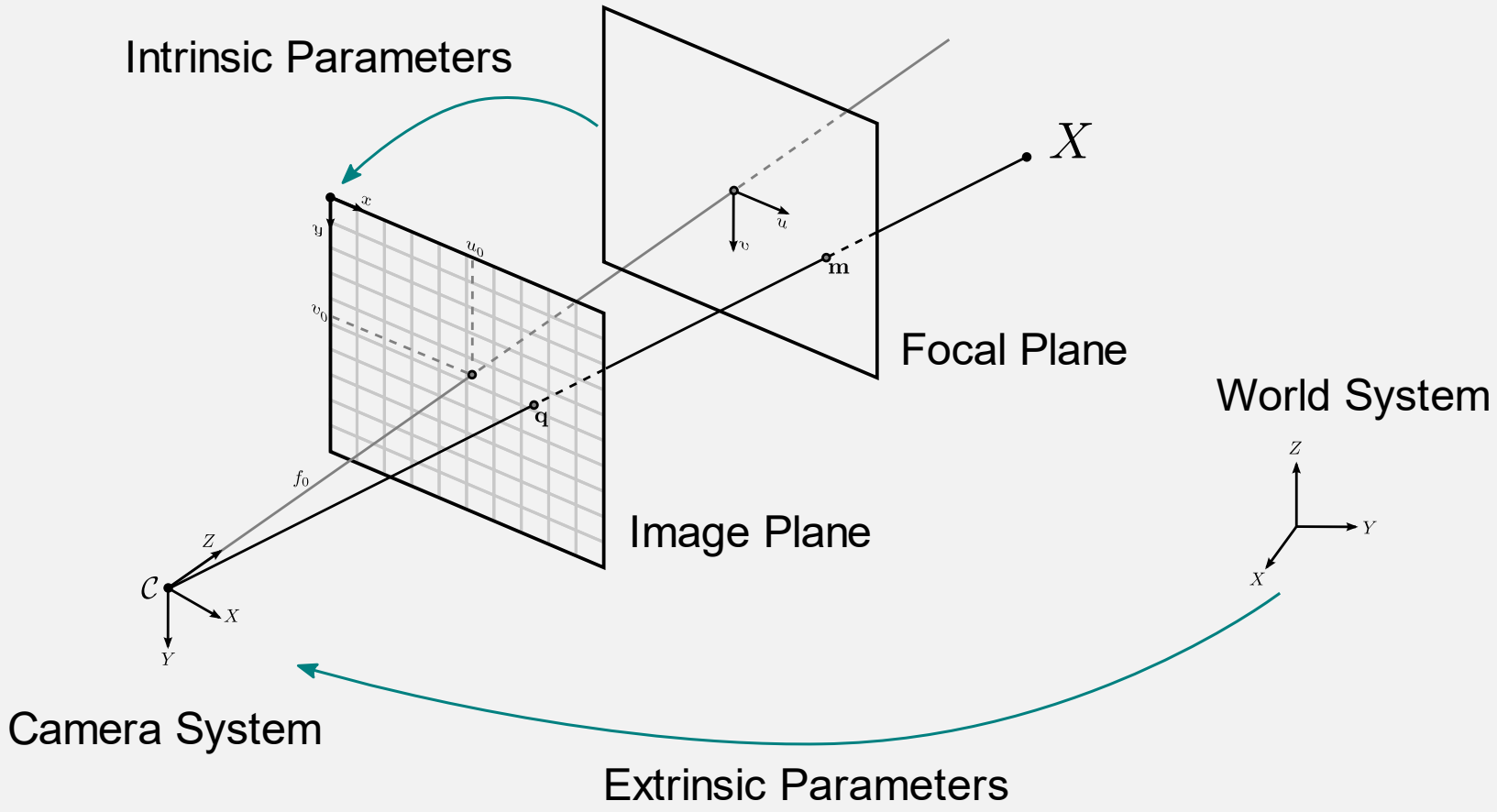
Instructor: Min-Chun Hu 胡敏君

Parts of the slides are from

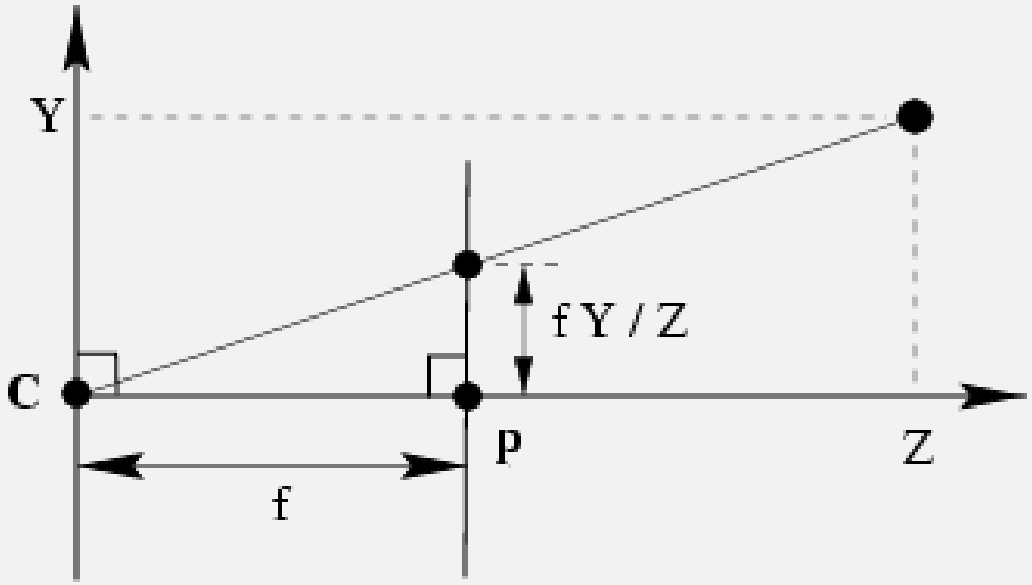
“Digital Visual Effects”

by Prof. Yung-Yu Chuang, NTU

Camera Calibration



Pinhole Camera Projection Model



$$x = \frac{fX}{Z} \qquad y = \frac{fY}{Z}$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

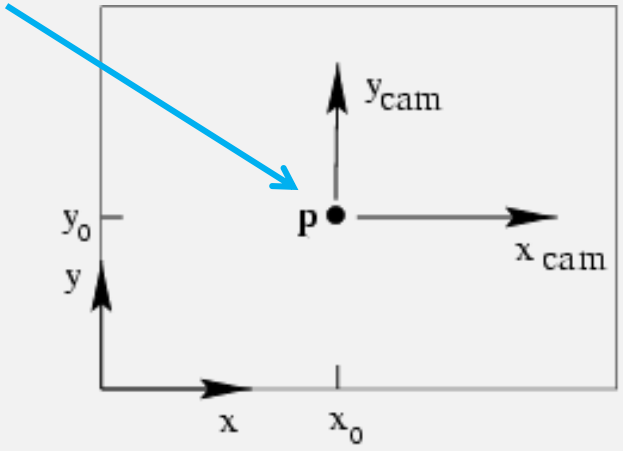
intrinsic matrix

extrinsic matrix (Camera Coordinate =World Coordinate)

$$\mathbf{x} \sim \mathbf{K}[\mathbf{I}|0]\mathbf{X}$$

Principal Point Offset

principal point



$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$\mathbf{x} \sim \mathbf{K}\left[\mathbf{I}|0\right]\mathbf{X}$$

Intrinsic Matrix

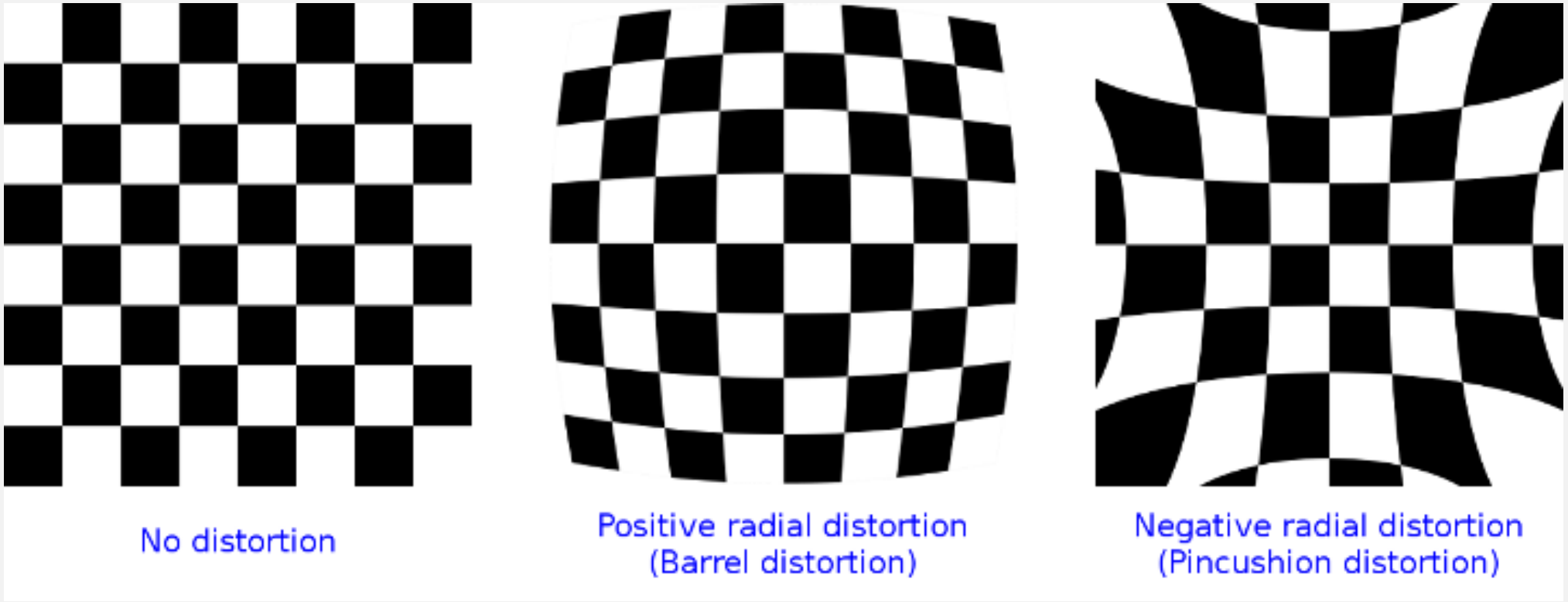
$$\mathbf{K} = \begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$\mathbf{K} = \begin{bmatrix} fa & s & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$

Good enough for modeling the camera projection?

a : aspect ratio (for non-square pixels)
s : skew (for non-rectangular pixels)

Radial Distortion Caused by Imperfect Lens



No distortion

Positive radial distortion
(Barrel distortion)

Negative radial distortion
(Pincushion distortion)

Camera Pose (Camera Rotation and Translation)

$$\begin{pmatrix} X' \\ Y' \\ Z' \end{pmatrix} = \mathbf{R}_{3\times3} \begin{pmatrix} X \\ Y \\ Z \end{pmatrix} + \mathbf{t}$$

$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \mathbf{R} | \mathbf{t} \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$\mathbf{x} \sim \mathbf{K} \begin{bmatrix} \mathbf{R} | \mathbf{t} \end{bmatrix} \mathbf{X}$$

Camera Calibration Methods

- Two main categories for estimating both intrinsic and extrinsic parameters
  - Photometric calibration
    - Use reference objects with known geometry (Chessboard)
  - Self Calibration
    - Assume static scene and use structure from motion

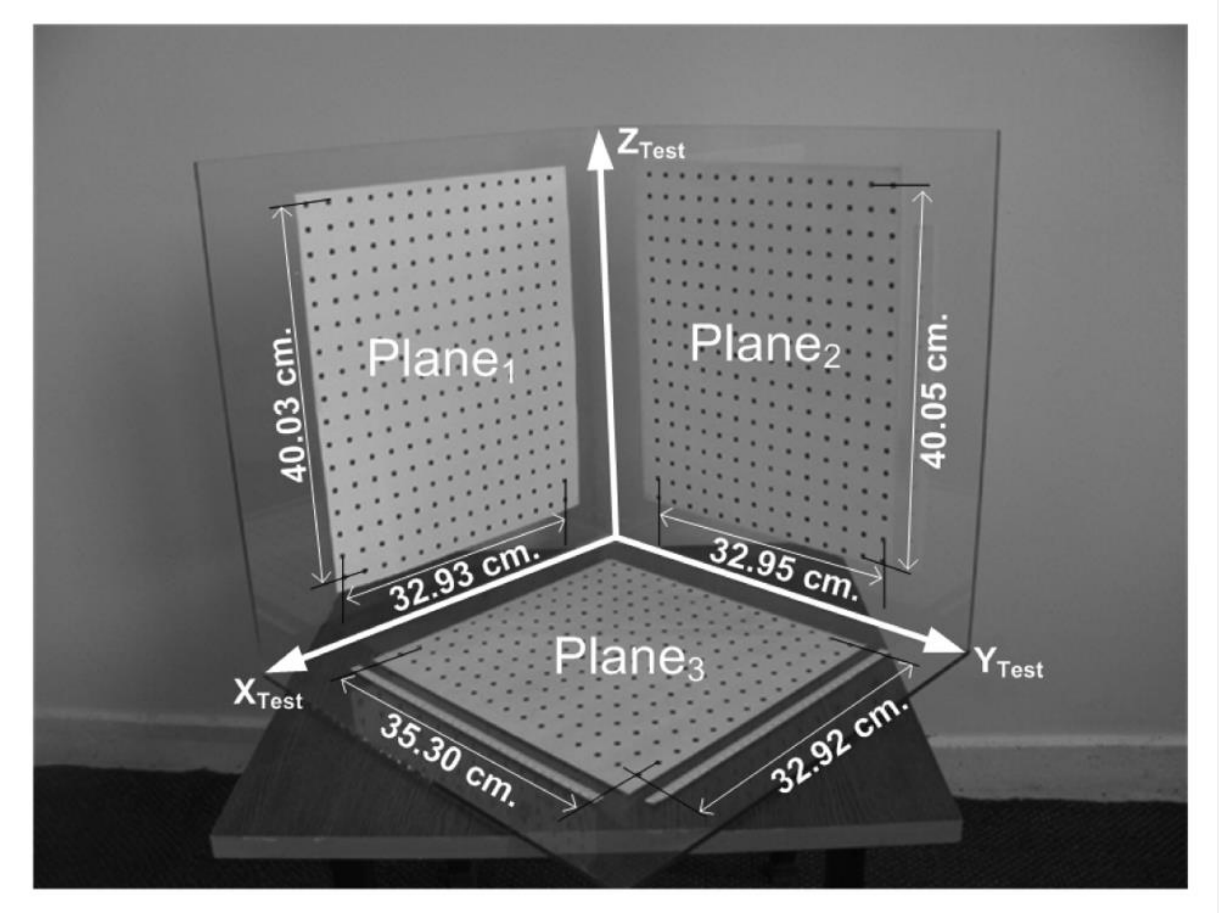
Linear Regression

$$\mathbf{x} \sim \mathbf{K}\left[\mathbf{R}|\mathbf{t}\right]\mathbf{X} = \mathbf{M}\mathbf{X}$$

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \sim \begin{bmatrix} m_{00} & m_{01} & m_{02} & m_{03} \\ m_{10} & m_{11} & m_{12} & m_{13} \\ m_{20} & m_{21} & m_{22} & 1 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$

- Directly estimate 11 unknowns in the **M** matrix using known 3D points (X_i, Y_i, Z_i) and measured feature positions (u_i, v_i)

Linear Regression (Cont.)



Linear Regression (Cont.)

$$u_i = \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1}$$

$$v_i = \frac{m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1}$$

$$u_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1) = m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}$$

$$v_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1) = m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}$$

Linear Regression (Cont.)

$$u_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1) = m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}$$

$$v_i(m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1) = m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}$$

$$\begin{bmatrix} X_i & Y_i & Z_i & 1 & 0 & 0 & 0 & 0 & -u_iX_i & -u_iY_i & -u_iZ_i & -u_i \\ 0 & 0 & 0 & 0 & X_i & Y_i & Z_i & 1 & -v_iX_i & -v_iY_i & -v_iZ_i & -v_i \end{bmatrix} \begin{bmatrix} m_{00} \\ m_{01} \\ m_{02} \\ m_{03} \\ m_{10} \\ m_{11} \\ m_{12} \\ m_{13} \\ m_{20} \\ m_{21} \\ m_{22} \\ m_{23} \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \end{bmatrix}$$

Solve for Projection Matrix M
using least-square techniques

Normal Equation

- Given an overdetermined linear system

$$\mathbf{Ax} = \mathbf{b}$$

- The normal equation is to minimize the sum of square differences between left and right sides

$$\mathbf{A^T Ax} = \mathbf{A^T b}$$

Normal Equation

$$\begin{bmatrix} a_{11} & \dots & a_{1m} \\ \vdots & & \vdots \\ \vdots & & \vdots \\ \vdots & & \vdots \\ a_{n1} & \dots & a_{nm} \end{bmatrix} \begin{bmatrix} x_1 \\ \vdots \\ x_m \end{bmatrix} = \begin{bmatrix} b_1 \\ \vdots \\ \vdots \\ \vdots \\ b_n \end{bmatrix}$$

$$\mathbf{Ax}-\mathbf{b} = \begin{bmatrix} \sum_{j=1}^{m} a_{1j}x_j \\ \vdots \\ \sum_{j=1}^{m} a_{ij}x_j \\ \vdots \\ \sum_{j=1}^{m} a_{nj}x_j \end{bmatrix} - \begin{bmatrix} b_1 \\ \vdots \\ b_i \\ \vdots \\ b_n \end{bmatrix} = \begin{bmatrix} \left(\sum_{j=1}^{m} a_{1j}x_j\right) - b_1 \\ \vdots \\ \left(\sum_{j=1}^{m} a_{ij}x_j\right) - b_i \\ \vdots \\ \left(\sum_{j=1}^{m} a_{nj}x_j\right) - b_n \end{bmatrix}$$

n×*m*, *n* equations, *m* variables

$$E(\mathbf{x}) = \left(\mathbf{Ax}-\mathbf{b}\right)^2 = \sum_{i=1}^{n}\left[\left(\sum_{j=1}^{m} a_{ij}x_j\right) - b_i\right]^2$$

Normal Equation

$$E(\mathbf{x}) = (\mathbf{Ax} - \mathbf{b})^2 = \sum_{i=1}^{n} \left[\left(\sum_{j=1}^{m} a_{ij} x_j \right) - b_i \right]^2$$

$$0 = \frac{\partial E}{\partial x_1} = \sum_{i=1}^{n} 2 \left[\left(\sum_{j=1}^{m} a_{ij} x_j \right) - b_i \right] a_{i1}$$

$$= 2 \sum_{i=1}^{n} a_{i1} \sum_{j=1}^{m} a_{ij} x_j - 2 \sum_{i=1}^{n} a_{i1} b_i$$

$$0 = \frac{\partial E}{\partial \mathbf{x}} = 2(\mathbf{A}^{\mathbf{T}}\mathbf{Ax} - \mathbf{A}^{\mathbf{T}}\mathbf{b}) \rightarrow \mathbf{A}^{\mathbf{T}}\mathbf{Ax} = \mathbf{A}^{\mathbf{T}}\mathbf{b}$$

Normal Equation

$$
\begin{aligned}
&\left(\mathbf{Ax}-\mathbf{b}\right)^2 \\
&=\left(\mathbf{Ax}-\mathbf{b}\right)^T\left(\mathbf{Ax}-\mathbf{b}\right) \\
&=\left(\left(\mathbf{Ax}\right)^T-\mathbf{b}^T\right)\left(\mathbf{Ax}-\mathbf{b}\right) \\
&=\left(\mathbf{x}^\mathbf{T}\mathbf{A}^\mathbf{T}-\mathbf{b}^\mathbf{T}\right)\left(\mathbf{Ax}-\mathbf{b}\right) \\
&=\mathbf{x}^\mathbf{T}\mathbf{A}^\mathbf{T}\mathbf{Ax}-\mathbf{b}^\mathbf{T}\mathbf{Ax}-\mathbf{x}^\mathbf{T}\mathbf{A}^\mathbf{T}\mathbf{b}+\mathbf{b}^\mathbf{T}\mathbf{b} \\
&=\mathbf{x}^\mathbf{T}\mathbf{A}^\mathbf{T}\mathbf{Ax}-\left(\mathbf{A}^\mathbf{T}\mathbf{b}\right)^\mathbf{T}\mathbf{x}-\left(\mathbf{A}^\mathbf{T}\mathbf{b}\right)^\mathbf{T}\mathbf{x}+\mathbf{b}^\mathbf{T}\mathbf{b}
\end{aligned}
$$

$$
\Rightarrow \frac{\partial E}{\partial \mathbf{x}}=2\mathbf{A}^\mathbf{T}\mathbf{Ax}-2\mathbf{A}^\mathbf{T}\mathbf{b}
$$

Linear Regression

- Advantages:
  - All parameters of the camera are summarized in one matrix
  - Given any world point, it is easy to predict the corresponding position in the image
- Disadvantages:
  - Intrinsic and extrinsic parameters can not be derived separately
  - More unknowns than true degrees of freedom

Non-Linear Optimization

- A probabilistic view of least square
- Feature measurement equations

$$u_i = f(\mathbf{M}, \mathbf{x}_i) + n_i = \hat{u}_i + n_i, \quad n_i \sim N(0, \sigma)$$
$$v_i = g(\mathbf{M}, \mathbf{x}_i) + m_i = \hat{v}_i + m_i, \quad m_i \sim N(0, \sigma)$$

- Probability of $\mathbf{M}$ given $\{(u_i, v_i)\}$:

$$P = \prod_i p(u_i|\hat{u}_i)p(v_i|\hat{v}_i)$$
$$= \prod_i e^{-(u_i-\hat{u}_i)^2/\sigma^2} e^{-(v_i-\hat{v}_i)^2/\sigma^2}$$

Optimal Estimation

- Likelihood of **M** given $\{(u_i, v_i)\}$:

$$L = -\log P = \sum_i (u_i - \hat{u}_i)^2/\sigma_i^2 + (v_i - \hat{v}_i)^2/\sigma_i^2$$

- Minimize L

How ?

Optimal Estimation

- Non-linear regression (least squares), because the relations between $\hat{u}_i$ and u_i are non-linear functions of $\mathbf{M}$

$$\mathbf{u} - \hat{\mathbf{u}} \sim \mathbf{u} - \mathbf{K}\left[\mathbf{R}|\mathbf{t}\right]\mathbf{X}$$

known constant

- Use Levenberg-Marquardt method to minimize the least square error

Least Square Fitting

Least Squares Problem

Find $\mathbf{x}^*$, a local minimizer for

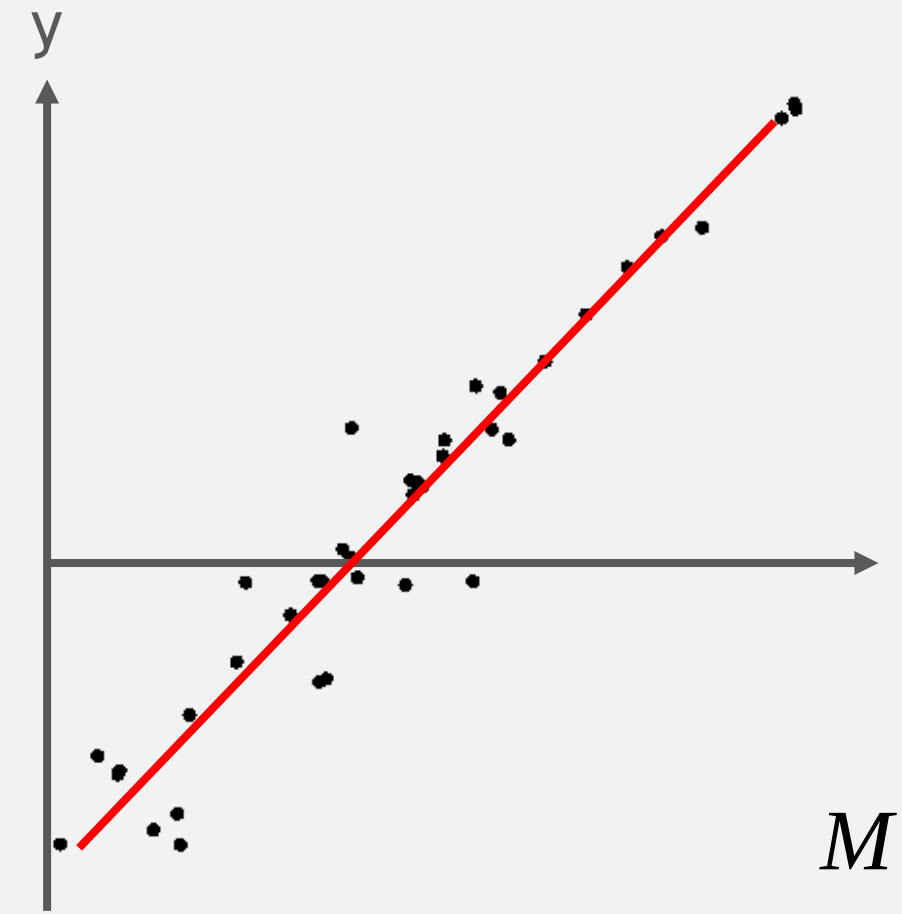
$$F(\mathbf{x}) = \frac{1}{2}\sum_{i=1}^{m}(f_i(\mathbf{x}))^2 \ ,$$

where $f_i : \mathbb{R}^n \mapsto \mathbb{R}$, $i = 1, \ldots, m$ are given functions, and $m \geq n$.

m: number of data points

n: number of parameters

Linear Least Square Fitting



model parameters

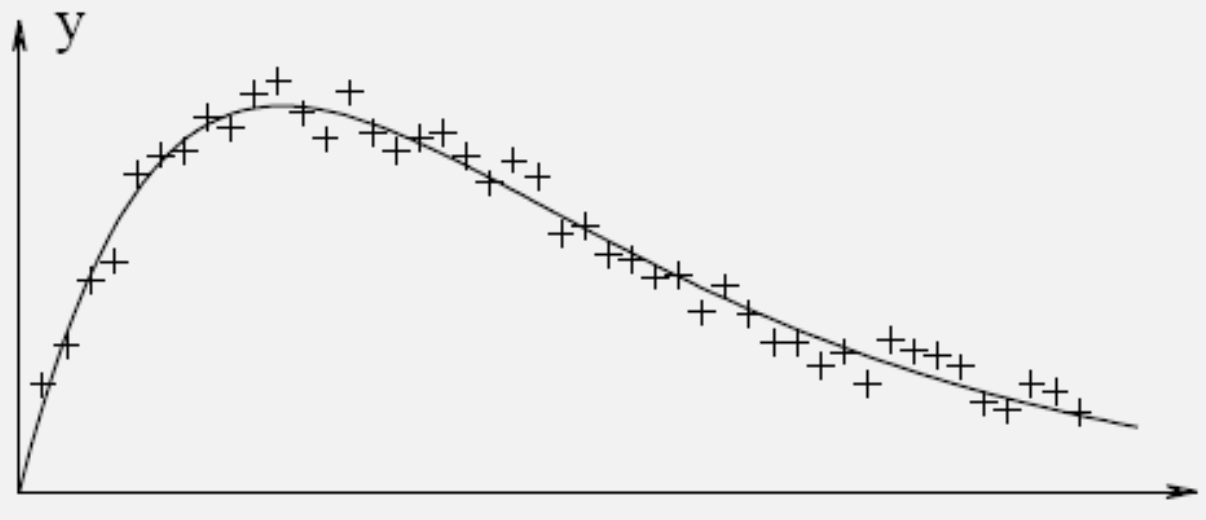
$$y(t) = M(t;\mathbf{x}) = x_0 + x_1 t$$

$$f_i(x) = y_i - M(t_i;\mathbf{x})$$

residual prediction

$M(t;\mathbf{x}) = x_0 + x_1 t + x_2 t^3$ is linear, too.

Nonlinear Least Square Fitting



model $M(t;\mathbf{x}) = x_3 e^{x_1 t} + x_4 e^{x_2 t}$

parameters $\mathbf{x} = [x_1, x_2, x_3, x_4]^T$

residuals $f_i(\mathbf{x}) = y_i - M(t_i;\mathbf{x})$

$= y_i - \left(x_3 e^{x_1 t} + x_4 e^{x_2 t}\right)$

Function Minimization

Least square is related to function minimization.

Global Minimizer

Given $F : \mathbb{R}^n \mapsto \mathbb{R}$. Find

$$\mathbf{x}^+ = \operatorname{argmin}_{\mathbf{x}} \{F(\mathbf{x})\} .$$

It is very hard to solve in general.
Here, we only consider a simpler problem of finding local minimum.

Local Minimizer

Given $F : \mathbb{R}^n \mapsto \mathbb{R}$. Find $\mathbf{x}^*$ so that

$$F(\mathbf{x}^*) \leq F(\mathbf{x}) \quad \text{for} \quad \|\mathbf{x} - \mathbf{x}^*\| < \delta .$$

Function Minimization

We assume that the cost function F is differentiable and so smooth that the following *Taylor expansion* is valid,[2)]

$$F(\mathbf{x}+\mathbf{h}) \;=\; F(\mathbf{x}) + \mathbf{h}^\top \mathbf{g} + \tfrac{1}{2}\mathbf{h}^\top \mathbf{H}\,\mathbf{h} + O(\|\mathbf{h}\|^3)\;,$$

where $\mathbf{g}$ is the *gradient*,

$$\mathbf{g} \;\equiv\; \mathbf{F}'(\mathbf{x}) \;=\; \begin{bmatrix} \dfrac{\partial F}{\partial x_1}(\mathbf{x}) \\ \vdots \\ \dfrac{\partial F}{\partial x_n}(\mathbf{x}) \end{bmatrix},$$

and $\mathbf{H}$ is the *Hessian*,

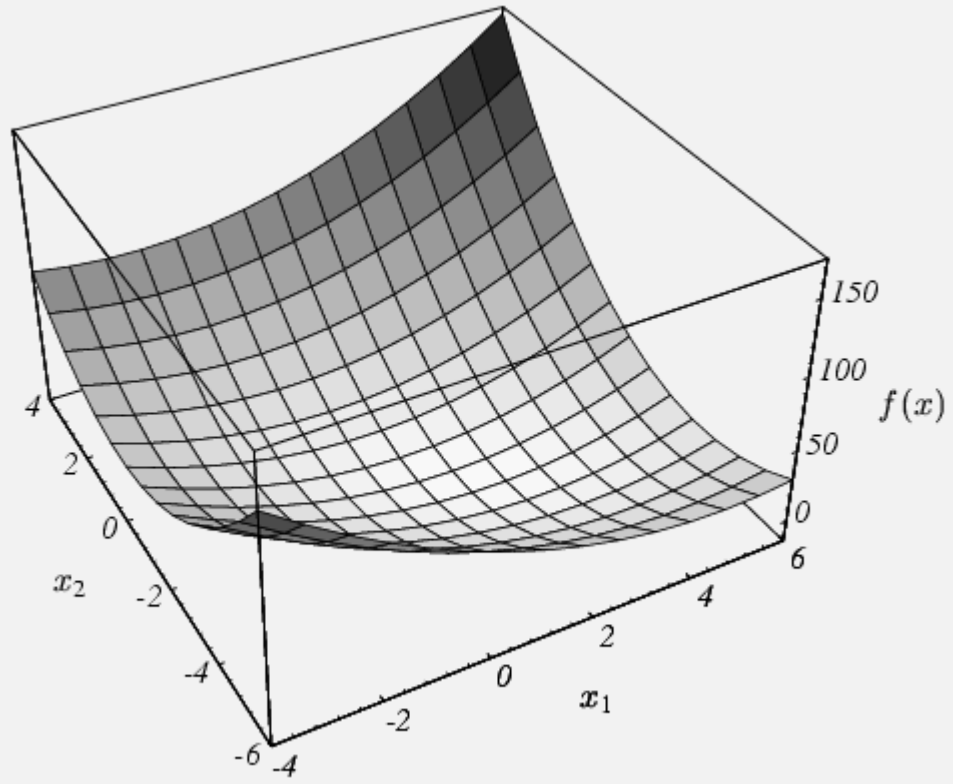
$$\mathbf{H} \;\equiv\; \mathbf{F}''(\mathbf{x}) \;=\; \left[\frac{\partial^2 F}{\partial x_i \partial x_j}(\mathbf{x})\right] .$$

Quadratic Functions

Approximate the function with
a quadratic function within
a small neighborhood

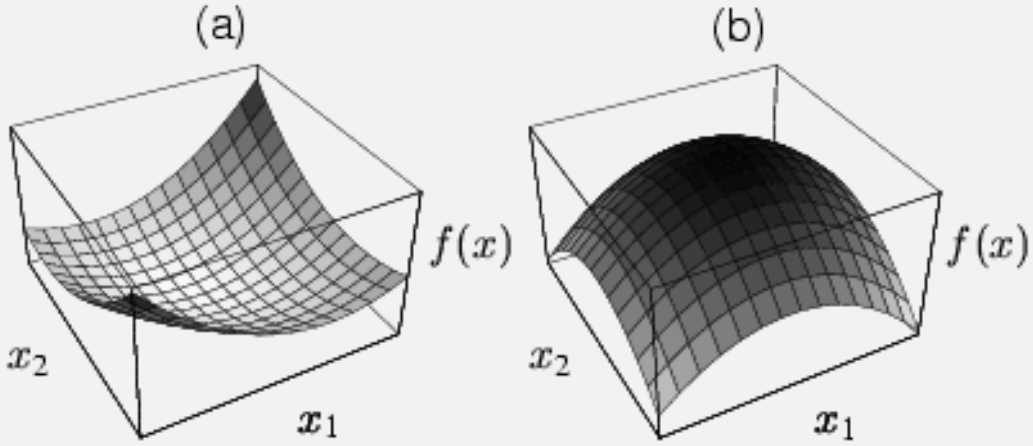
$$f(x) = \frac{1}{2}x^T Ax - b^T x + c$$

$$A = \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}, \qquad b = \begin{bmatrix} 2 \\ -8 \end{bmatrix}, \qquad c = 0.$$



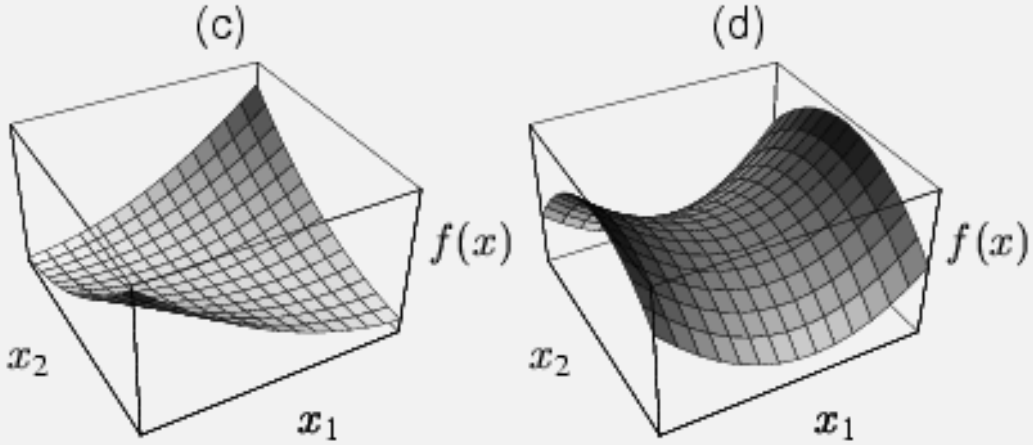
Quadratic Functions

A is positive definite.
All eigenvalues are positive.
For all x, $x^T Ax>0$.



A is negative definite.
All eigenvalues are negative.
For all x, $x^T Ax<0$.

A is singular



A is indefinite

Function Minimization

Theorem 1.5. Necessary condition for a local minimizer.
If x^* is a local minimizer, then

$$g^* \equiv F'(x^*) = 0 .$$

Why?

By definition, if $\mathbf{x}^*$ is a local minimizer,

$$\|\mathbf{h}\| \text{ is small enough} \longrightarrow \mathbf{F(x^* + h) > F(x^*)}$$

$$\mathbf{F(x^* + h) = F(x^*) + h^T F'(x^*) + O(\|h\|^2)}$$

Function Minimization

Theorem 1.5. Necessary condition for a local minimizer.
If $\mathbf{x}^*$ is a local minimizer, then

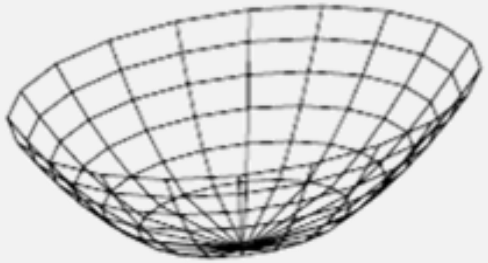
$$\mathbf{g}^* \equiv \mathbf{F}'(\mathbf{x}^*) = \mathbf{0}\,.$$

Definition 1.6. Stationary point. If

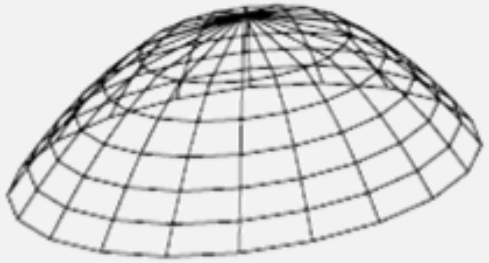
$$\mathbf{g}_s \equiv \mathbf{F}'(\mathbf{x}_s) = \mathbf{0}\,,$$

then $\mathbf{x}_s$ is said to be a *stationary point* for F.

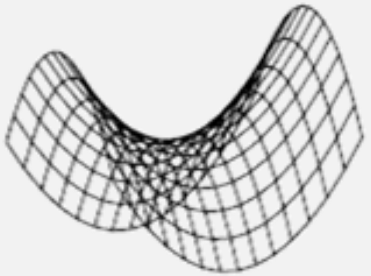
$$F(\mathbf{x}_s+\mathbf{h}) = F(\mathbf{x}_s) + \tfrac{1}{2}\mathbf{h}^\top \mathbf{H}_s\,\mathbf{h} + O(\|\mathbf{h}\|^3)$$



a) minimum



b) maximum



c) saddle point

Function Minimization

Theorem 1.8. Sufficient condition for a local minimizer.
Assume that $\mathbf{x}_s$ is a stationary point and that $\mathbf{F}''(\mathbf{x}_s)$ is positive definite.
Then $\mathbf{x}_s$ is a local minimizer.

$$F(\mathbf{x}_s+\mathbf{h}) = F(\mathbf{x}_s) + \tfrac{1}{2}\mathbf{h}^\top \mathbf{H}_s\, \mathbf{h} + O(\|\mathbf{h}\|^3)$$

$$\text{with } \mathbf{H}_s = \mathbf{F}''(\mathbf{x}_s)$$

If we request that $\mathbf{H}_s$ is *positive definite*, then its eigenvalues are greater than some number $\delta > 0$

$$\mathbf{h}^\top \mathbf{H}_s\, \mathbf{h} > \delta\, \|\mathbf{h}\|^2$$

Descent Methods

$\mathbf{x}_0, \mathbf{x}_1, \mathbf{x}_2, \ldots, \mathbf{x}_k \rightarrow \mathbf{x}^* \quad \text{for} \quad k \rightarrow \infty$

1. Find a descent direction $\mathbf{h}_d$
2. find a step length giving a good decrease in the F-value.

Algorithm Descent method

begin

$k := 0$; $\mathbf{x} := \mathbf{x}_0$; *found* := **false** {Starting point}

while (**not** *found*) **and** ($k < k_{max}$)

$\mathbf{h}_d$:= search_direction($\mathbf{x}$) {From $\mathbf{x}$ and downhill}

if (no such $\mathbf{h}$ exists)

found := **true** {$\mathbf{x}$ is stationary}

else

α := step_length($\mathbf{x}, \mathbf{h}_d$) {from $\mathbf{x}$ in direction $\mathbf{h}_d$}

$\mathbf{x} := \mathbf{x} + \alpha\mathbf{h}_d$; $k := k+1$ {next iterate}

end

Descent Direction

$$F(\mathbf{x}+\alpha\mathbf{h}) = F(\mathbf{x}) + \alpha\mathbf{h}^\top\mathbf{F}'(\mathbf{x}) + O(\alpha^2)$$
$$\simeq F(\mathbf{x}) + \alpha\mathbf{h}^\top\mathbf{F}'(\mathbf{x}) \quad \text{for } \alpha \text{ sufficiently small.}$$

Definition: Descent direction.

$\mathbf{h}$ is a descent direction for F at $\mathbf{x}$ if $\mathbf{h}^\top\mathbf{F}'(\mathbf{x}) < 0$.

Steepest Descent Method

$$F(\mathbf{x}+\alpha\mathbf{h}) = F(\mathbf{x}) + \alpha\mathbf{h}^\top\mathbf{F}'(\mathbf{x}) + O(\alpha^2)$$
$$\simeq F(\mathbf{x}) + \alpha\mathbf{h}^\top\mathbf{F}'(\mathbf{x}) \quad \text{for } \alpha \text{ sufficiently small.}$$

$$\boxed{\frac{F(\mathbf{x}) - F(\mathbf{x}+\alpha\mathbf{h})}{\alpha\|\mathbf{h}\|}} = -\frac{1}{\|\mathbf{h}\|}\mathbf{h}^\top\mathbf{F}'(\mathbf{x}) = -\|\mathbf{F}'(\mathbf{x})\|\cos\theta$$

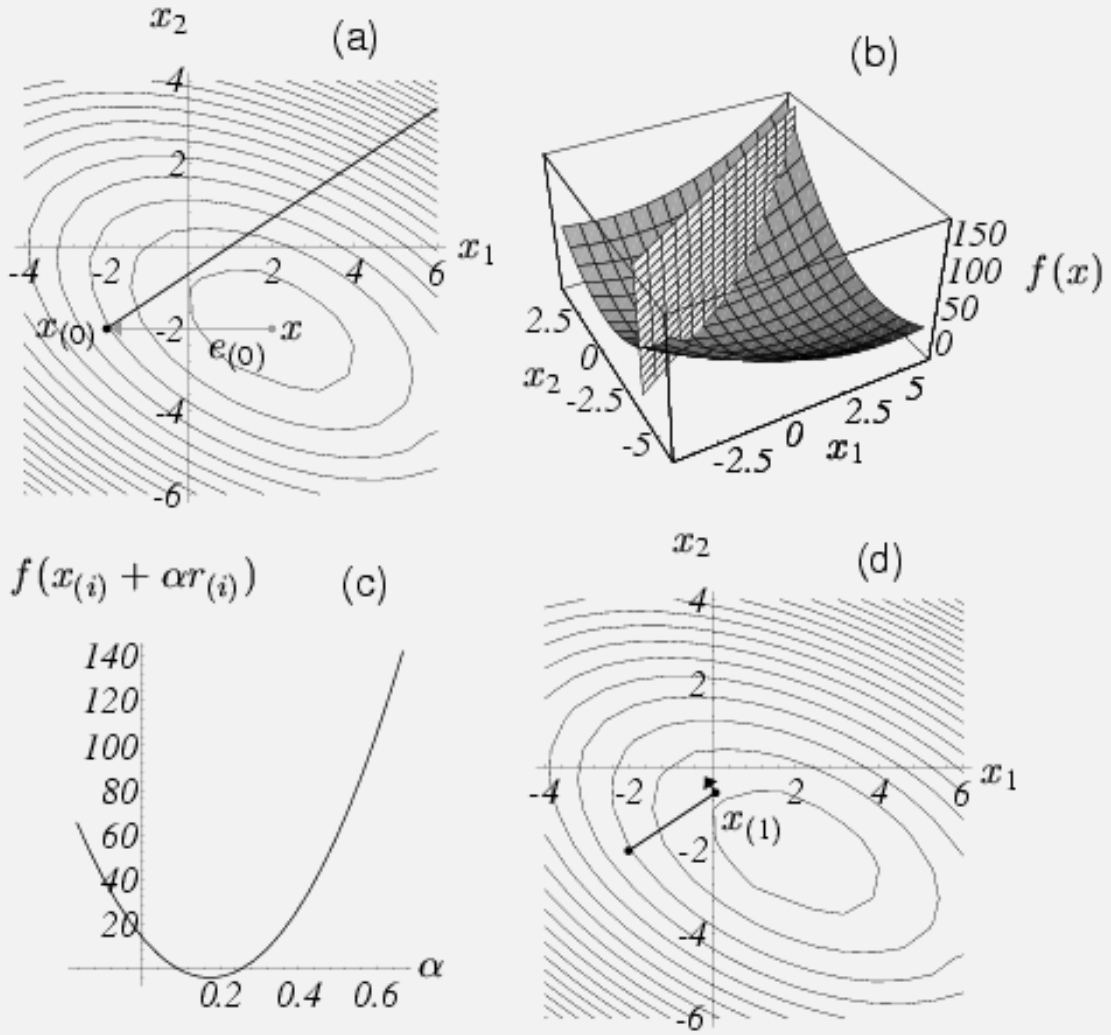
the decrease of ***F(x)*** per unit along h direction

$$\text{greatest gain rate if } \theta = \pi \quad \longrightarrow \quad \mathbf{h}_{sd} = -\mathbf{F}'(\mathbf{x})$$

h_{sd} is a descent direction because $h^T_{sd}\, F'(x) = -F'(x)^2 < 0$

Line Search

$\varphi(\alpha) = F(\mathbf{x}+\alpha\mathbf{h})$, $\mathbf{x}$ and $\mathbf{h}$ fixed, $\alpha \geq 0$.



Find α so that

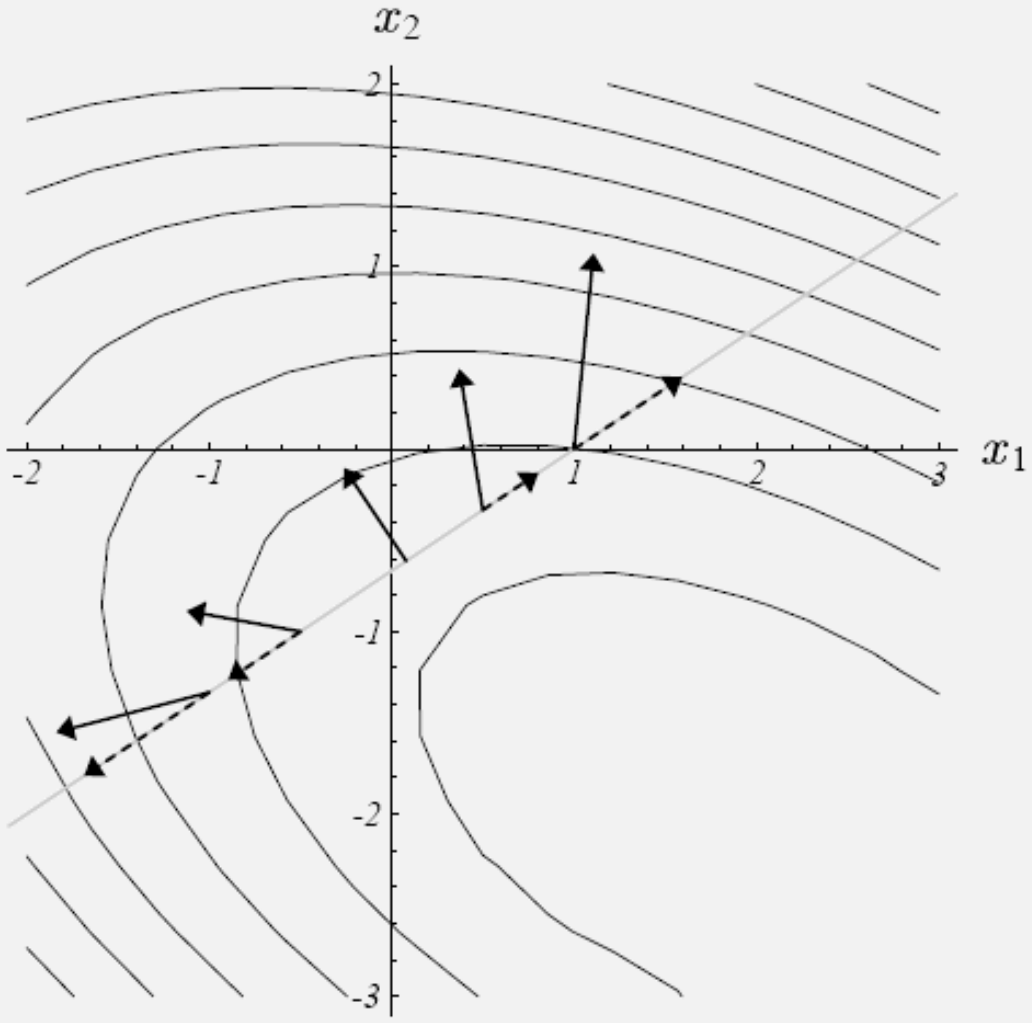
$\varphi(\alpha) = \mathbf{F}(\mathbf{x}_0 + \alpha\mathbf{h})$ is minimum

$$0 = \frac{\partial \varphi(\alpha)}{\partial \alpha} = \frac{\partial \mathbf{F}(\mathbf{x}_0 + \alpha\mathbf{h})}{\partial \alpha}$$

$$= \frac{\partial \mathbf{F}}{\partial \mathbf{x}} \frac{\partial \mathbf{x}}{\partial \alpha} = \mathbf{h}^{\mathrm{T}} \mathbf{F}'(\mathbf{x}_0 + \alpha\mathbf{h})$$

$$\mathbf{h} = -\mathbf{F}'(\mathbf{x}_0)$$

Line Search



$$\mathbf{h}^{\mathrm{T}}\mathbf{F}'(\mathbf{x}_0+\alpha\mathbf{h})=0$$

$$\mathbf{h}=-\mathbf{F}'(\mathbf{x}_0)$$

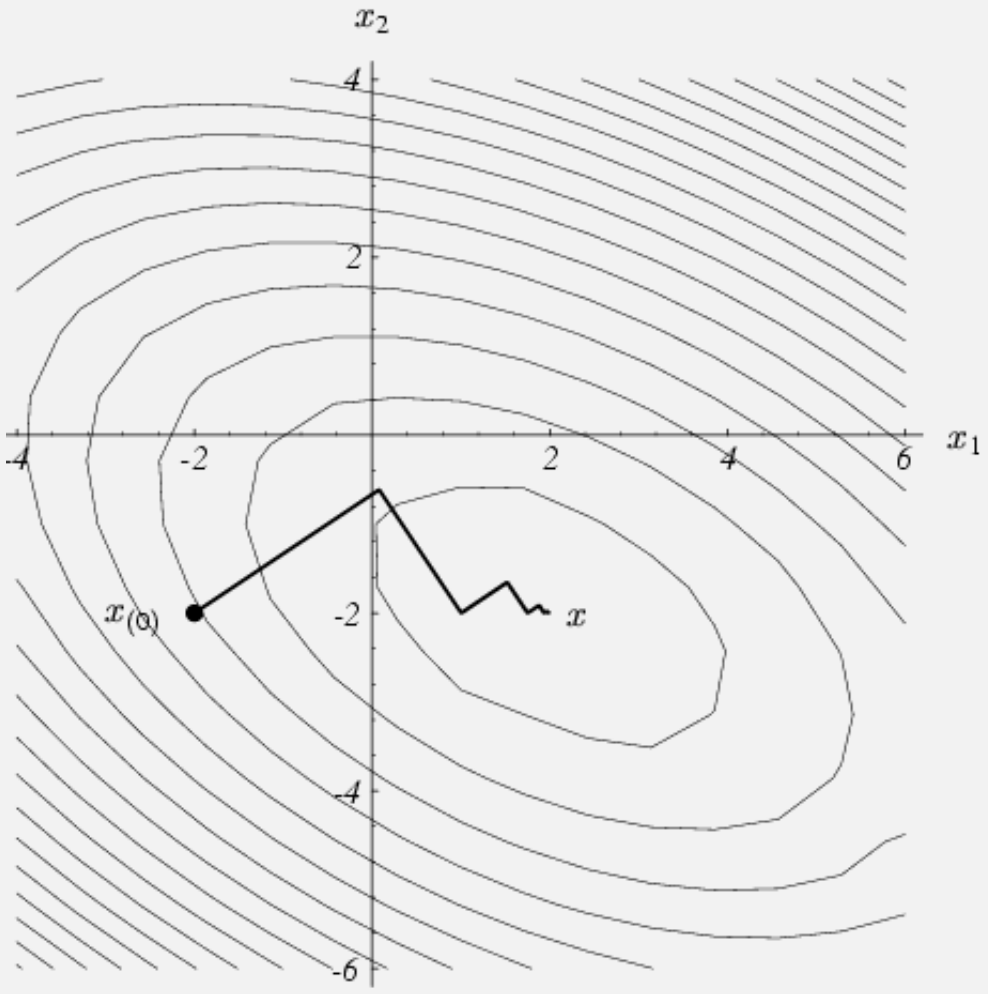
$$\mathbf{h}^{\mathrm{T}}\mathbf{F}'(\mathbf{x}_0+\alpha\mathbf{h})$$

$$=\mathbf{h}^{\mathrm{T}}\left(\mathbf{F}'(\mathbf{x}_0)+\alpha\mathbf{F}''(\mathbf{x}_0)^{\mathrm{T}}\mathbf{h}\right)$$

$$=-\mathbf{h}^{\mathrm{T}}\mathbf{h}+\alpha\mathbf{h}^{\mathrm{T}}\mathbf{H}\mathbf{h}=0$$

$$\alpha=\frac{\mathbf{h}^{\mathrm{T}}\mathbf{h}}{\mathbf{h}^{\mathrm{T}}\mathbf{H}\mathbf{h}}$$

Steepest Descent Method

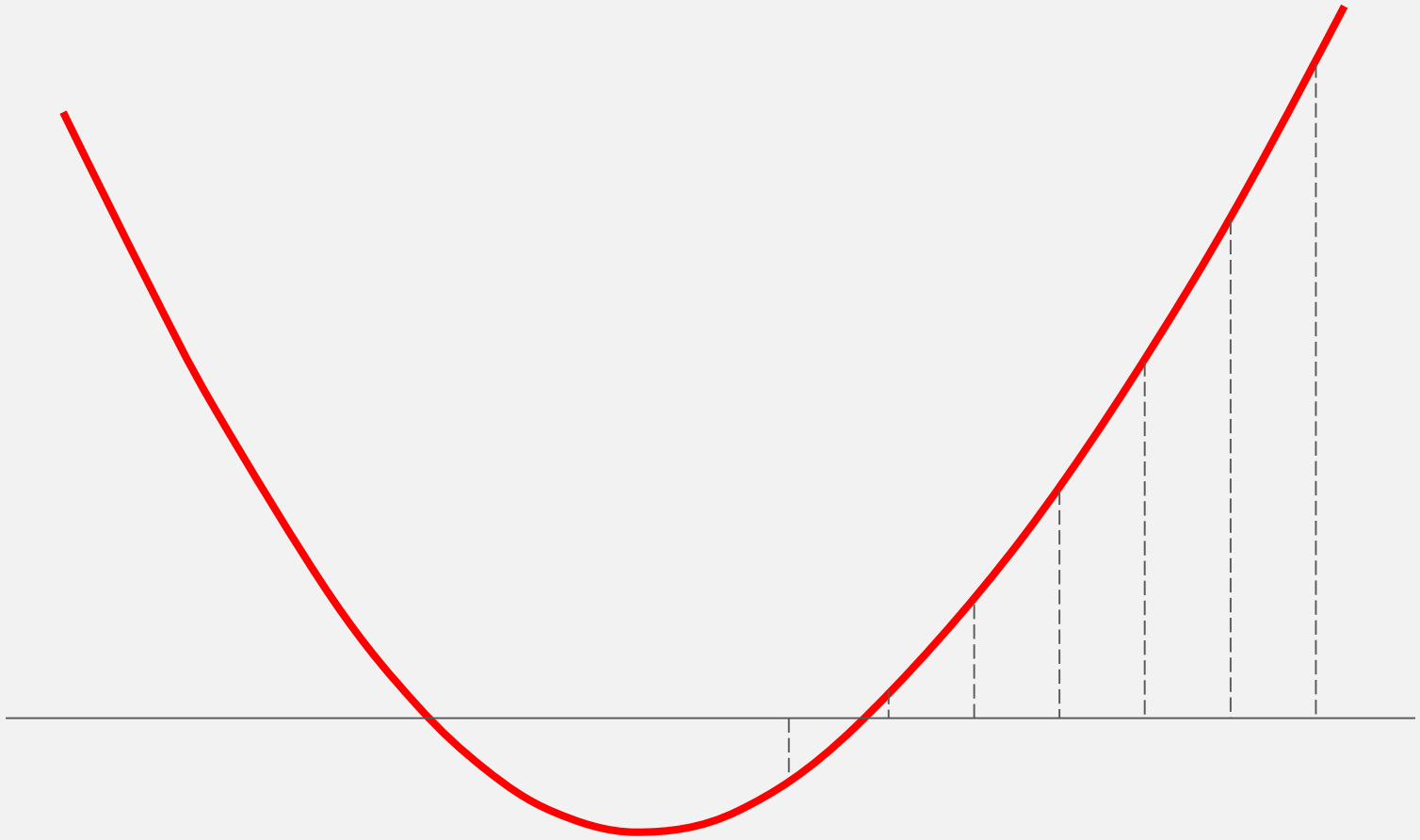


It has good performance in the initial stages of the iterative process.

Converge very slow with a linear rate.

Newton's Method

- Root finding for $f(x)=0$
- March x and test signs
- Determine Δx
  (small→slow; large→ miss)



Newton's Method

- Root finding for *f(x)=0*

Taylor's expansion:

$$f(x_0+\varepsilon) = f(x_0) + f'(x_0)\varepsilon + \frac{1}{2} f''(x_0)\varepsilon^2 + \dots$$

$$0 = \quad f(x_0+\varepsilon) \approx f(x_0) + f'(x_0)\varepsilon$$

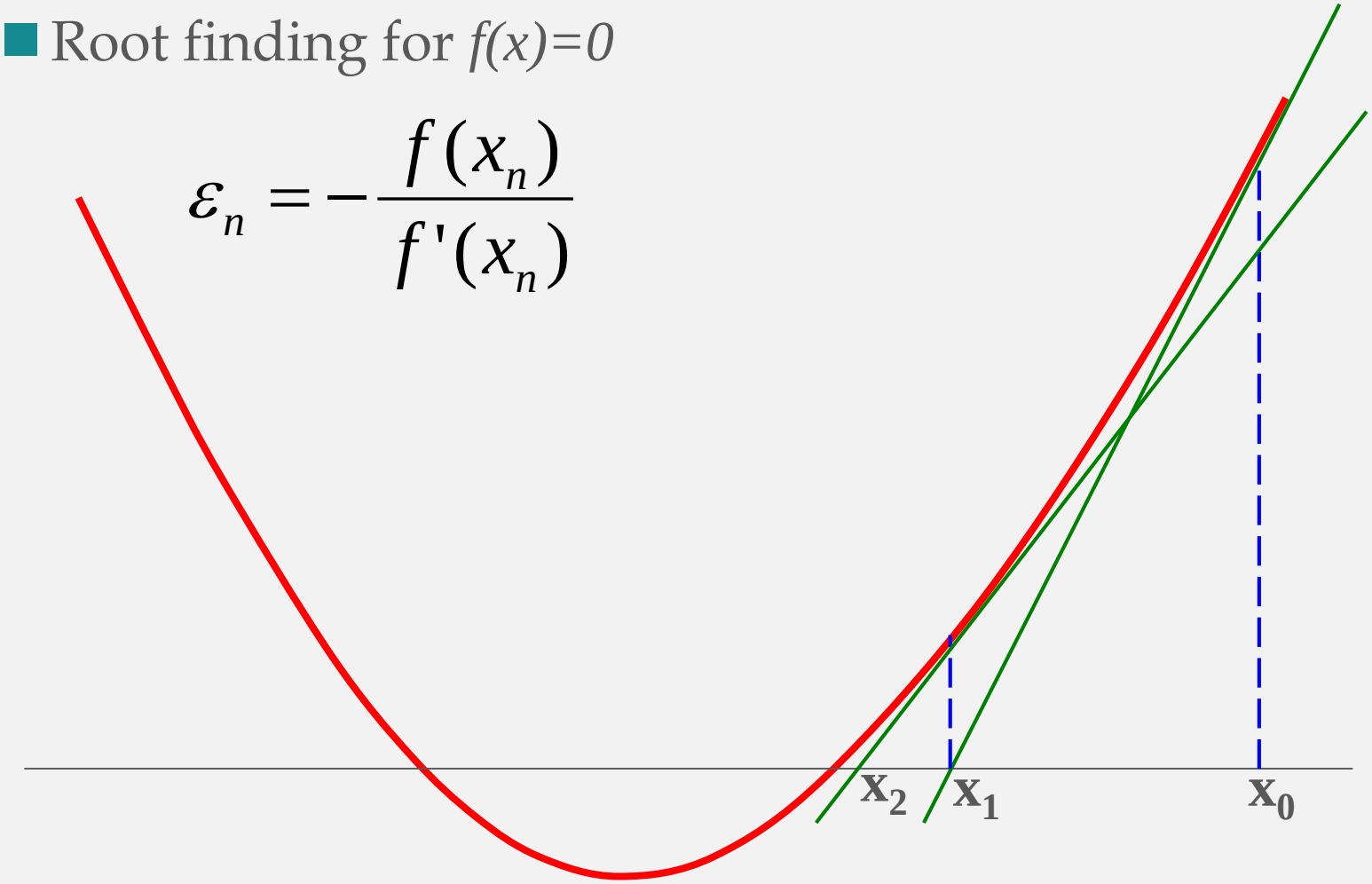
$$\varepsilon = -\frac{f(x_0)}{f'(x_0)}$$

$$x_{n+1} = x_n - \frac{f(x_n)}{f'(x_n)}$$

Newton's Method

- Root finding for *f(x)=0*

$$\varepsilon_n = -\frac{f(x_n)}{f'(x_n)}$$



Newton's Method

$\mathbf{x}^*$ is a stationary point $\longrightarrow$ it satisfies $\mathbf{F}'(\mathbf{x}^*) = \mathbf{0}$.

$$\begin{aligned}\mathbf{F}'(\mathbf{x}+\mathbf{h}) &= \mathbf{F}'(\mathbf{x}) + \mathbf{F}''(\mathbf{x})\mathbf{h} + O(\|\mathbf{h}\|^2) \\ &\simeq \mathbf{F}'(\mathbf{x}) + \mathbf{F}''(\mathbf{x})\mathbf{h} \quad \text{for } \|\mathbf{h}\| \text{ sufficiently small} \\ &= 0\end{aligned}$$

$$\longrightarrow \mathbf{H}\,\mathbf{h}_n = -\mathbf{F}'(\mathbf{x}) \quad \text{with } \mathbf{H} = \mathbf{F}''(\mathbf{x})$$

$$\mathbf{x} := \mathbf{x} + \mathbf{h}_n$$

Suppose that $\mathbf{H}$ is positive definite

$\longrightarrow \mathbf{u}^\top \mathbf{H}\,\mathbf{u} > 0$ for all nonzero $\mathbf{u}$.

$\longrightarrow 0 < \mathbf{h}_n^\top \mathbf{H}\,\mathbf{h}_n = -\mathbf{h}_n^\top \mathbf{F}'(\mathbf{x})$

$\longrightarrow \mathbf{h}_n$ is a descent direction

Newton's Method

$$\mathbf{h} = -\mathbf{H}^{-1}\mathbf{g}$$

- It requires solving a linear system and H is not always positive definite.
- It has good performance in the final stage of the iterative process, where x is close to x*.

Gauss-Newton Method

- H is not always positive definite
  - ➔ Use the approximate Hessian

$$\mathbf{H} \approx \mathbf{J}^{\mathbf{T}}\mathbf{J}$$

  - No need for second derivative
  - H is positive semi-definite

Hybrid Method

$$
\begin{aligned}
&\textbf{if } \mathbf{F}''(\mathbf{x}) \text{ is positive definite} \\
&\quad \mathbf{h} := \mathbf{h}_{\mathrm{n}} \\
&\textbf{else} \\
&\quad \mathbf{h} := \mathbf{h}_{\mathrm{sd}} \\
&\mathbf{x} := \mathbf{x} + \alpha \mathbf{h}
\end{aligned}
$$

This needs to calculate second-order derivative which might not be available.

Levenberg-Marquardt Method

- LM can be thought of as a combination of steepest descent and the Newton method.
  - When the current solution is far from the correct one, the algorithm behaves like a steepest descent method: slow, but guaranteed to converge.
  - When the current solution is close to the correct solution, it becomes a Newton's method.

Jacobian matrix

- The Jacobian matrix is the matrix of all first-order partial derivatives of a vector-valued function. $F : \mathbf{R}^n \to \mathbf{R}^m$

$$F(x_1, x_2, \ldots x_n)$$
$$= (f_1(x_1, x_2, \ldots x_n), f_2(x_1, x_2, \ldots x_n), \ldots f_m(x_1, x_2, \ldots x_n))$$

$$J_F(x_1, x_2, \ldots x_n) \quad \text{or} \quad \frac{\partial(f_1, f_2, \ldots f_m)}{\partial(x_1, x_2, \ldots x_n)} = \begin{bmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & \ddots & \vdots \\ \frac{\partial f_m}{\partial x_1} & \cdots & \frac{\partial f_m}{\partial x_n} \end{bmatrix}$$

$$F(\mathbf{x} + \Delta\mathbf{x}) \approx F(\mathbf{x}) + J_F(\mathbf{x})\Delta\mathbf{x}$$

Levenberg-Marquardt Method

Given a set of measurements $\mathbf{x}$, try to find the best parameter vector $\mathbf{p}$ so that the squared distance $\varepsilon^T \varepsilon$ is minimal.

Here, $\varepsilon = \mathbf{x} - \hat{\mathbf{x}}$, with $\hat{\mathbf{x}} = f(\mathbf{p})$.

For a small $||\delta_\mathbf{p}||$, $f(\mathbf{p} + \delta_\mathbf{p}) \approx f(\mathbf{p}) + \mathbf{J}\delta_\mathbf{p}$

$\mathbf{J}$ is the Jacobian matrix $\frac{\partial f(\mathbf{p})}{\partial \mathbf{p}}$

it is required to find the $\delta_\mathbf{p}$ that minimizes the quantity

$$||\mathbf{x} - f(\mathbf{p} + \delta_\mathbf{p})|| \approx ||\mathbf{x} - f(\mathbf{p}) - \mathbf{J}\delta_\mathbf{p}|| = ||\epsilon - \mathbf{J}\delta_\mathbf{p}||$$

$$\mathbf{J}^T\mathbf{J}\delta_\mathbf{p} = \mathbf{J}^T\epsilon$$

$$\mathbf{N}\delta_\mathbf{p} = \mathbf{J}^T\epsilon$$

damping term

$$\mathbf{N}_{ii} = \mu + \left[\mathbf{J}^T\mathbf{J}\right]_{ii}$$

Levenberg-Marquardt Method

$$(\mathbf{J}^{\mathbf{T}}\mathbf{J} + \mu\mathbf{I})\mathbf{h} = -\mathbf{g}$$

- μ=0 → Newton's method
- μ→∞ → steepest descent method

- Strategy for choosing μ
  - Start with some small μ
  - If F is not reduced, keep trying larger μ until it does
  - If F is reduced, accept it and reduce μ for the next iteration

Structure from Motion /Matchmove

2017 Fall

National Cheng Kung University

Instructor: Min-Chun Hu 胡敏君

Move Matching

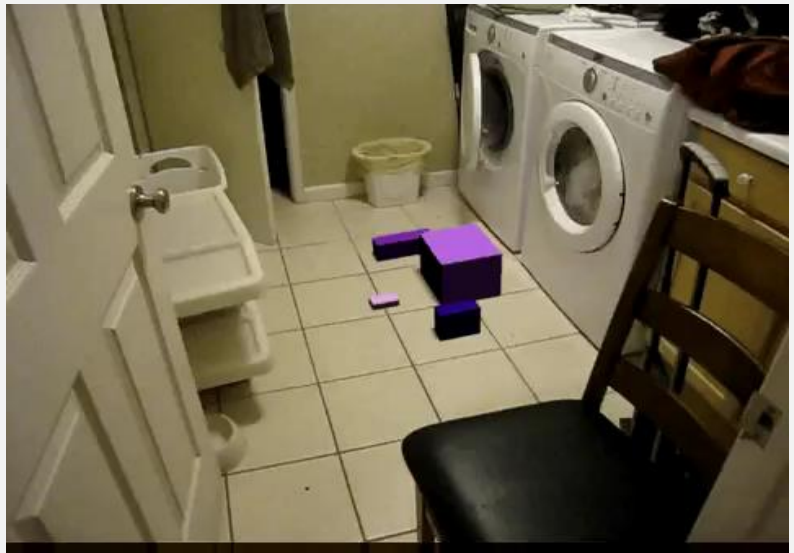
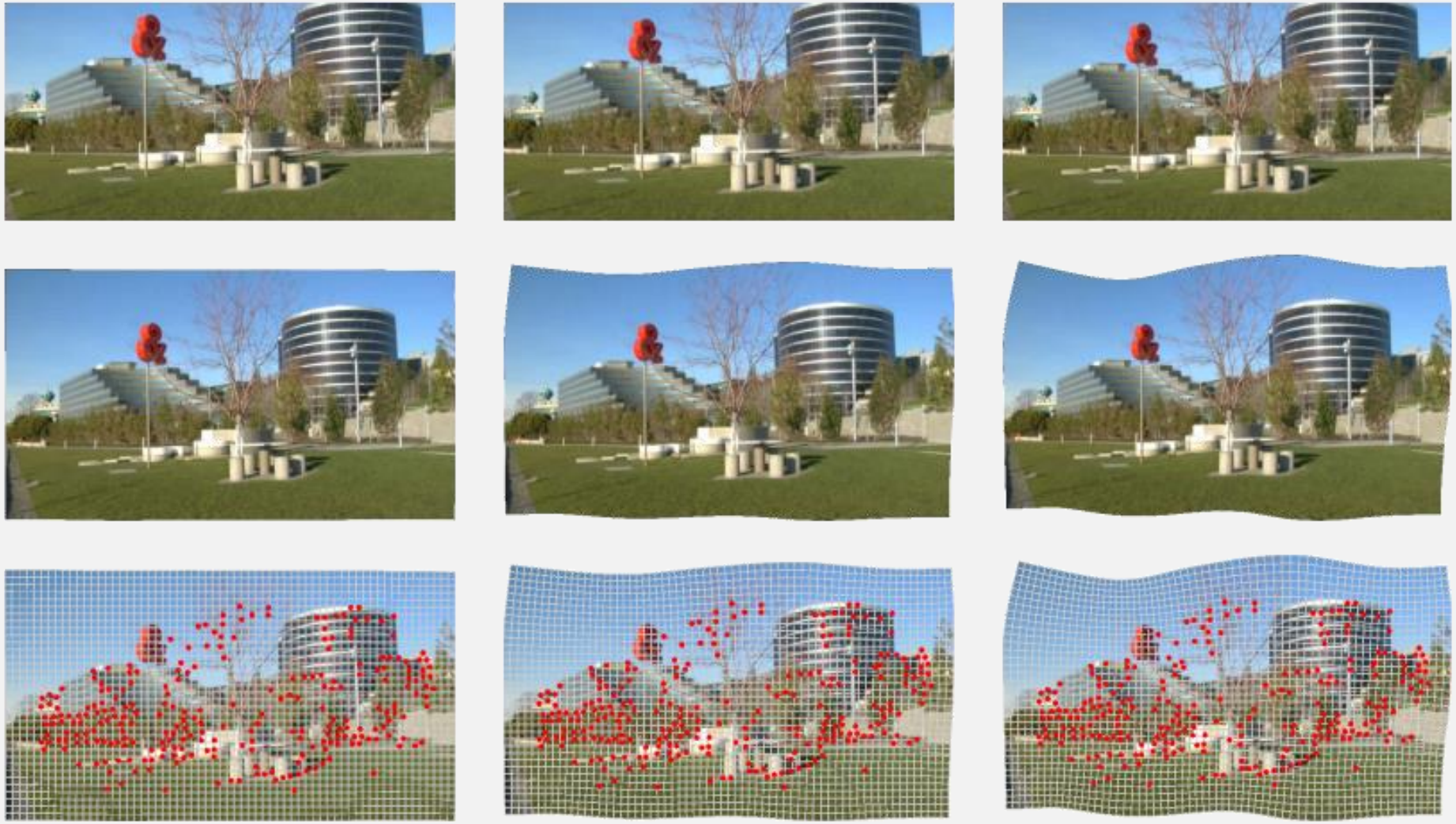
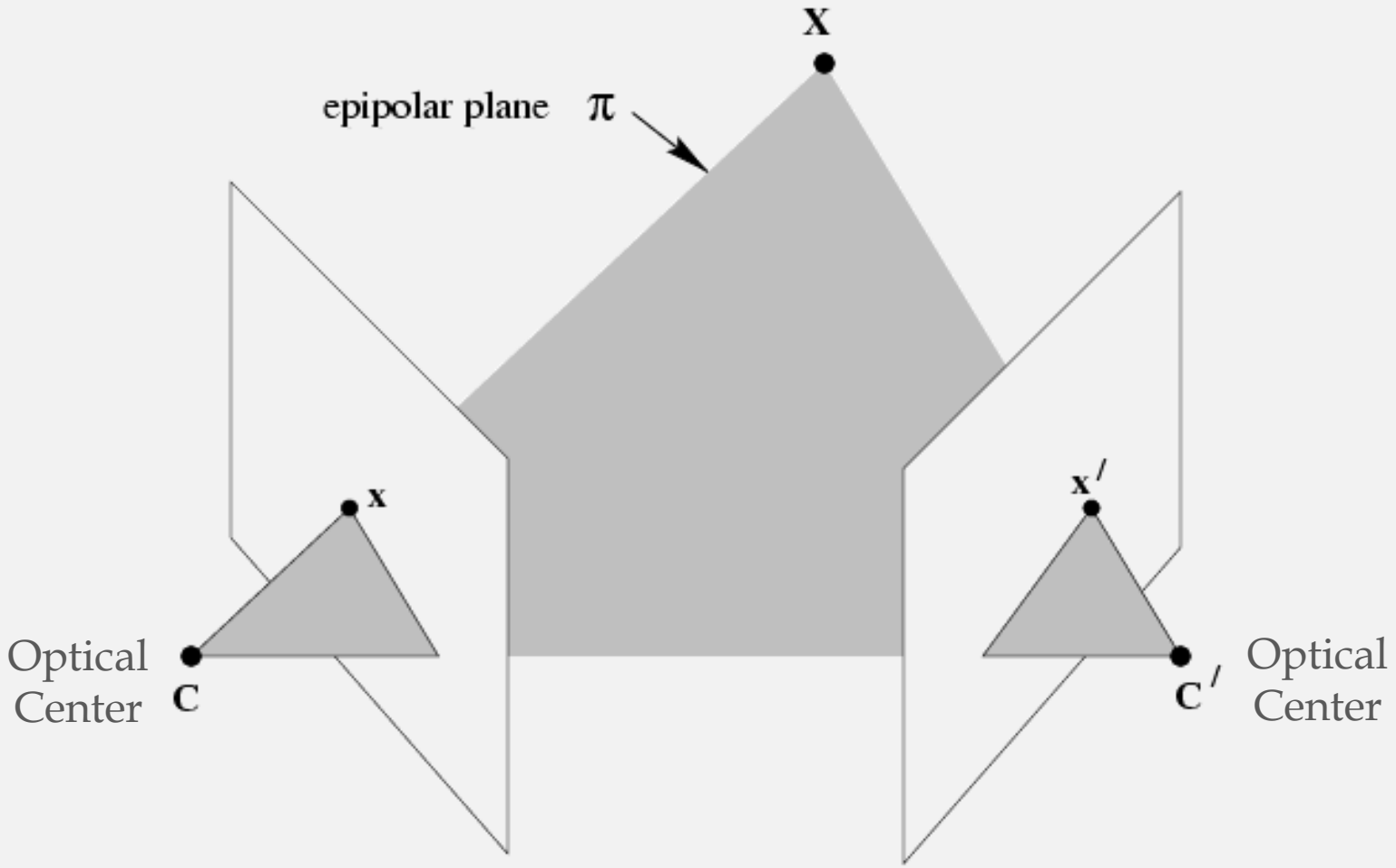


Photo Tourism

Video Stabilization

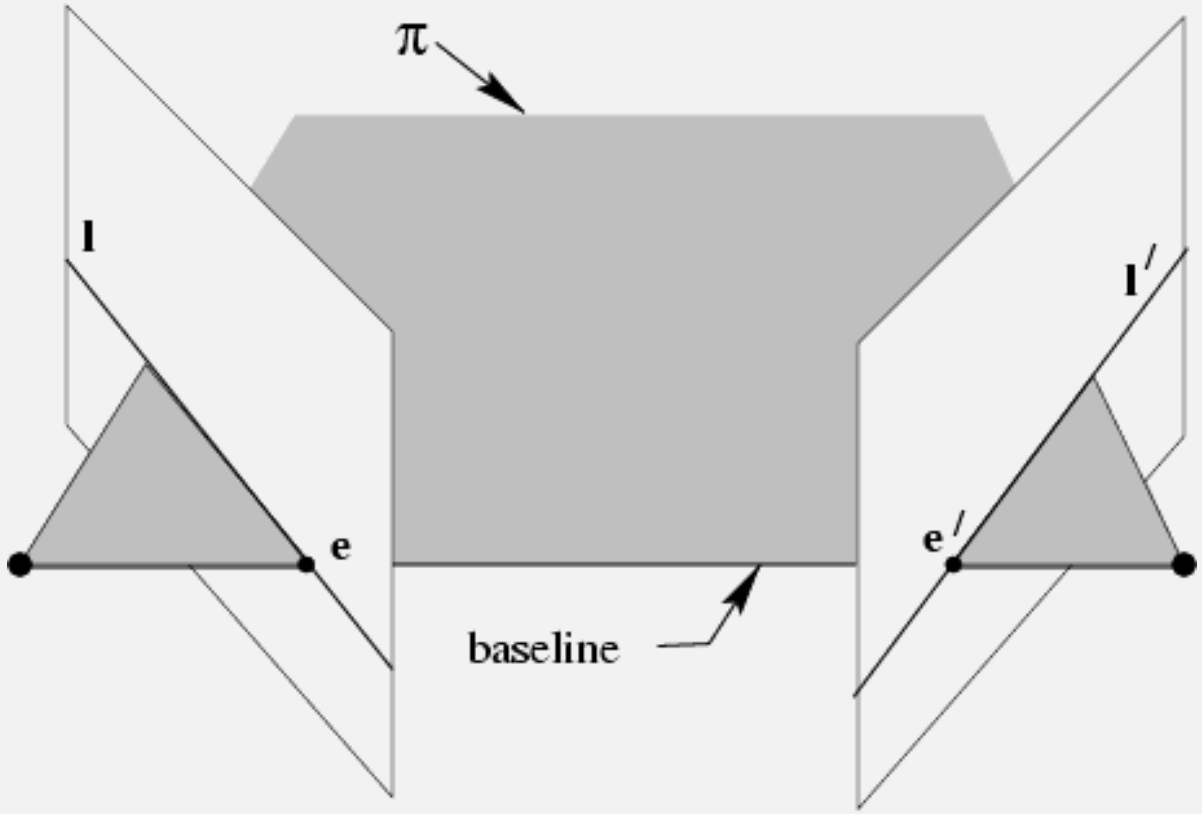


Epipolar geometry & Fundamental Matrix



C,C′,x,x′ and X are coplanar

Epipolar Geometry



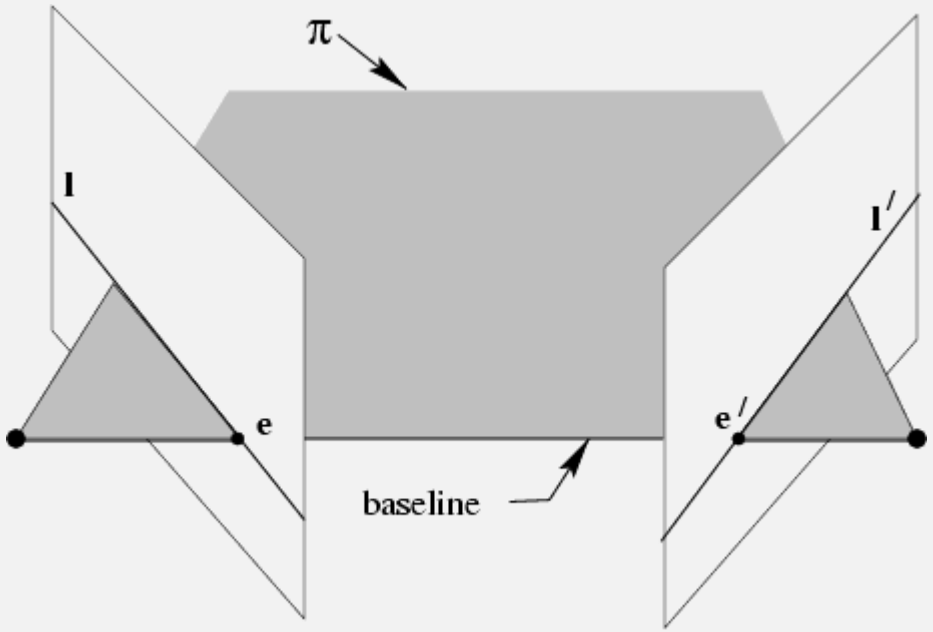
All points on π project on l and l′

Epipolar Geometry

epipolar pole

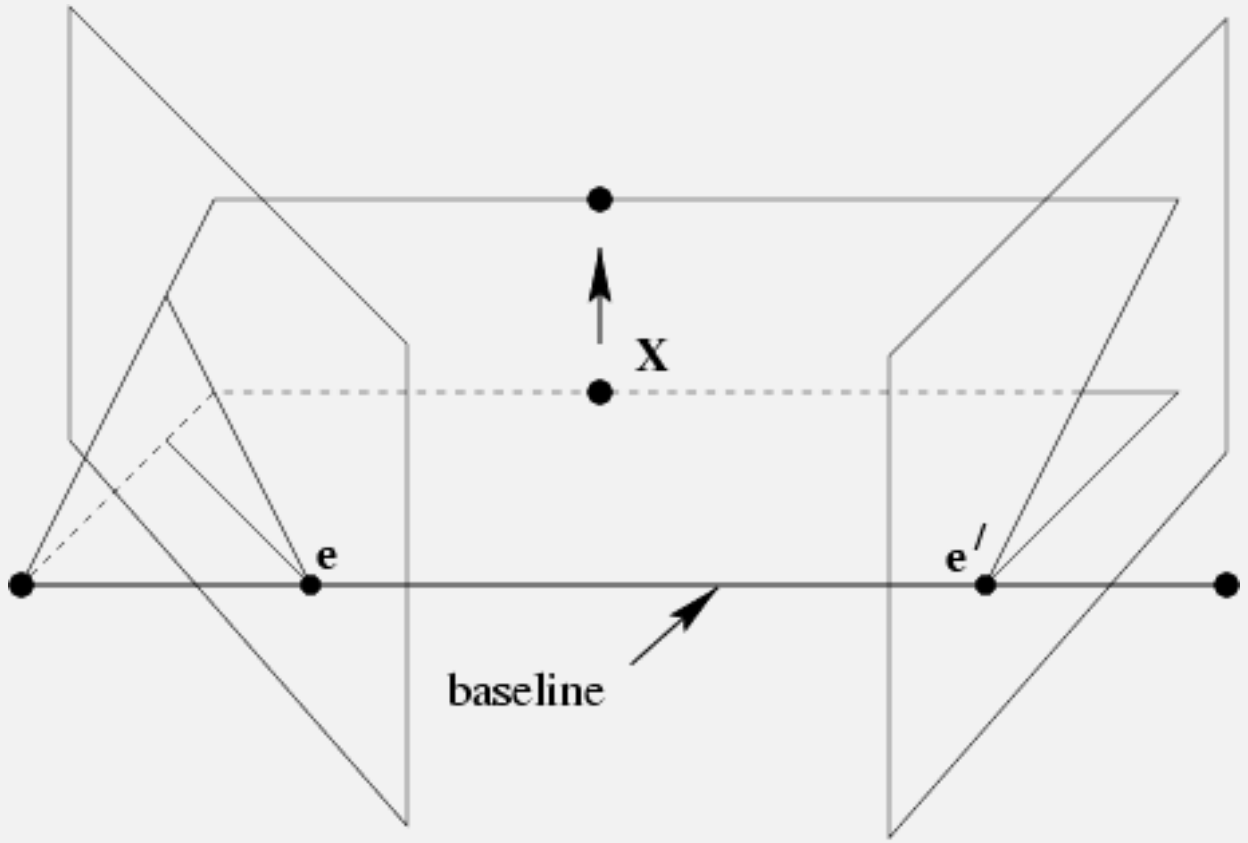
= intersection of baseline with image plane

= projection of projection center in other image



epipolar plane = plane containing baseline
epipolar line = intersection of epipolar plane with image

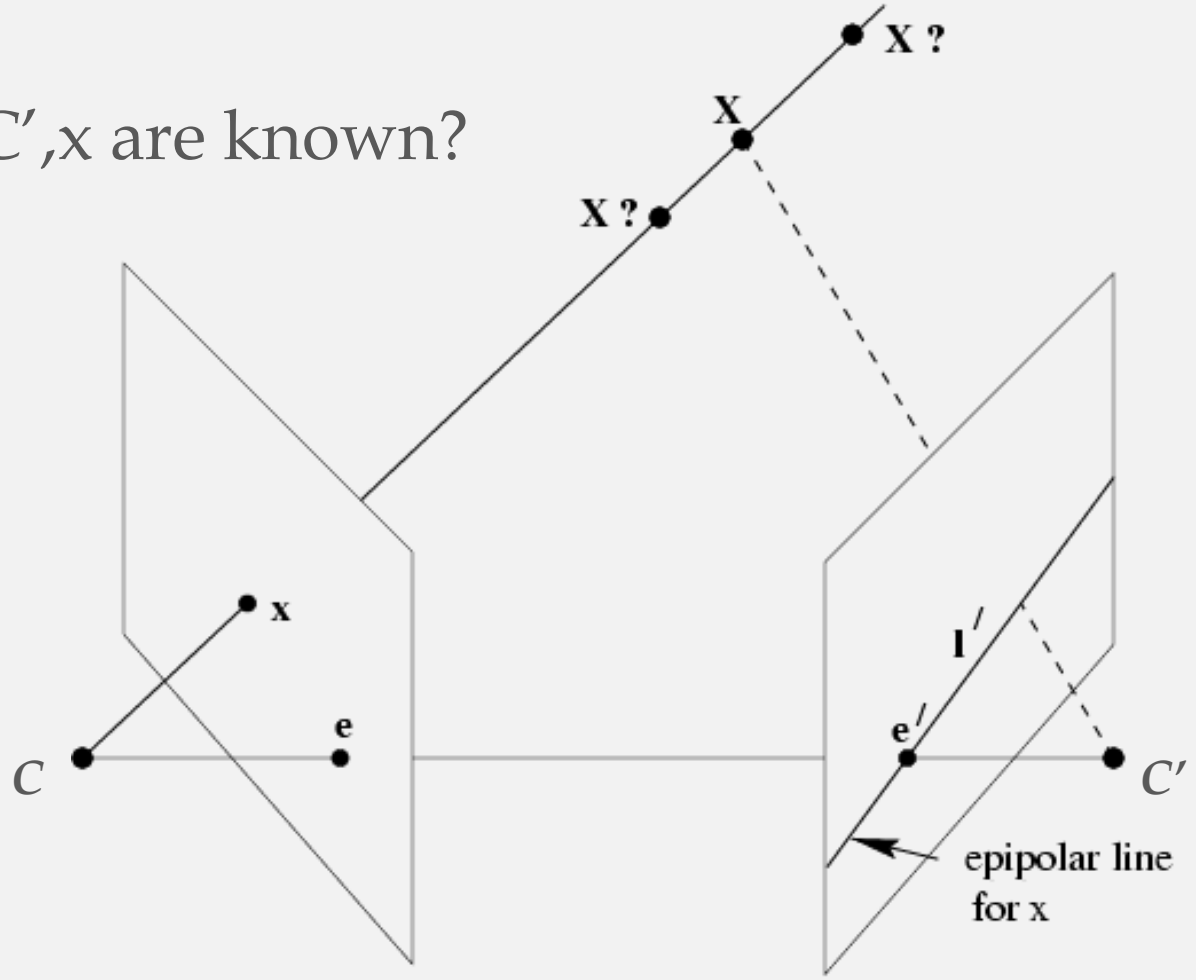
Epipolar Geometry



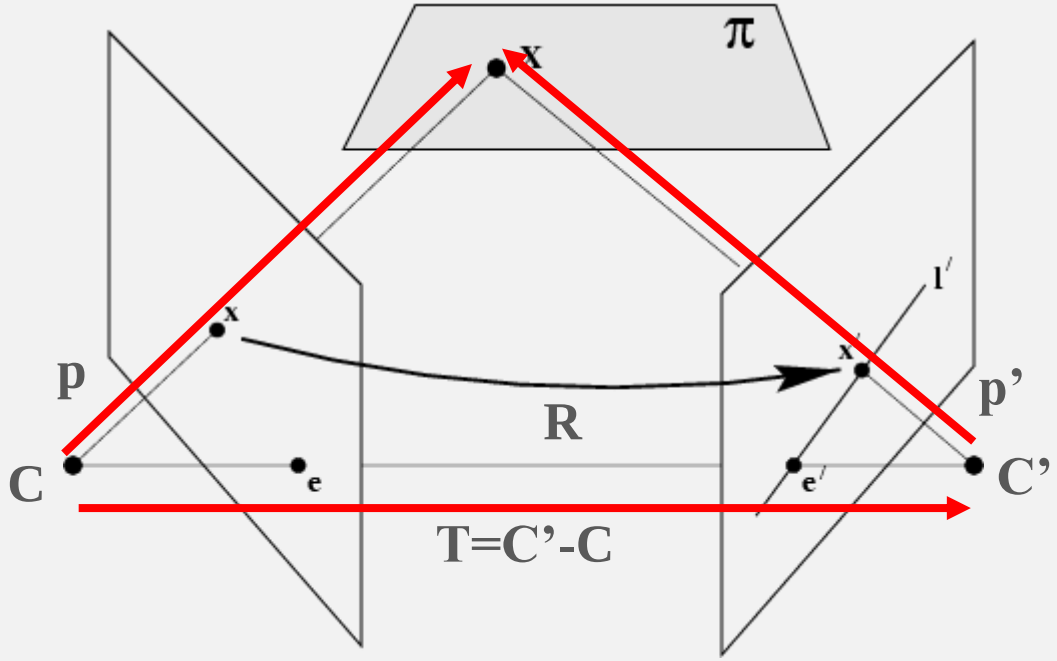
Family of planes π and lines *l* and *l'* intersect at *e* and *e'*

Epipolar Geometry

What if only C,C′,x are known?



Fundamental Matrix F



Two reference frames are related via the extrinsic parameters

$$\mathbf{p} = \mathbf{R}\mathbf{p}' + \mathbf{T}$$

Fundamental Matrix F

$$\mathbf{p} = \mathbf{R}\mathbf{p}' + \mathbf{T}$$

Multiply both sides by $\mathbf{p}^T [\mathbf{T}]_\times$

$$[\mathbf{T}]_\times = \begin{bmatrix} 0 & -T_z & T_y \\ T_z & 0 & -T_x \\ -T_y & T_x & 0 \end{bmatrix}$$

$$\mathbf{p}^T [\mathbf{T}]_\times \mathbf{p} = \mathbf{p}^T [\mathbf{T}]_\times (\mathbf{R}\mathbf{p}' + \mathbf{T})$$

$$\Rightarrow \quad 0 = \mathbf{p}^T [\mathbf{T}]_\times \mathbf{R}\mathbf{p}'$$

$$\Rightarrow \quad \mathbf{p}^T \mathbf{E} \mathbf{p}' = 0$$

essential matrix

Fundamental Matrix F

$$\mathbf{p}^{T}\mathbf{E}\mathbf{p}' = 0$$

Let $\mathbf{M}$ and $\mathbf{M}'$ be the intrinsic matrices, then

$$\mathbf{p} = \mathbf{M}^{-1}\mathbf{x} \qquad \mathbf{p}' = \mathbf{M}'^{-1}\mathbf{x}'$$

$$\Rightarrow \quad (\mathbf{M}^{-1}\mathbf{x})^{T}\mathbf{E}(\mathbf{M}'^{-1}\mathbf{x}') = 0$$

$$\Rightarrow \quad \mathbf{x}^{T}\boxed{\mathbf{M}^{-T}\mathbf{E}\mathbf{M}'^{-1}}\mathbf{x}' = 0$$

$$\Rightarrow \quad \mathbf{x}^{T}\boxed{\mathbf{F}}\mathbf{x}' = 0$$

fundamental matrix

Fundamental Matrix F

- The fundamental matrix is the algebraic representation of epipolar geometry

- The fundamental matrix satisfies the condition that for any pair of corresponding points $x \leftrightarrow x'$ in the two images

$$x^T F x' = 0 \qquad \left(x^T l = 0\right)$$

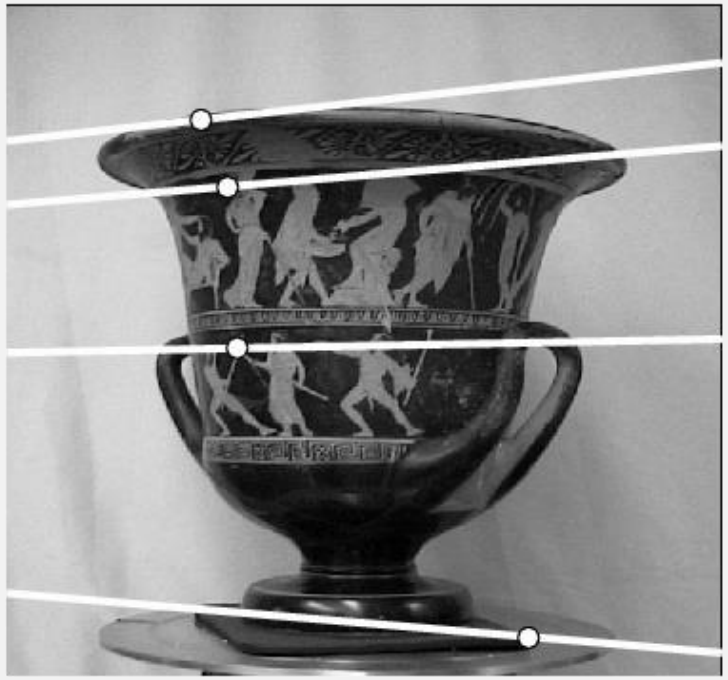
Fundamental Matrix F

- F is the unique 3x3 rank 2 matrix that satisfies $x^T F x' = 0$ for all $x \leftrightarrow x'$

1. Transpose: if F is fundamental matrix for (P,P'), then F^T is fundamental matrix for (P',P)
2. Epipolar lines: $l = Fx'$ & $l' = F^T x$
3. Epipoles: on all epipolar lines, thus $e^T F x' = 0$, $\forall x' \Rightarrow e^T F = 0$, similarly $Fe' = 0$
4. F has 7 d.o.f. , i.e. 3x3-1(homogeneous)-1(rank2)
5. F is a correlation, projective mapping from a point x to a line $l = Fx'$ (not a proper correlation, i.e. not invertible)

Fundamental Matrix F

- It can be used for
  - Simplifies matching
  - Allows to detect wrong matches



Estimation of F : 8-point algorithm

- The fundamental matrix F is defined by

$$\mathbf{x}^{\mathrm{T}}\mathbf{F}\mathbf{x}'=0$$

for any pair of matches x and x′ in two images.

- Let x=$(u,v,1)^{\mathrm{T}}$ and x′=$(u',v',1)^{\mathrm{T}}$, $\mathbf{F}=\begin{bmatrix} f_{11} & f_{12} & f_{13} \\ f_{21} & f_{22} & f_{23} \\ f_{31} & f_{32} & f_{33} \end{bmatrix}$

each match gives a linear equation :

$$uu'f_{11}+uv'f_{12}+uf_{13}+vu'f_{21}+vv'f_{22}+vf_{23}+u'f_{31}+v'f_{32}+f_{33}=0$$

Estimation of F : 8-point algorithm

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

- In reality, instead of solving $\mathbf{Af} = 0$, we seek $\mathbf{f}$ to minimize $\|\mathbf{Af}\|$ subj. $\|\mathbf{f}\| = 1$. Find the vector corresponding to the least singular value.

Estimation of F : 8-point algorithm

- To enforce that F is of rank 2, F is replaced by F′ that minimizes $\|\mathbf{F}-\mathbf{F}'\|$ subject to $\det \mathbf{F}' = 0$.
- It is achieved by SVD. Let $\mathbf{F} = \mathbf{U}\Sigma\mathbf{V}^{\mathrm{T}}$, where

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

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

Estimation of F : 8-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';
```

Estimation of F : 8-point algorithm

- Pros: it is linear, easy to implement and fast
- Cons: susceptible to noise

Problem with 8-point algorithm

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

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

Orders of magnitude difference between column of data matrix

→ least-squares yields poor results

Normalized 8-point algorithm

1. Transform input by $\hat{\mathbf{x}}_i = \mathbf{T}\mathbf{x}_i$, $\hat{\mathbf{x}}_i' = \mathbf{T}\mathbf{x}_i'$
2. Call 8-point on $\hat{\mathbf{x}}_i, \hat{\mathbf{x}}_i'$ to obtain $\hat{\mathbf{F}}$
3. $\mathbf{F} = \mathbf{T}'^{\mathrm{T}}\hat{\mathbf{F}}\mathbf{T}$

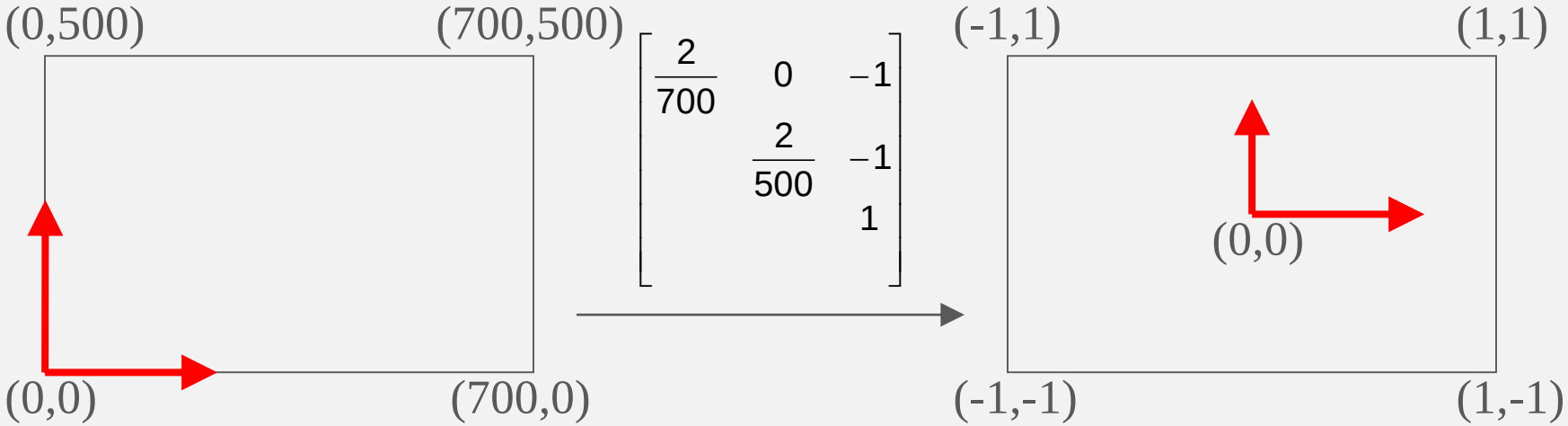
$$\mathbf{x}'^{\mathrm{T}}\mathbf{F}\mathbf{x} = 0$$

$$\hat{\mathbf{x}}'^{\mathrm{T}}\mathbf{T}'^{-\mathrm{T}}\mathbf{F}\mathbf{T}^{-1}\hat{\mathbf{x}} = 0$$

$$\hat{\mathbf{F}}$$

Normalized 8-point algorithm

- Normalized least squares yields good results
- Transform image to ~[-1,1]x[-1,1]



Normalized 8-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;
```

Normalization

```
function [newpts, T] = normalise2dpts(pts)

    c = mean(pts(1:2,:)')';    % Centroid
    newp(1,:) = pts(1,:)-c(1); % Shift origin to centroid.
    newp(2,:) = pts(2,:)-c(2);

    meandist = mean(sqrt(newp(1,:).^2 + newp(2,:).^2));
    scale = sqrt(2)/meandist;

    T = [scale     0    -scale*c(1)
           0     scale  -scale*c(2)
           0       0          1      ];
    newpts = T*pts;
```

RANSAC

repeat

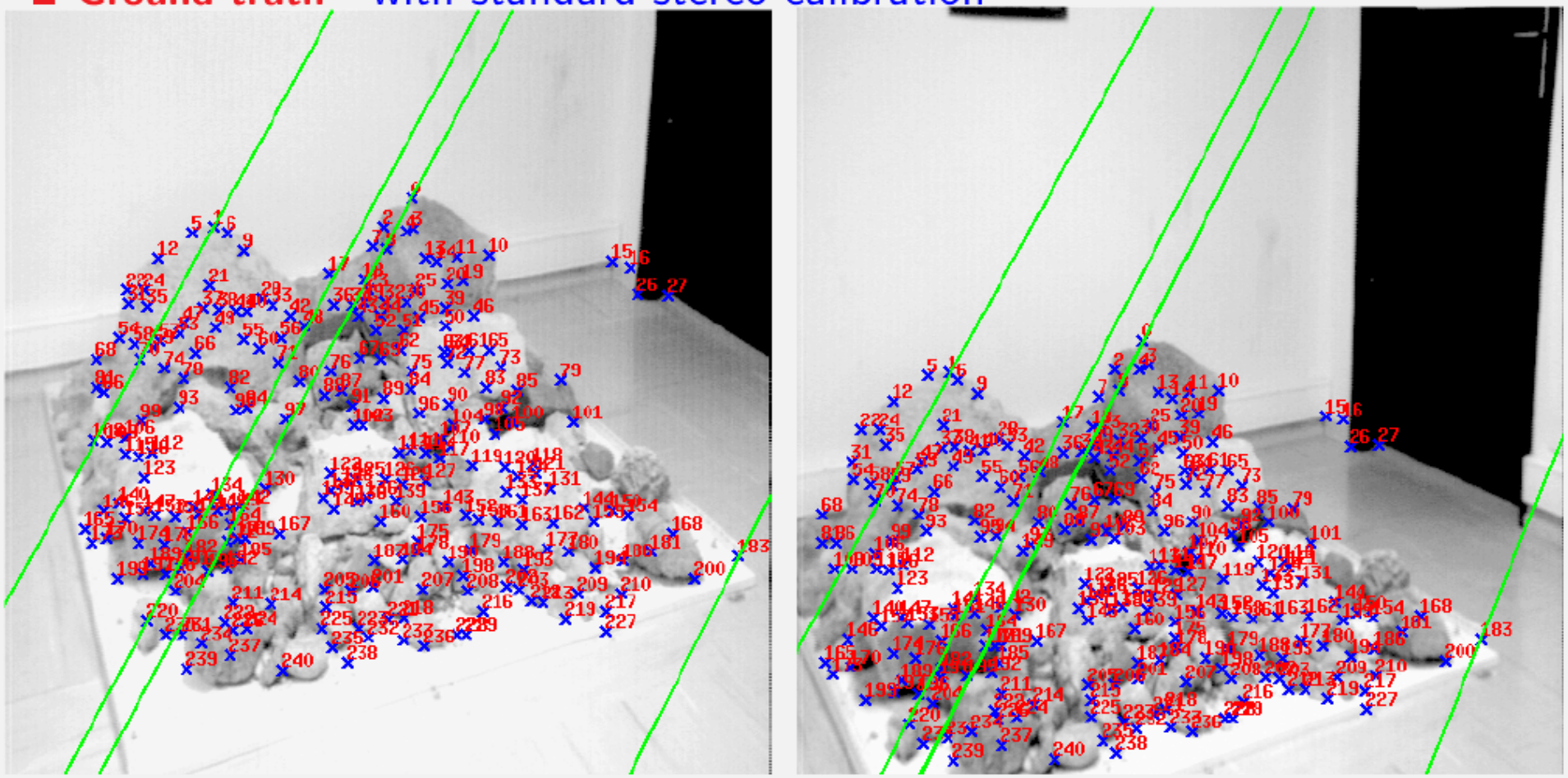
- select minimal sample (8 matches)
- compute solution(s) for F
- determine inliers

until $\Gamma(\#inliers, \#samples) > 95\%$ or too many times

compute F based on all inliers

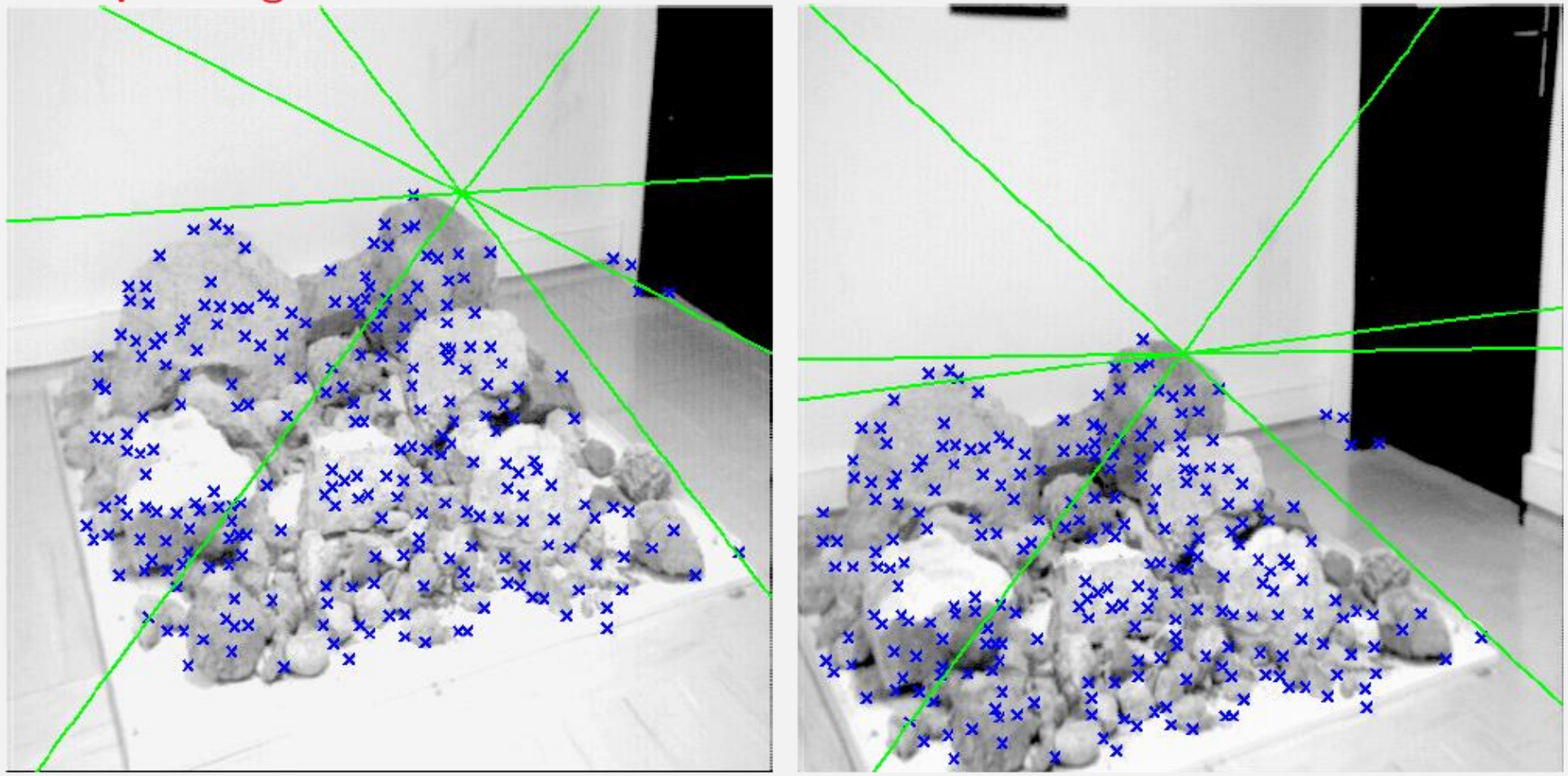
Results (ground truth)

- **Ground truth** with standard stereo calibration



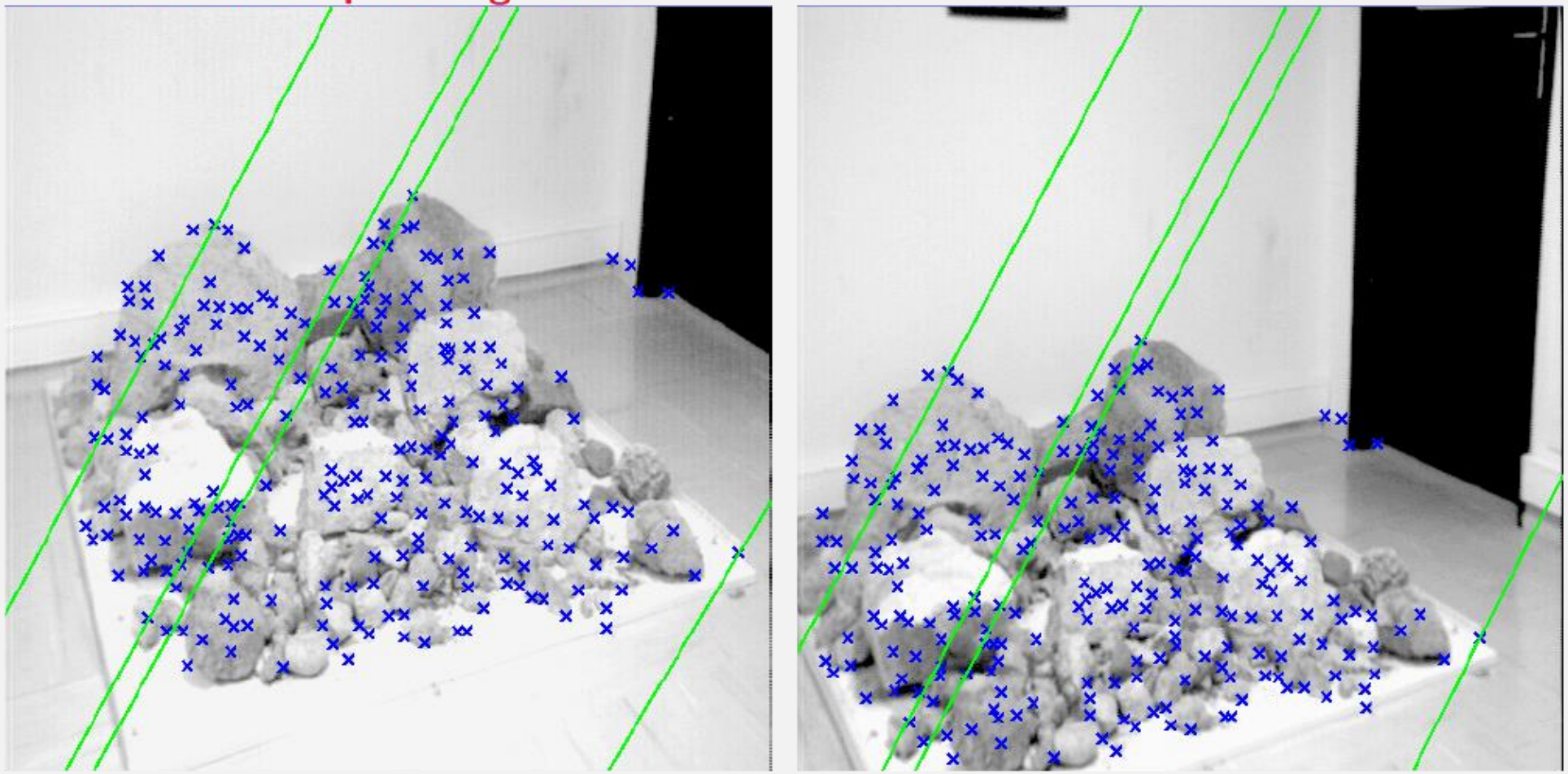
Results (8-point algorithm)

- 8-point algorithm

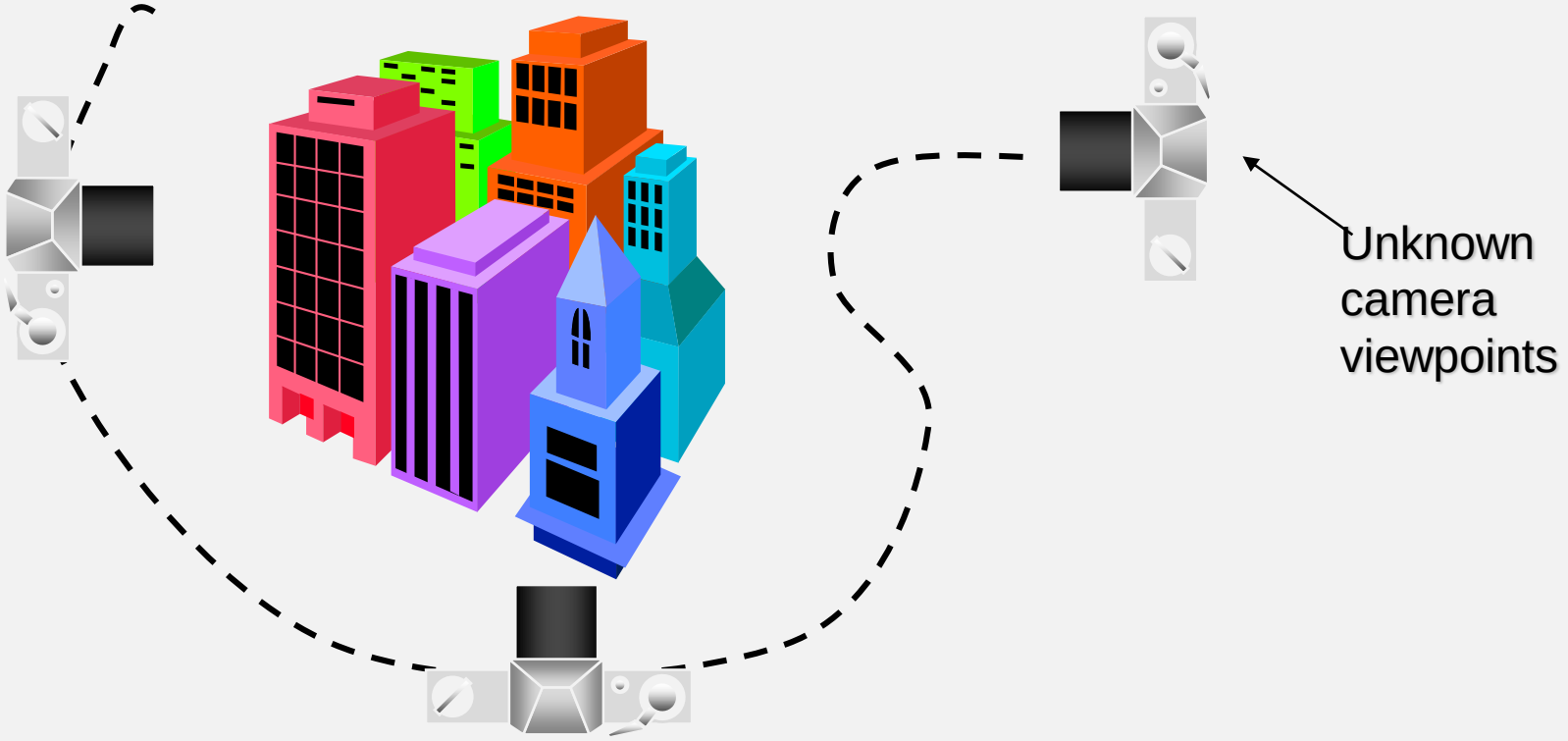


Results (normalized 8-point algorithm)

- Normalized 8-point algorithm

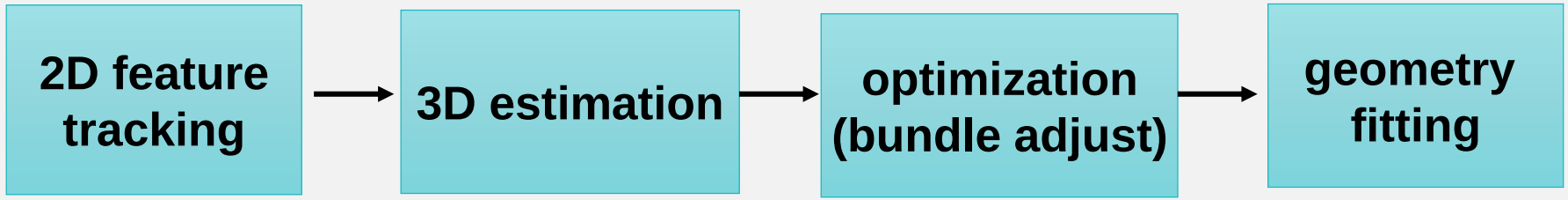


Structure from Motion

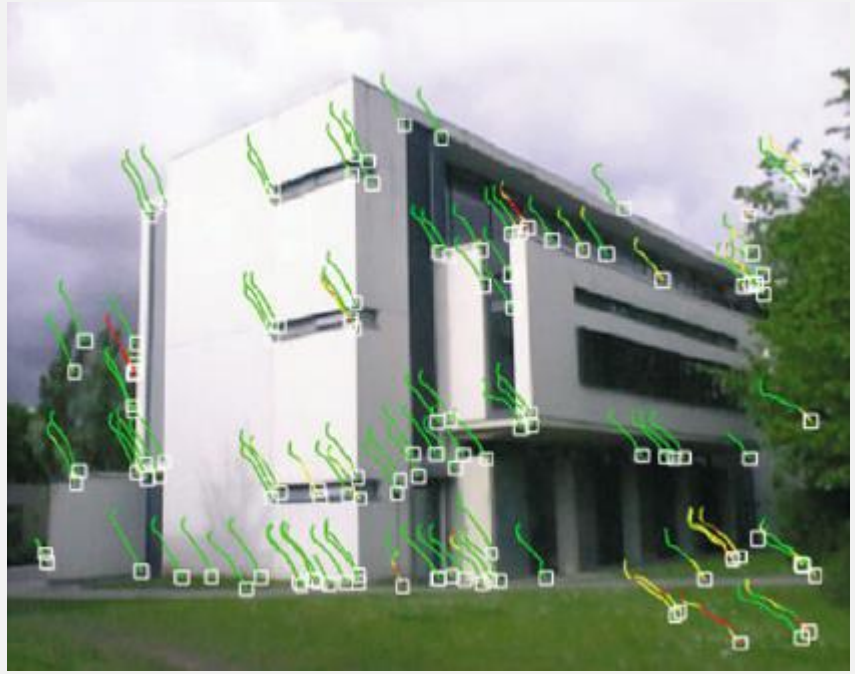


- structure from motion: automatic recovery of **camera motion** and **scene structure** from two or more images.
- It is a self calibration technique and called *automatic camera tracking* or *matching move* **or** *SLAM*.

SFM Pipeline



- Step 1: Track Features
  - Detect good features, Shi & Tomasi, SIFT
  - Find correspondences between frames
    - Lucas & Kanade-style motion estimation
    - window-based correlation
    - SIFT matching

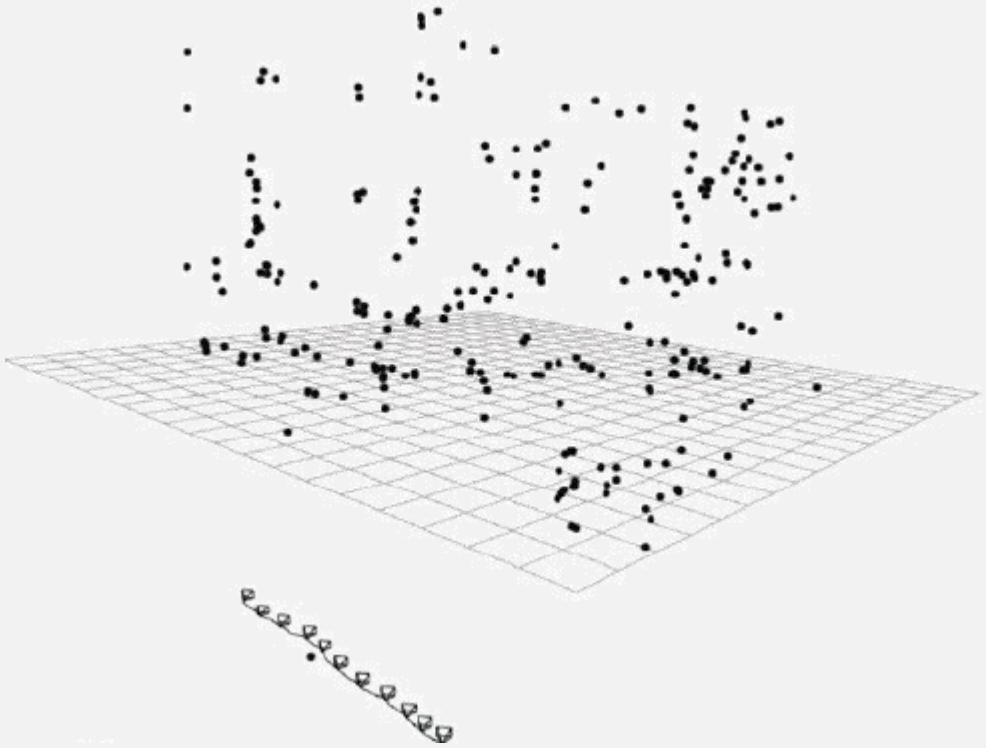


SFM Pipeline

- Step 2: Estimate Motion and Structure
  - Simplified projection model, e.g., **[Tomasi 92]**
  - 2 or 3 views at a time **[Hartley 00]**

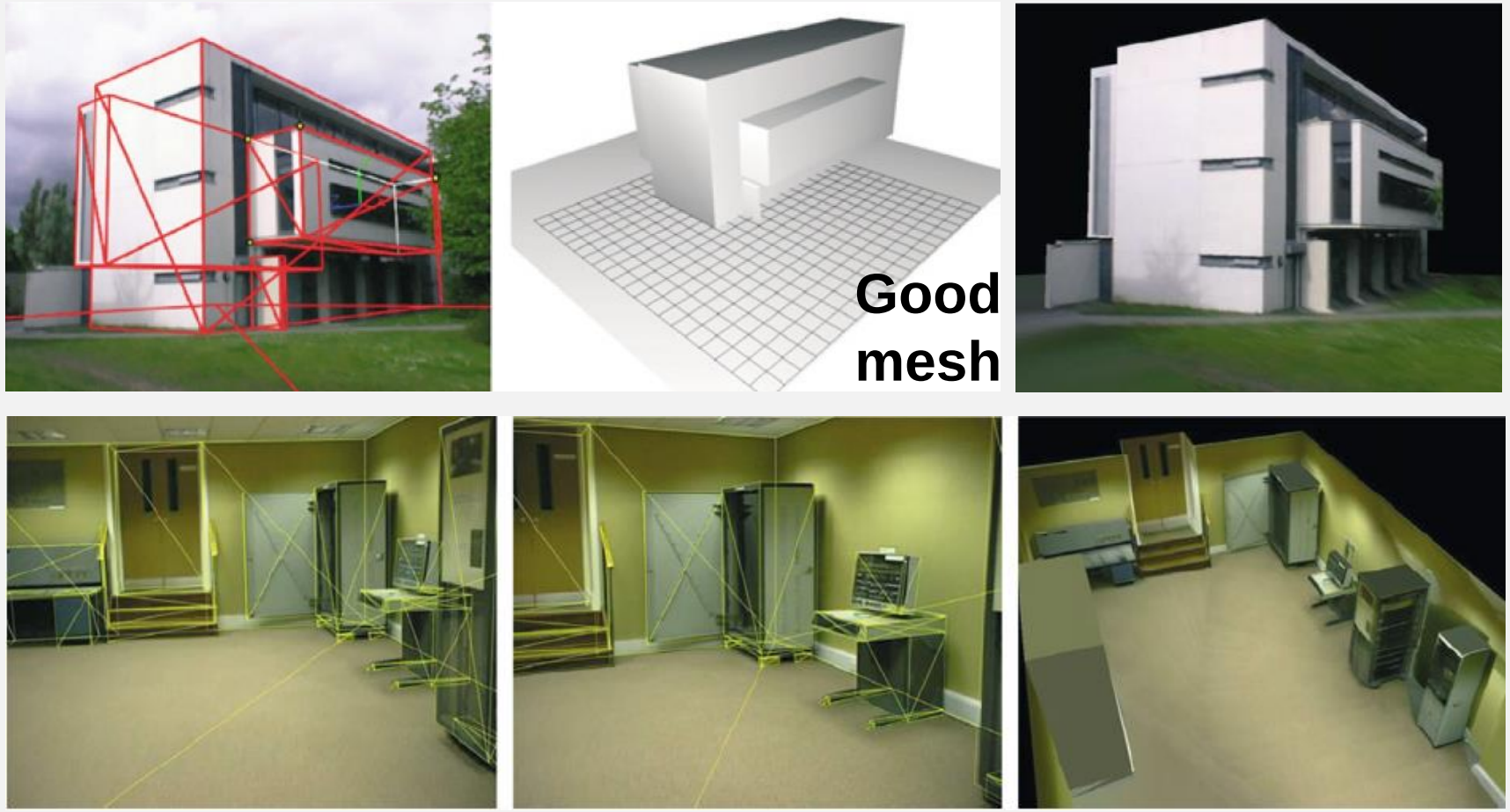
SFM Pipeline

- Step 3: Refine estimates
  - “Bundle adjustment” in photogrammetry
  - Other iterative methods



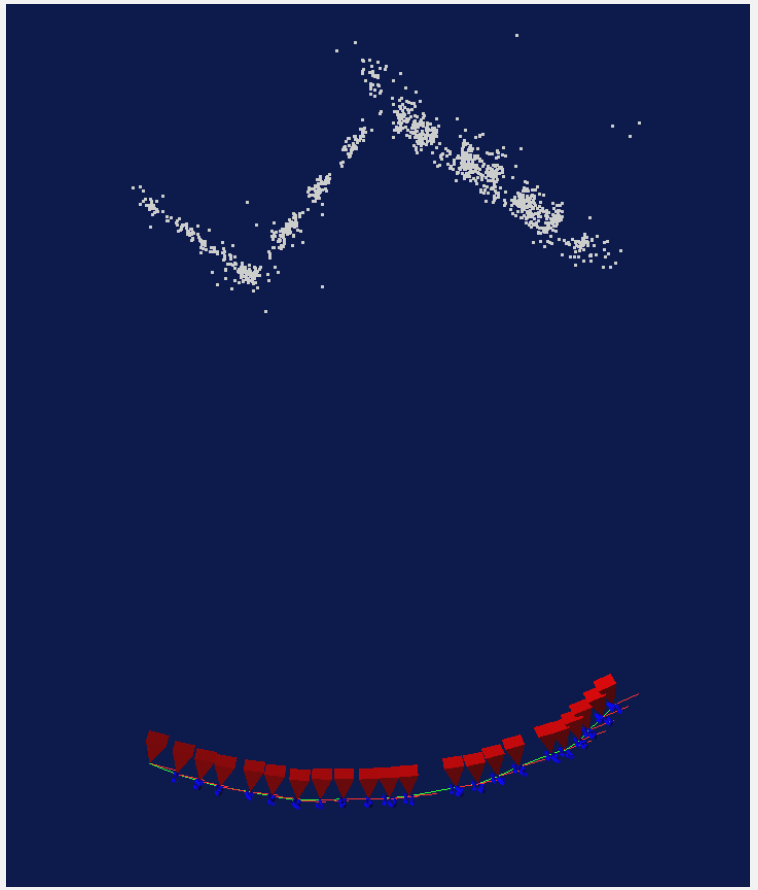
SFM Pipeline

- Step 4: Recover surfaces (image-based triangulation, silhouettes, stereo…)



Factorization Methods (for Initial Guess)

- Problem statement



Notations

- n 3D points are seen in m views
- $\mathbf{q}=(u,v,1)$: *2D image point*
- $\mathbf{p}=(x,y,z,1)$: *3D scene point*
- Π: projection matrix (camera parameters)
- π: projection function
- q_{ij} is the projection of the i-th point on image j
- λ_{ij} projective depth of q_{ij}

$$\mathbf{q}_{ij} = \pi(\Pi_j \mathbf{p}_i)$$

$$\pi(x, y, z) = (x / z, y / z)$$

$$\lambda_{ij} = z$$

Structure from Motion

- Estimate $\prod_j$ and $\mathbf{p}_i$ to minimize

$$\varepsilon(\mathbf{\Pi}_1,\cdots,\mathbf{\Pi}_m,\mathbf{p}_1,\cdots,\mathbf{p}_n)=\sum_{j=1}^{m}\sum_{i=1}^{n} w_{ij}\log P(\pi(\mathbf{\Pi}_j\mathbf{p}_i);\mathbf{q}_{ij})$$

$$w_{ij}=\begin{cases}1 & \text{if } p_i \text{ is visible in view j}\\ 0 & \text{otherwise}\end{cases}$$

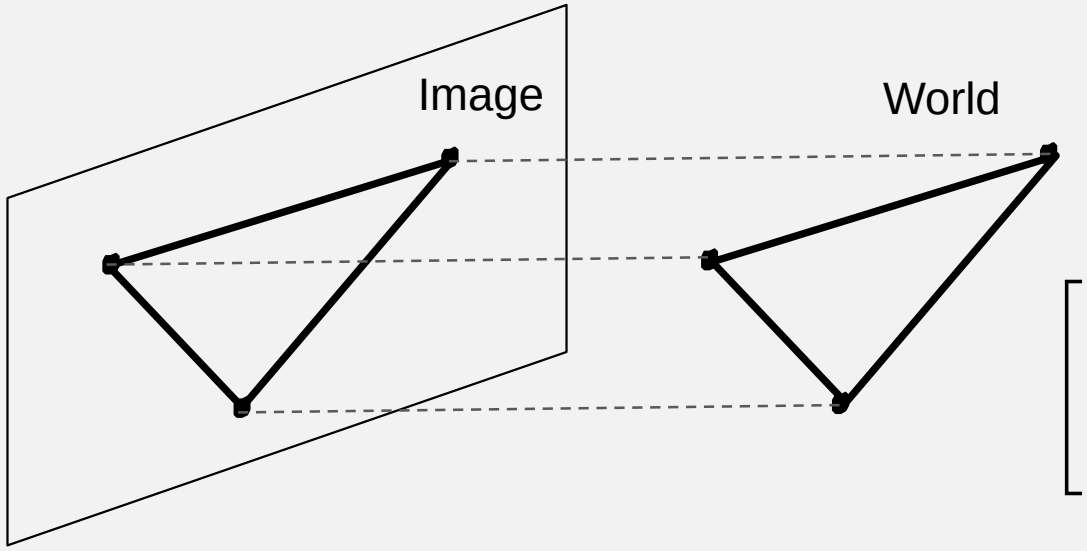
- Assume isotropic Gaussian noise, it is reduced to

$$\varepsilon(\mathbf{\Pi}_1,\cdots,\mathbf{\Pi}_m,\mathbf{p}_1,\cdots,\mathbf{p}_n)=\sum_{j=1}^{m}\sum_{i=1}^{n} w_{ij}\left\|\pi(\mathbf{\Pi}_j\mathbf{p}_i)-\mathbf{q}_{ij}\right\|^2$$

- Start from a simpler projection model

Orthographic Projection

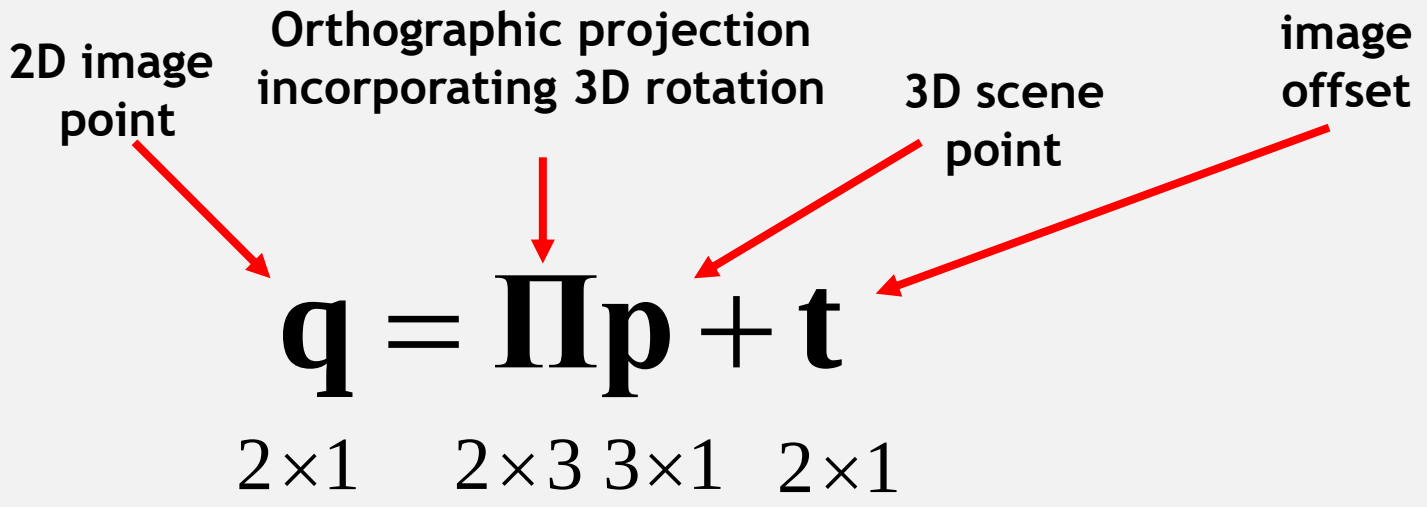
- Special case of perspective projection
  - Distance from the COP to the PP is infinite



$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \Rightarrow (x, y)$$

  - Also called “parallel projection”: (x, y, z) → (x, y)

SFM Under Orthographic Projection



- Trick
  - Choose scene origin to be centroid of 3D points
  - Choose image origins to be centroid of 2D points
  - Allows us to drop the camera translation:

$$\mathbf{q} = \Pi\mathbf{p}$$

Factorization (Tomasi & Kanade)

projection of *n* features in one image:

$$\underset{2\times n}{\begin{bmatrix}\mathbf{q}_1 & \mathbf{q}_2 & \cdots & \mathbf{q}_n\end{bmatrix}} = \underset{2\times 3}{\prod} \underset{3\times n}{\begin{bmatrix}\mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_n\end{bmatrix}}$$

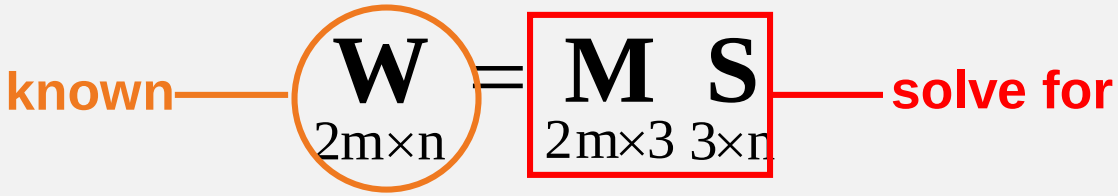
projection of *n* features in *m* images

$$\underset{2m\times n}{\begin{bmatrix}\mathbf{q}_{11} & \mathbf{q}_{12} & \cdots & \mathbf{q}_{1n} \\ \mathbf{q}_{21} & \mathbf{q}_{22} & \cdots & \mathbf{q}_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ \mathbf{q}_{m1} & \mathbf{q}_{m2} & \cdots & \mathbf{q}_{mn}\end{bmatrix}} = \underset{2m\times 3}{\begin{bmatrix}\mathbf{\Pi}_1 \\ \mathbf{\Pi}_2 \\ \vdots \\ \mathbf{\Pi}_m\end{bmatrix}} \underset{3\times n}{\begin{bmatrix}\mathbf{p}_1 & \mathbf{p}_2 & \cdots & \mathbf{p}_n\end{bmatrix}}$$

Key Observation: *rank*(**W**) <= 3

W measurement **M** motion **S** shape

Factorization



- Factorization Technique
  - W is at most rank 3 (assuming no noise)
  - We can use singular value decomposition to factor W:

$$\underset{2m\times n}{\mathbf{W}} = \underset{2m\times 3}{\mathbf{M}'}\ \underset{3\times n}{\mathbf{S}'}$$

  - S′ differs from S by a linear transformation A:

$$\mathbf{W} = \mathbf{M}'\mathbf{S}' = (\mathbf{M}\mathbf{A}^{-1})(\mathbf{A}\mathbf{S})$$

  - Solve for A by enforcing metric constraints on M

Metric Constraints

- Orthographic Camera
  - Rows of Π are orthonormal: $\Pi\Pi^T = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$
- Enforcing "Metric" Constraints
  - Compute $\mathbf{A}$ such that rows of $\mathbf{M}$ have these properties

$$\mathbf{M}'\mathbf{A} = \mathbf{M}$$

Trick (not in original Tomasi/Kanade paper, but in followup work)

- Constraints are linear in $\mathbf{A}\mathbf{A}^T$:

$$\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \Pi\Pi^T = \Pi'\mathbf{A}(\mathbf{A}\Pi')^T = \Pi'\mathbf{G}\Pi'^T \qquad \textit{where } \mathbf{G} = \mathbf{A}\mathbf{A}^T$$

- Solve for **G** first by writing equations for every Π_i in **M**
- Then $\mathbf{G} = \mathbf{A}\mathbf{A}^T$ by SVD (since $\mathbf{U} = \mathbf{V}$)

Factorization with Noisy Data

$$\underset{2m\times n}{\mathbf{W}} = \underset{2m\times 3}{\mathbf{M}}\ \underset{3\times n}{\mathbf{S}} + \underset{2m\times n}{\mathbf{E}}$$

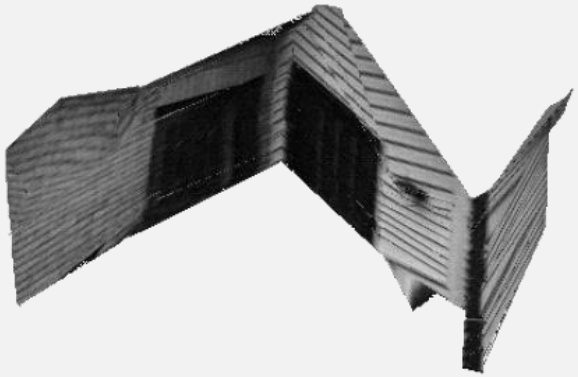
- SVD gives this solution
  - Provides optimal rank 3 approximation W′ of W

$$\underset{2m\times n}{\mathbf{W}} = \underset{2m\times n}{\mathbf{W}'} + \underset{2m\times n}{\mathbf{E}}$$

- Approach
  - Estimate W′, then use noise-free factorization of W′ as before
  - Result minimizes the SSD between positions of image features and projection of the reconstruction

Results




Extensions to Factorization Methods

- Projective projection
- With missing data
- Projective projection with missing data

Bundle adjustment

Levenberg-Marquardt Method

- LM can be thought of as a combination of steepest descent and the Newton method.
- When the current solution is far from the correct one, the algorithm behaves like a steepest descent method: slow, but guaranteed to converge.
- When the current solution is close to the correct solution, it becomes a Newton's method.

Nonlinear Least Square

Given a set of measurements $\mathbf{x}$, try to find the best parameter vector $\mathbf{p}$ so that the squared distance $\varepsilon^T \varepsilon$ is minimal. Here, $\varepsilon = \mathbf{x} - \hat{\mathbf{x}}$, with $\hat{\mathbf{x}} = f(\mathbf{p})$.

Levenberg-Marquardt Method

For a small $||\delta_{\mathbf{p}}||$, $f(\mathbf{p}+\delta_{\mathbf{p}}) \approx f(\mathbf{p}) + \mathbf{J}\delta_{\mathbf{p}}$

$\mathbf{J}$ is the Jacobian matrix $\frac{\partial f(\mathbf{p})}{\partial \mathbf{p}}$

it is required to find the $\delta_{\mathbf{p}}$ that minimizes the quantity

$$||\mathbf{x} - f(\mathbf{p}+\delta_{\mathbf{p}})|| \approx ||\mathbf{x} - f(\mathbf{p}) - \mathbf{J}\delta_{\mathbf{p}}|| = ||\epsilon - \mathbf{J}\delta_{\mathbf{p}}||$$

$$\mathbf{J}^T\mathbf{J}\delta_{\mathbf{p}} = \mathbf{J}^T\epsilon$$

$$\mathbf{N}\delta_{\mathbf{p}} = \mathbf{J}^T\epsilon$$

$$\mathbf{N}_{ii} = \mu + \left[\mathbf{J}^T\mathbf{J}\right]_{ii}$$

damping term

Levenberg-Marquardt Method

- $\mu=0 \rightarrow$ Newton's method
- $\mu \rightarrow \infty \rightarrow$ steepest descent method
- Strategy for choosing μ
  - Start with some small μ
  - If error is not reduced, keep trying larger μ until it does
  - If error is reduced, accept it and reduce μ for the next iteration

Bundle Adjustment

- Bundle adjustment (BA) is a technique for simultaneously refining the 3D structure and camera parameters
- It is capable of obtaining an optimal reconstruction under certain assumptions on image error models. For zero-mean Gaussian image errors, BA is the maximum likelihood estimator.

Bundle Adjustment

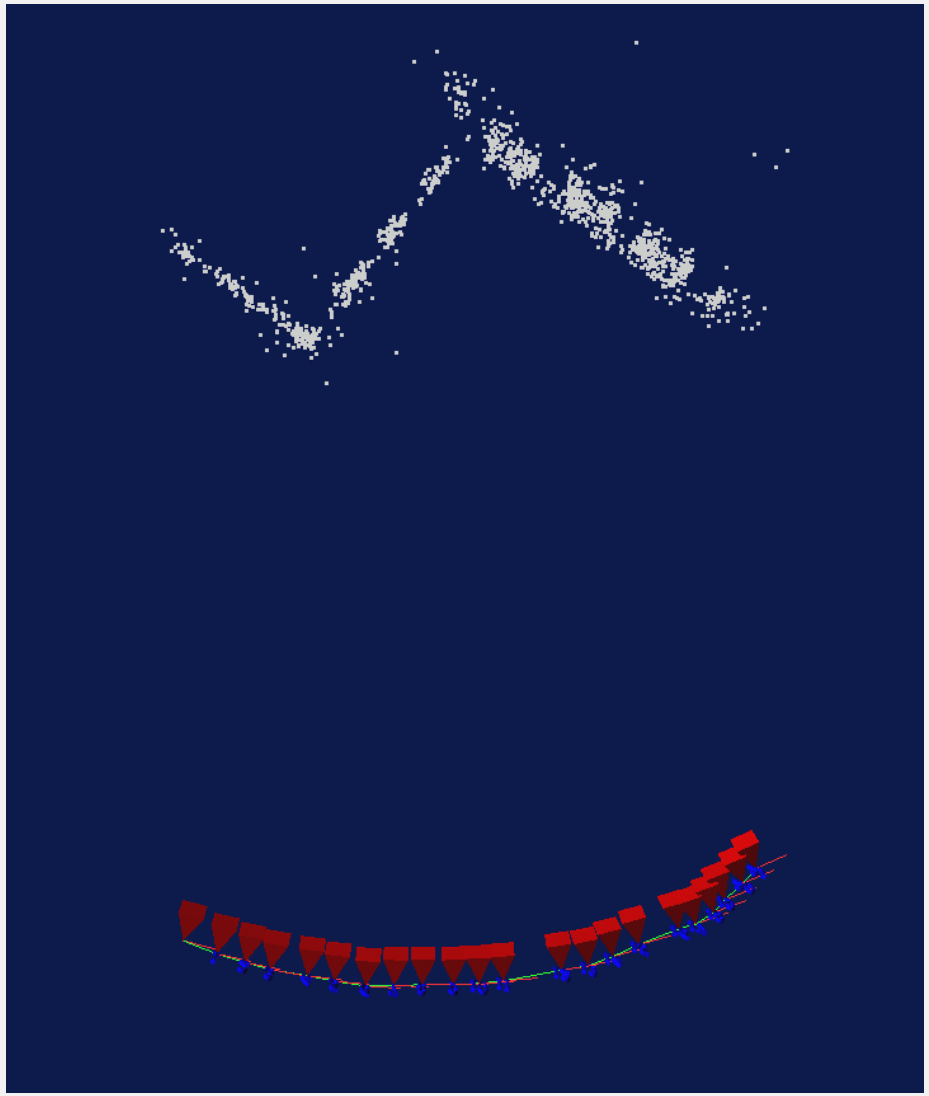
- n 3D points are seen in m views
- x_{ij} is the projection of the i-th point on image j
- a_j is the parameters for the j-th camera
- b_i is the parameters for the i-th point
- BA attempts to minimize the projection error

$$\min_{\mathbf{a}_j, \mathbf{b}_i} \sum_{i=1}^{n} \sum_{j=1}^{m} d(\mathbf{Q}(\mathbf{a}_j, \mathbf{b}_i),\ \mathbf{x}_{ij})^2$$

predicted projection

Euclidean distance

Bundle Adjustment



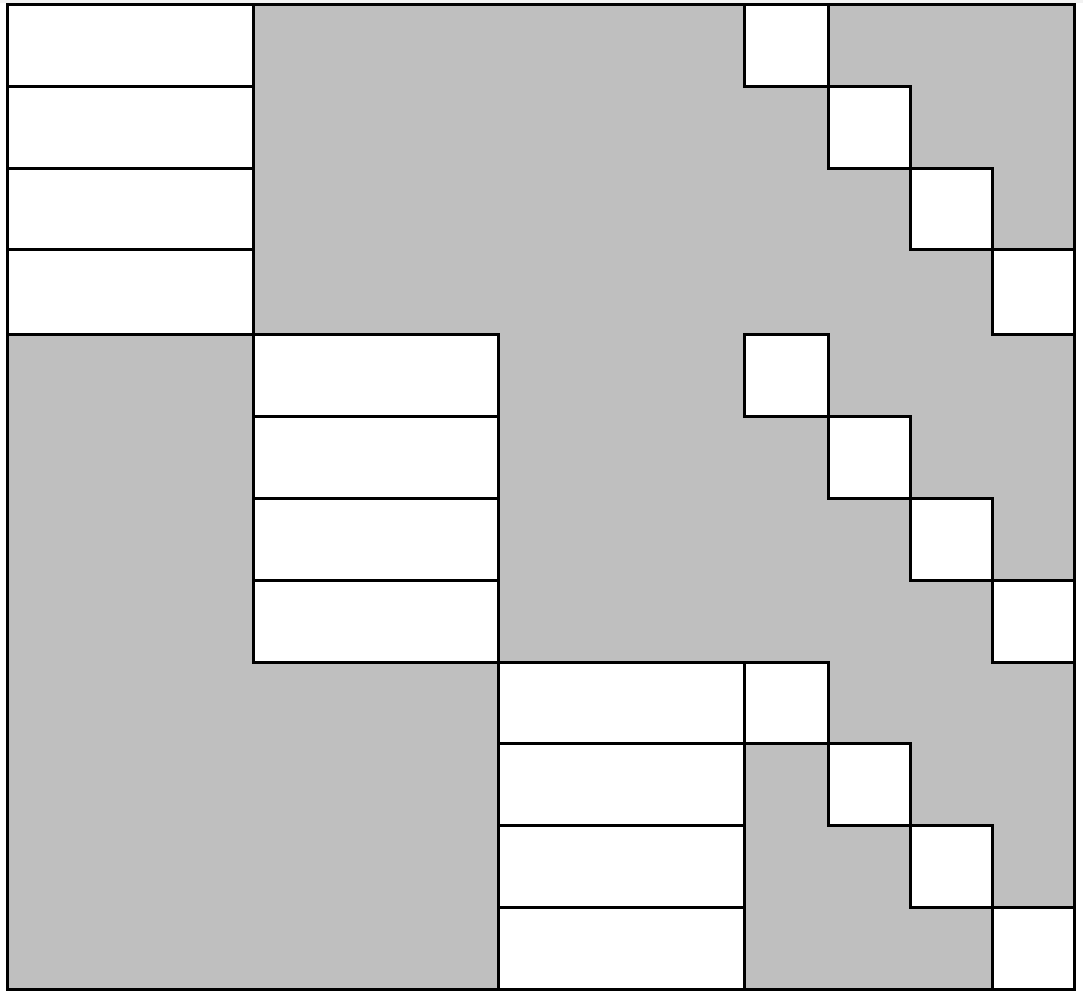
Bundle Adjustment

3 views and 4 points $\mathbf{P} = (\mathbf{a}_1^T, \mathbf{a}_2^T, \mathbf{a}_3^T, \mathbf{b}_1^T, \mathbf{b}_2^T, \mathbf{b}_3^T, \mathbf{b}_4^T)^T$

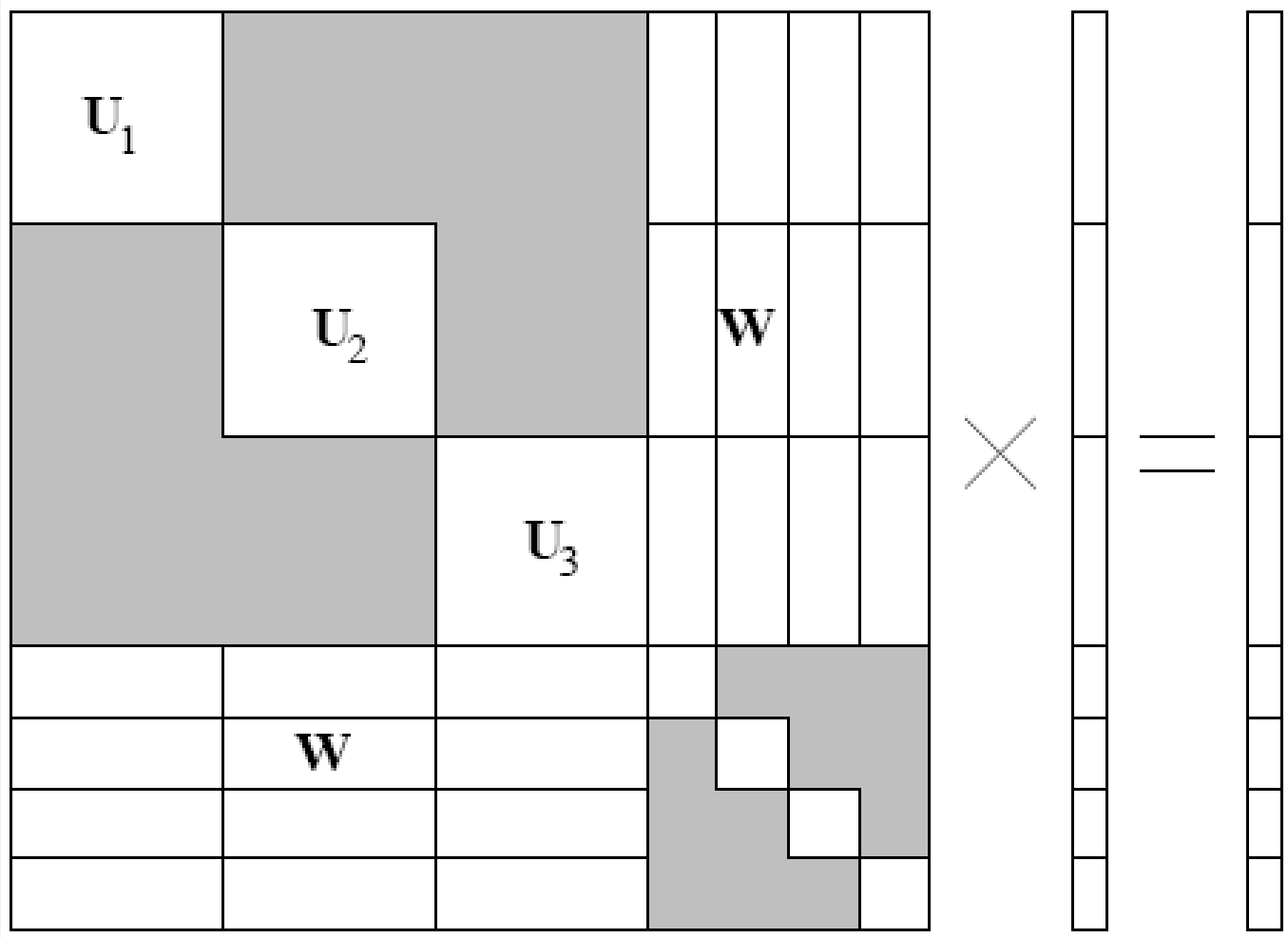
$$\mathbf{X} = (\mathbf{x}_{11}^T, \mathbf{x}_{12}^T, \mathbf{x}_{13}^T, \mathbf{x}_{21}^T, \mathbf{x}_{22}^T, \mathbf{x}_{23}^T, \mathbf{x}_{31}^T, \mathbf{x}_{32}^T, \mathbf{x}_{33}^T, \mathbf{x}_{41}^T, \mathbf{x}_{42}^T, \mathbf{x}_{43}^T)^T$$

$$\frac{\partial \mathbf{X}}{\partial \mathbf{P}} = \begin{pmatrix}
\mathbf{A}_{11} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{11} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{A}_{12} & \mathbf{0} & \mathbf{B}_{12} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{A}_{13} & \mathbf{B}_{13} & \mathbf{0} & \mathbf{0} & \mathbf{0} \\
\mathbf{A}_{21} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{21} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{A}_{22} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{22} & \mathbf{0} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{A}_{23} & \mathbf{0} & \mathbf{B}_{23} & \mathbf{0} & \mathbf{0} \\
\mathbf{A}_{31} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{31} & \mathbf{0} \\
\mathbf{0} & \mathbf{A}_{32} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{32} & \mathbf{0} \\
\mathbf{0} & \mathbf{0} & \mathbf{A}_{33} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{33} & \mathbf{0} \\
\mathbf{A}_{41} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{41} \\
\mathbf{0} & \mathbf{A}_{42} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{42} \\
\mathbf{0} & \mathbf{0} & \mathbf{A}_{43} & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{B}_{43}
\end{pmatrix}$$

Typical Jacobian



Block Structure of Normal Equation



Bundle Adjustment

$$\begin{pmatrix} \mathbf{U}_1 & \mathbf{0} & \mathbf{0} & \mathbf{W}_{11} & \mathbf{W}_{21} & \mathbf{W}_{31} & \mathbf{W}_{41} \\ \mathbf{0} & \mathbf{U}_2 & \mathbf{0} & \mathbf{W}_{12} & \mathbf{W}_{22} & \mathbf{W}_{32} & \mathbf{W}_{42} \\ \mathbf{0} & \mathbf{0} & \mathbf{U}_3 & \mathbf{W}_{13} & \mathbf{W}_{23} & \mathbf{W}_{33} & \mathbf{W}_{43} \\ \mathbf{W}_{11}{}^T & \mathbf{W}_{12}{}^T & \mathbf{W}_{13}{}^T & \mathbf{V}_1 & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{W}_{21}{}^T & \mathbf{W}_{22}{}^T & \mathbf{W}_{23}{}^T & \mathbf{0} & \mathbf{V}_2 & \mathbf{0} & \mathbf{0} \\ \mathbf{W}_{31}{}^T & \mathbf{W}_{32}{}^T & \mathbf{W}_{33}{}^T & \mathbf{0} & \mathbf{0} & \mathbf{V}_3 & \mathbf{0} \\ \mathbf{W}_{41}{}^T & \mathbf{W}_{42}{}^T & \mathbf{W}_{43}{}^T & \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{V}_4 \end{pmatrix} \begin{pmatrix} \delta_{\mathbf{a}_1} \\ \delta_{\mathbf{a}_2} \\ \delta_{\mathbf{a}_3} \\ \delta_{\mathbf{b}_1} \\ \delta_{\mathbf{b}_2} \\ \delta_{\mathbf{b}_3} \\ \delta_{\mathbf{b}_4} \end{pmatrix} = \begin{pmatrix} \epsilon_{\mathbf{a}_1} \\ \epsilon_{\mathbf{a}_2} \\ \epsilon_{\mathbf{a}_3} \\ \epsilon_{\mathbf{b}_1} \\ \epsilon_{\mathbf{b}_2} \\ \epsilon_{\mathbf{b}_3} \\ \epsilon_{\mathbf{b}_4} \end{pmatrix}$$

$$\mathbf{U}^* = \begin{pmatrix} \mathbf{U}_1^* & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{U}_2^* & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{U}_3^* \end{pmatrix}, \mathbf{V}^* = \begin{pmatrix} \mathbf{V}_1^* & \mathbf{0} & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{V}_2^* & \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{V}_3^* & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \mathbf{0} & \mathbf{V}_4^* \end{pmatrix}, \mathbf{W} = \begin{pmatrix} \mathbf{W}_{11} & \mathbf{W}_{21} & \mathbf{W}_{31} & \mathbf{W}_{41} \\ \mathbf{W}_{12} & \mathbf{W}_{22} & \mathbf{W}_{32} & \mathbf{W}_{42} \\ \mathbf{W}_{13} & \mathbf{W}_{23} & \mathbf{W}_{33} & \mathbf{W}_{43} \end{pmatrix}$$

$$\begin{pmatrix} \mathbf{U}^* & \mathbf{W} \\ \mathbf{W}^T & \mathbf{V}^* \end{pmatrix} \begin{pmatrix} \delta_{\mathbf{a}} \\ \delta_{\mathbf{b}} \end{pmatrix} = \begin{pmatrix} \epsilon_{\mathbf{a}} \\ \epsilon_{\mathbf{b}} \end{pmatrix}$$

Bundle Adjustment

Multiplied by $\begin{pmatrix} \mathbf{I} & -\mathbf{W}\,\mathbf{V}^{*-1} \\ \mathbf{0} & \mathbf{I} \end{pmatrix}$

$$\begin{pmatrix} \mathbf{U}^* - \mathbf{W}\,\mathbf{V}^{*-1}\,\mathbf{W}^T & \mathbf{0} \\ \mathbf{W}^T & \mathbf{V}^* \end{pmatrix} \begin{pmatrix} \delta_\mathbf{a} \\ \delta_\mathbf{b} \end{pmatrix} = \begin{pmatrix} \epsilon_\mathbf{a} - \mathbf{W}\,\mathbf{V}^{*-1}\,\epsilon_\mathbf{b} \\ \epsilon_\mathbf{b} \end{pmatrix}$$

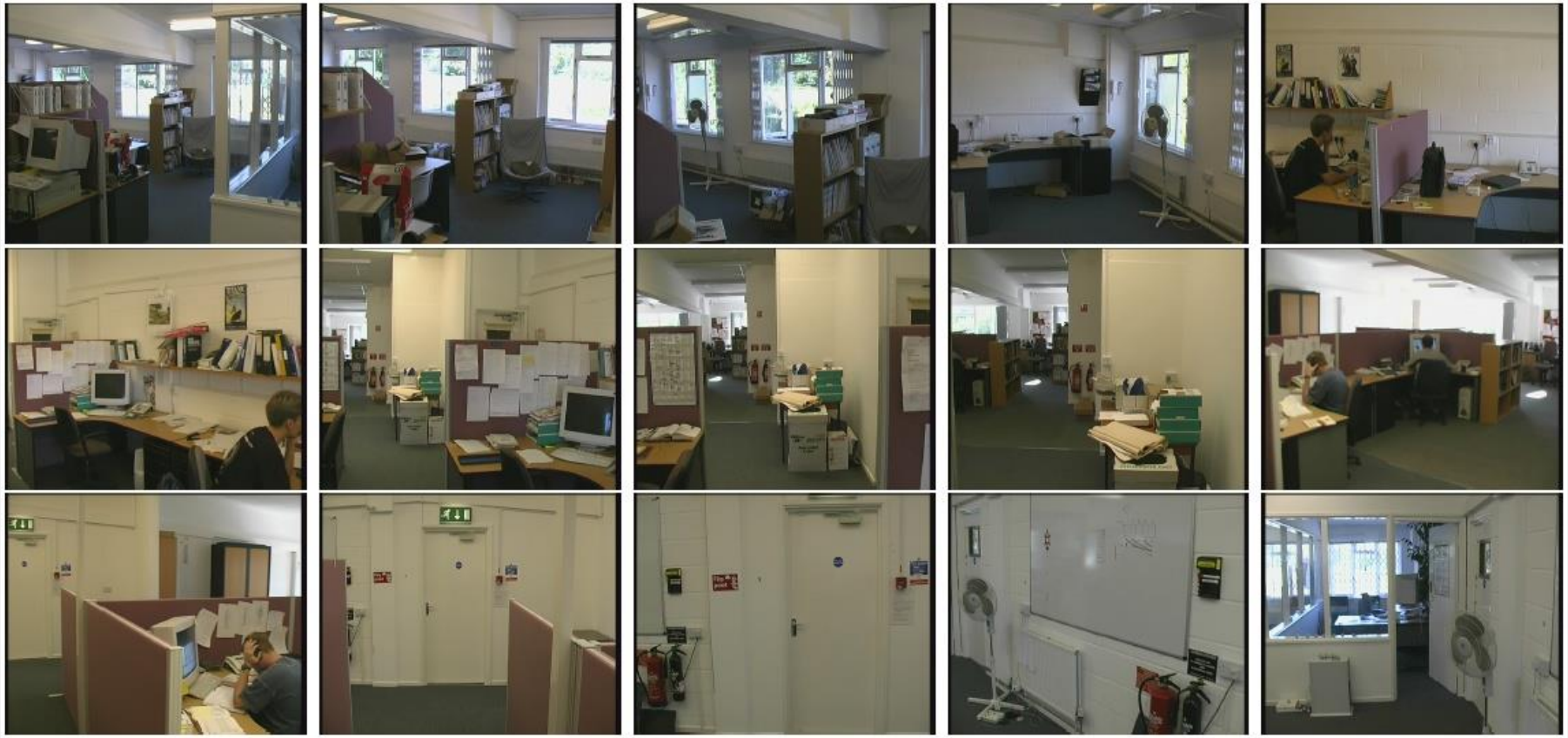
$$(\mathbf{U}^* - \mathbf{W}\,\mathbf{V}^{*-1}\,\mathbf{W}^T)\,\delta_\mathbf{a} = \epsilon_\mathbf{a} - \mathbf{W}\,\mathbf{V}^{*-1}\,\epsilon_\mathbf{b}$$

$$\mathbf{V}^*\,\delta_\mathbf{b} = \epsilon_\mathbf{b} - \mathbf{W}^T\,\delta_\mathbf{a}$$

Issues in SFM

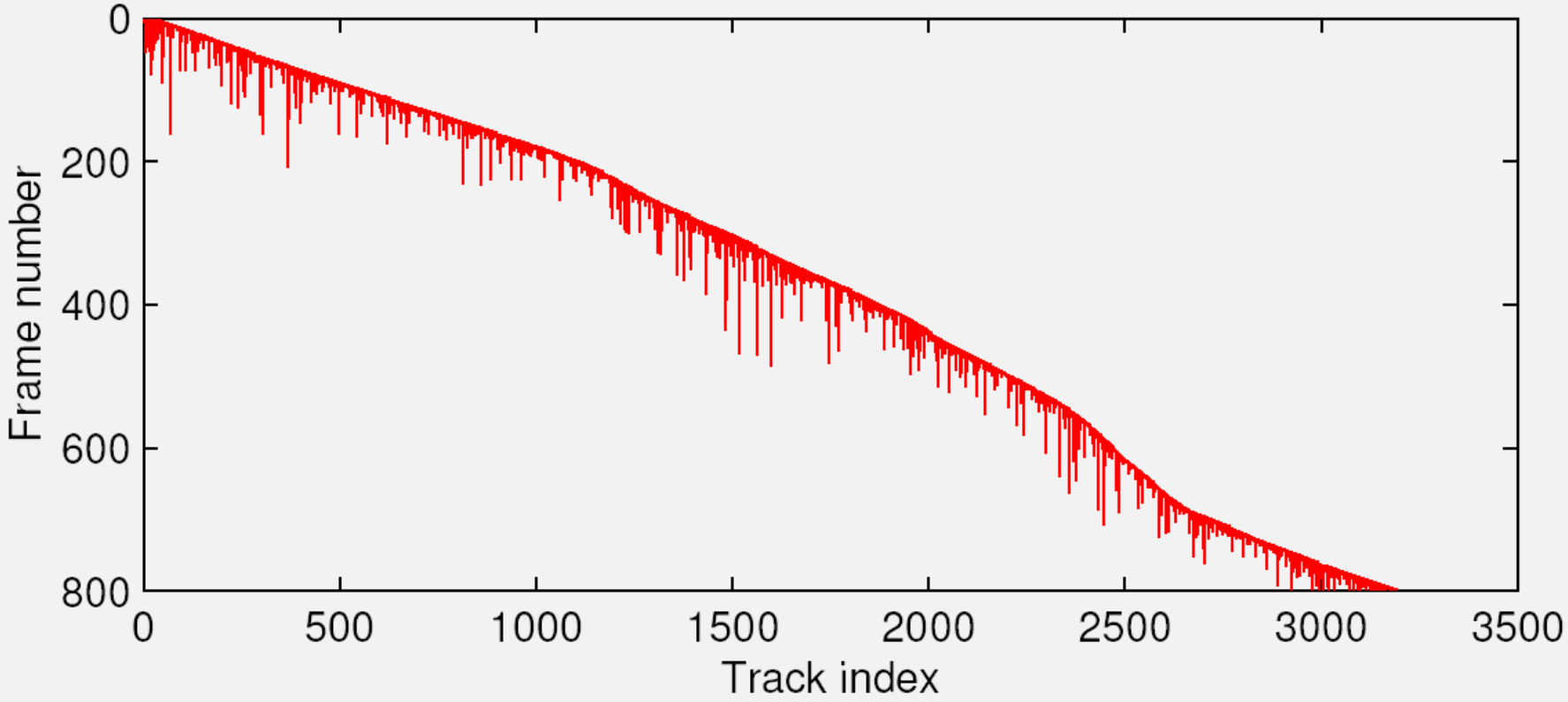
- Track lifetime
- Nonlinear lens distortion
- Degeneracy and critical surfaces
- Prior knowledge and scene constraints
- Multiple motions

Track Lifetime



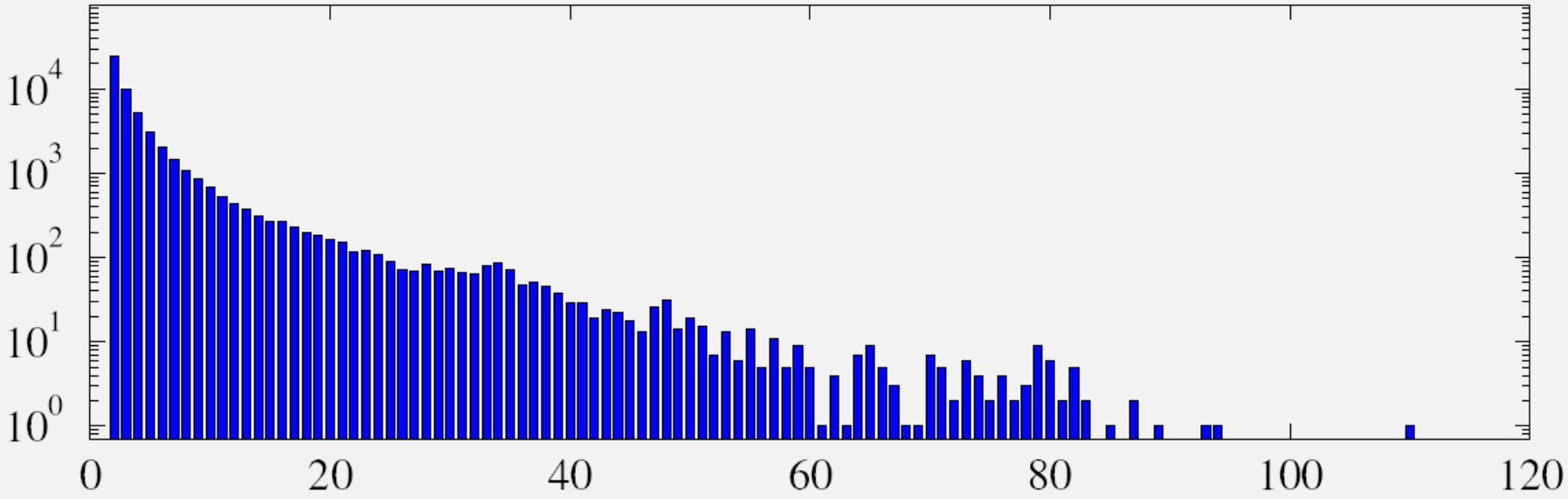
every 50th frame of a 800-frame sequence

Track Lifetime



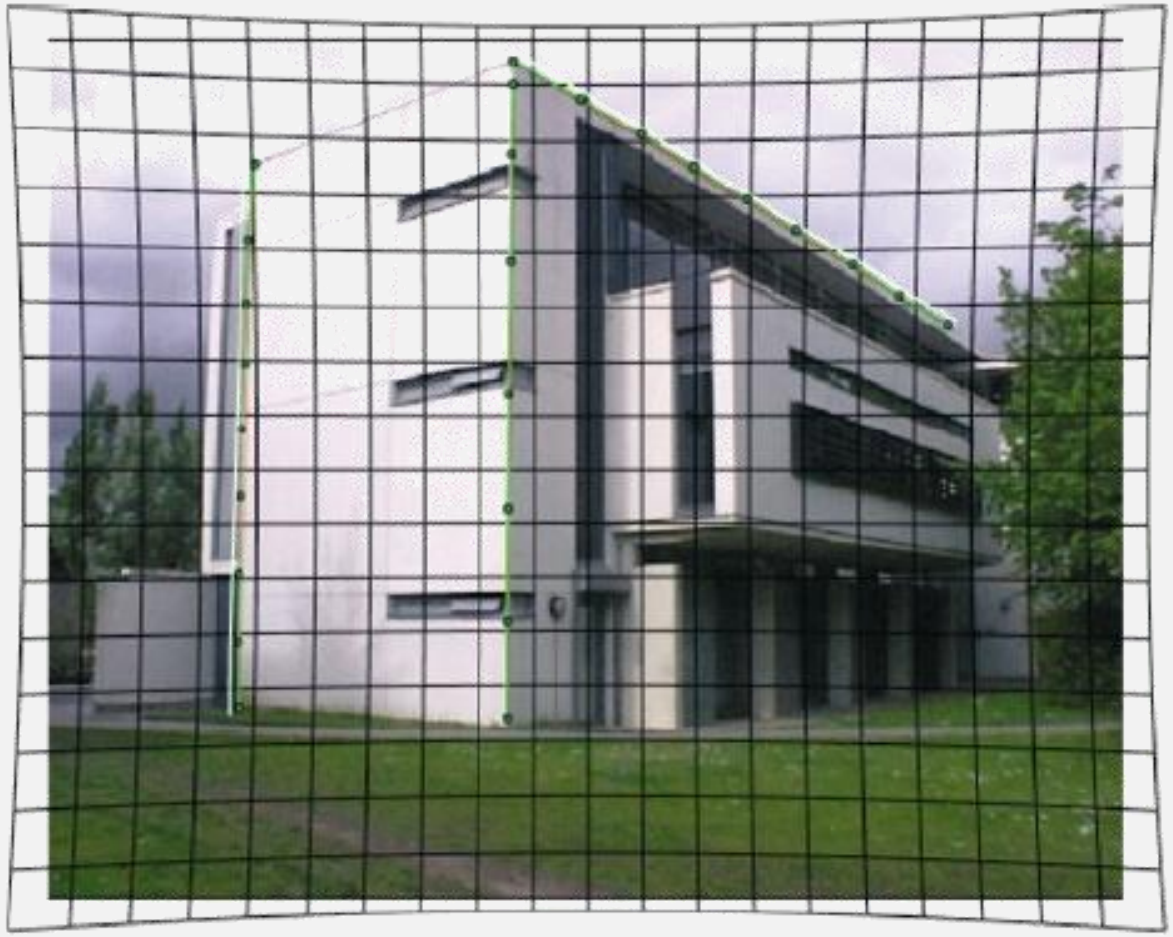
lifetime of 3192 tracks from the previous sequence

Track Lifetime

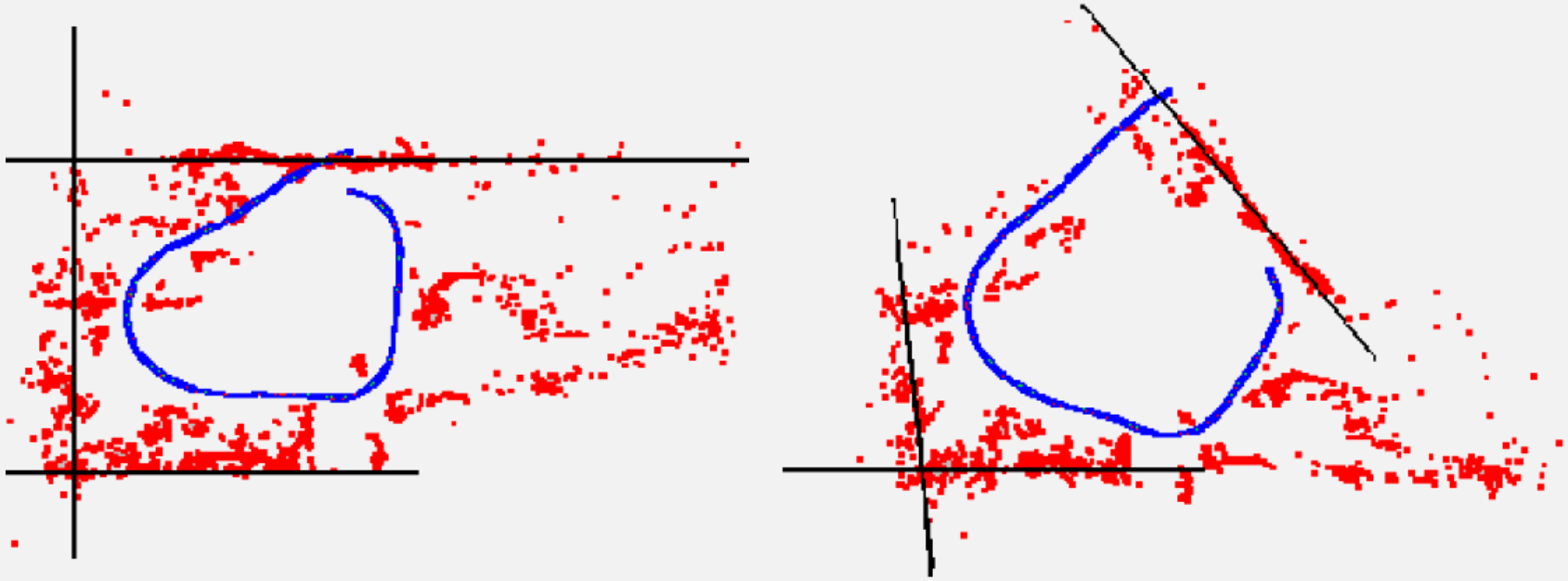


track length histogram

Nonlinear Lens Distortion



Nonlinear Lens Distortion

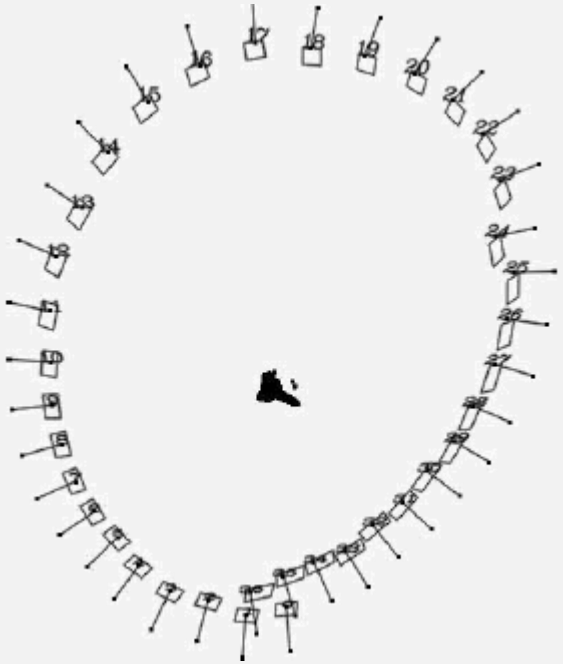


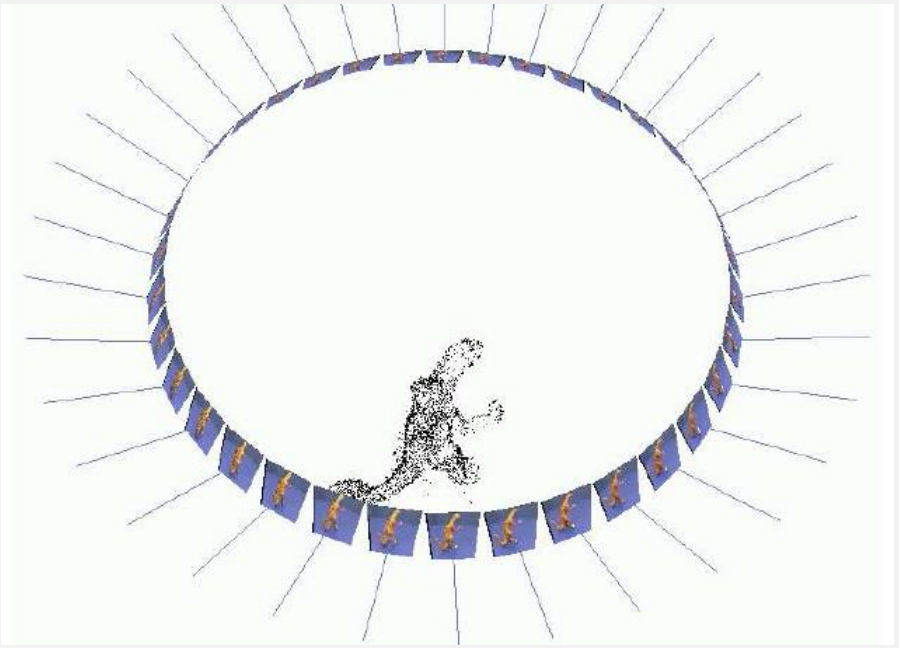
effect of lens distortion

Prior Knowledge and Scene Constraints

add a constraint that several lines are parallel

Prior Knowledge and Scene Constraints





add a constraint that it is a turntable sequence