

Camera calibration

Digital Visual Effects, Spring 2008

Yung-Yu Chuang

2008/4/15

with slides by Richard Szeliski, Steve Seitz, and Marc Pollefeys

Announcements

- Project #2 is due midnight next Monday
- Results for project #1 artifacts voting

Project #1 artifact voting

- Total 141 votes
- 33 of 37 artifacts got votes

Honorable mention(8): 胡傳甞 高紹航

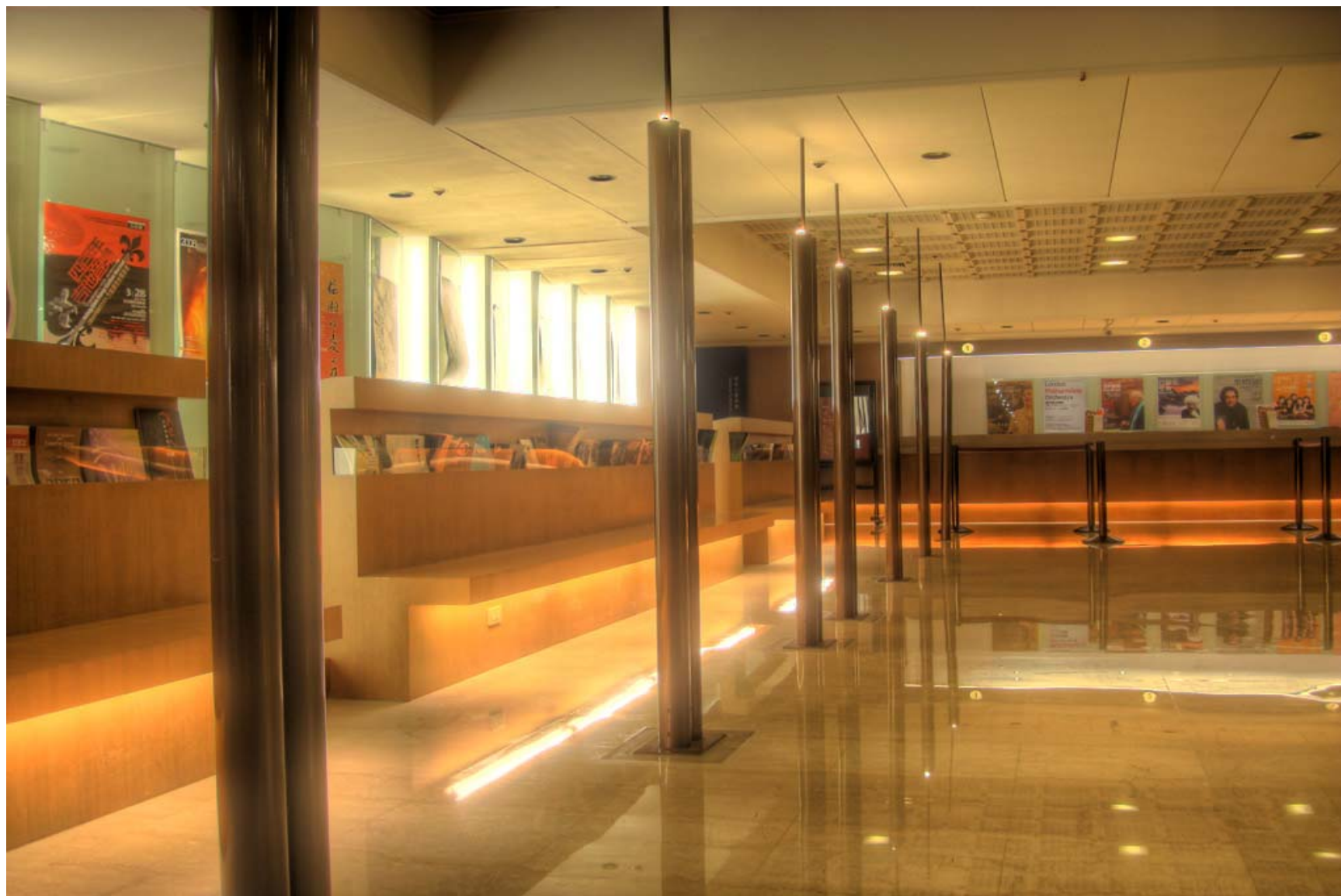
DigiVFX



Honorable mention(8): 劉俊良



Honorable mention(10): 周伯相



Third place (13): 羅聖傑 鄭京恆



Second place (14): 葉蓉蓉 劉冠廷



First place (17): 梁彧 吳孟松



Outline

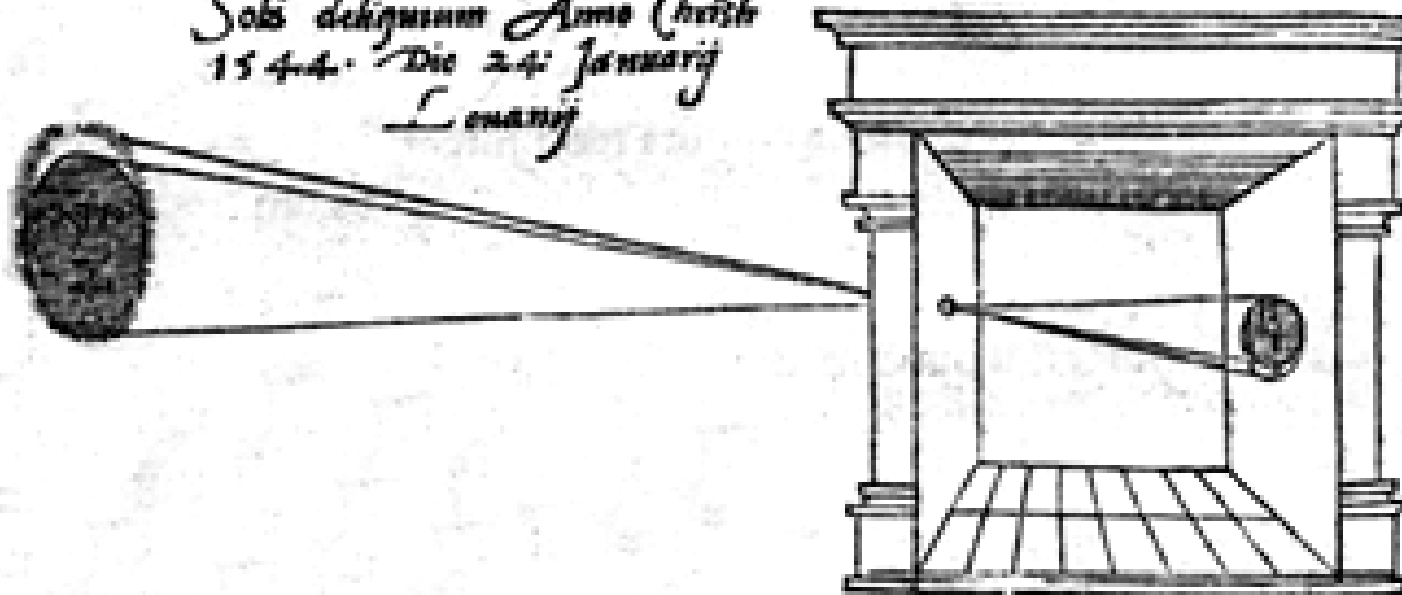
- Camera projection models
- Camera calibration
- Nonlinear least square methods

Camera projection models

Pinhole camera

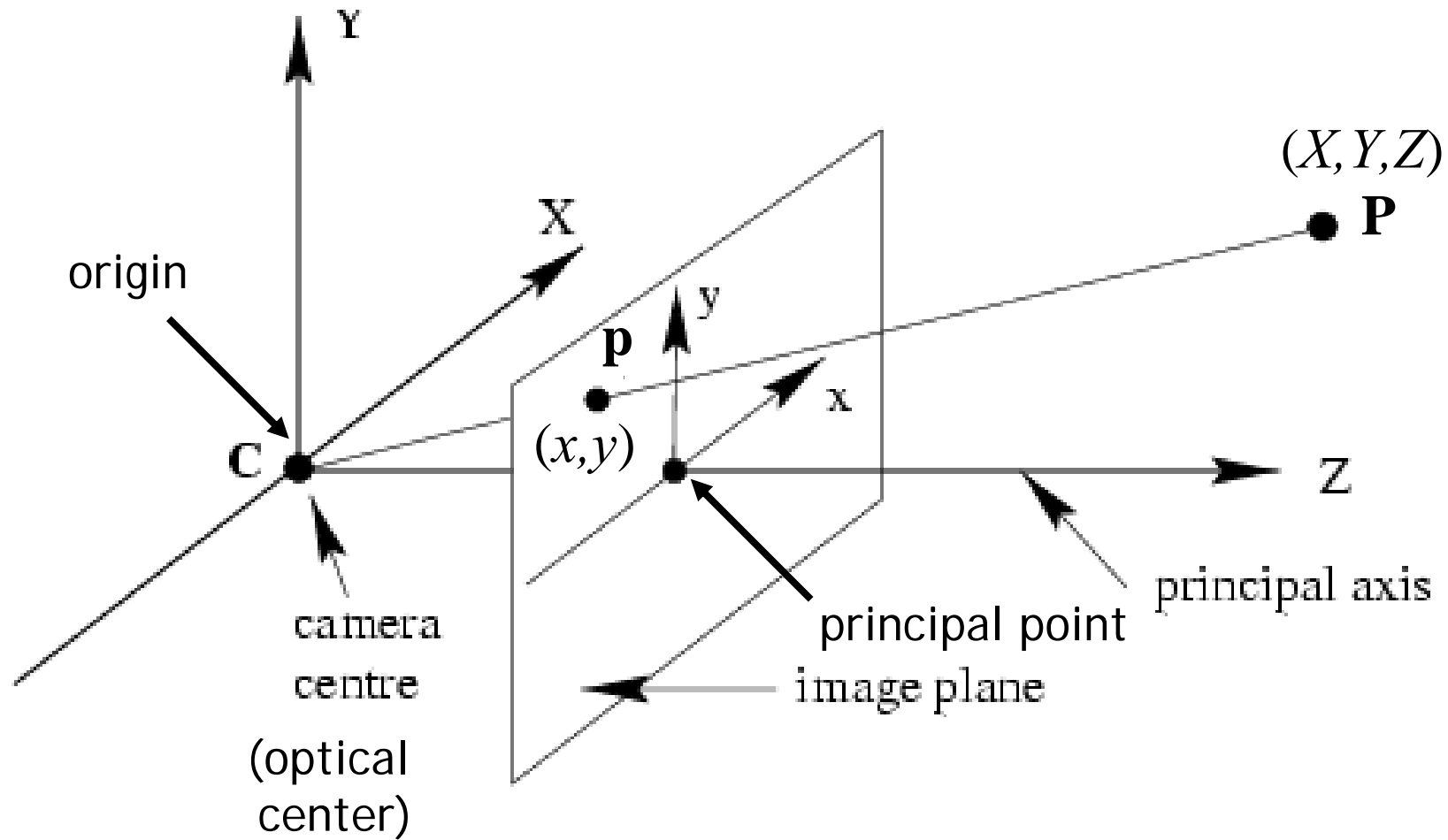
illum in tabula per radios Solis, quàm in cælo contin-
git: hoc est, si in cælo superior pars deliquiū patiatur, in
radiis apparebit inferior deficere, vt ratio exigit optica.

*Solis deliquium Anno (Christi)
1544. Die 24. Januarij
Louanij*

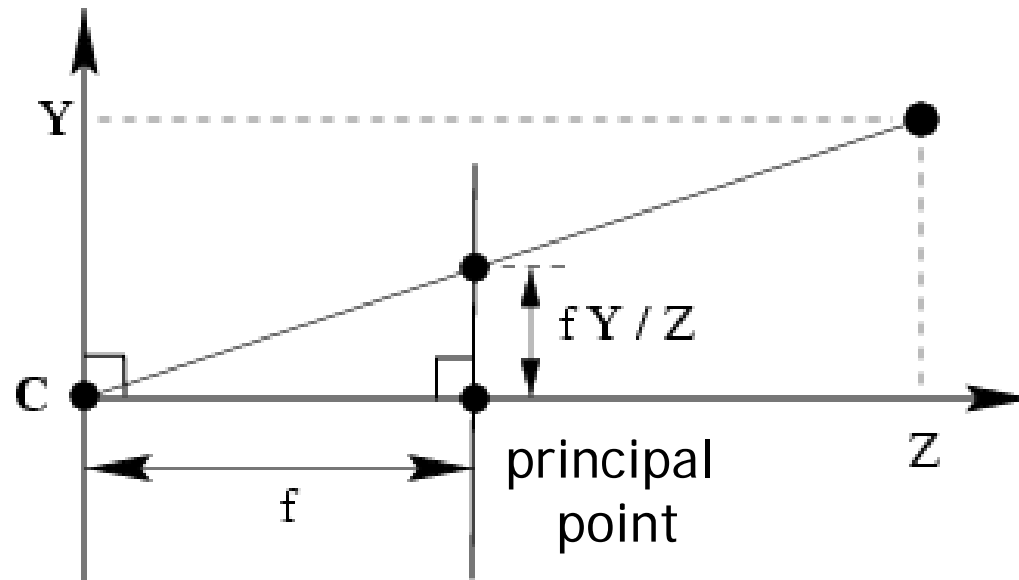


Sic nos exactè Anno .1544. Louanii eclipsim Solis
obseruauimus, inuenimusq; deficere paulò plus q̄ dex-

Pinhole camera model



Pinhole camera model

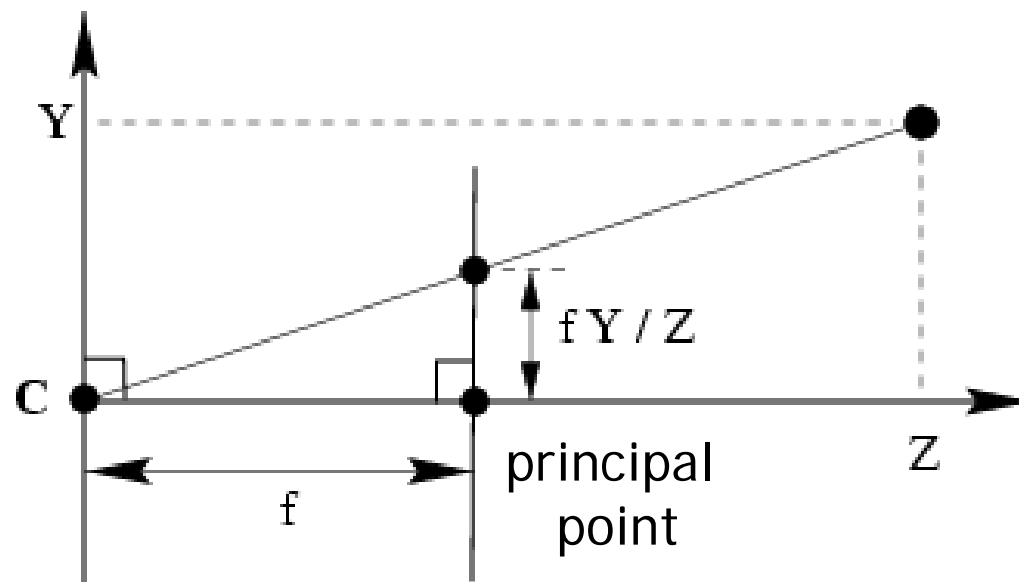


$$x = \frac{fX}{Z}$$

$$y = \frac{fY}{Z}$$

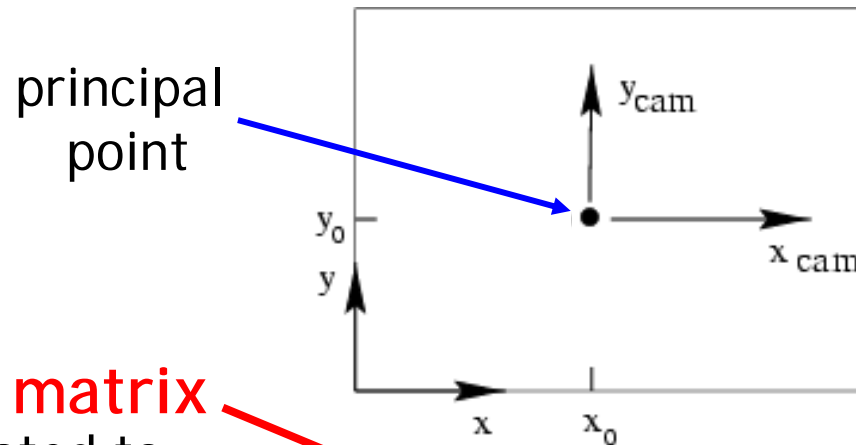
$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

Pinhole camera model



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

Principal point offset



intrinsic matrix
only related to
camera projection

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

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

Intrinsic matrix

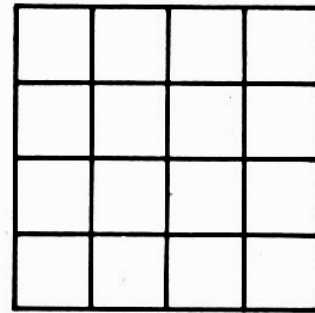
Is this form of \mathbf{K} good enough?

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

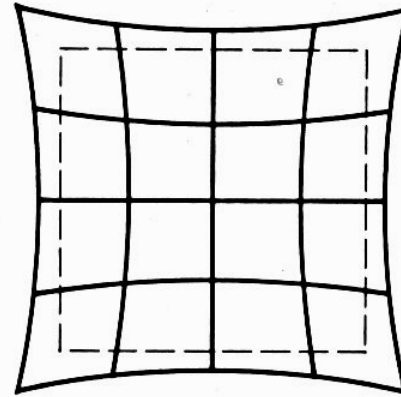
- non-square pixels (digital video)
- skew
- radial distortion

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

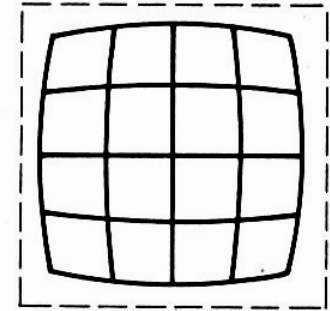
Distortion



No distortion



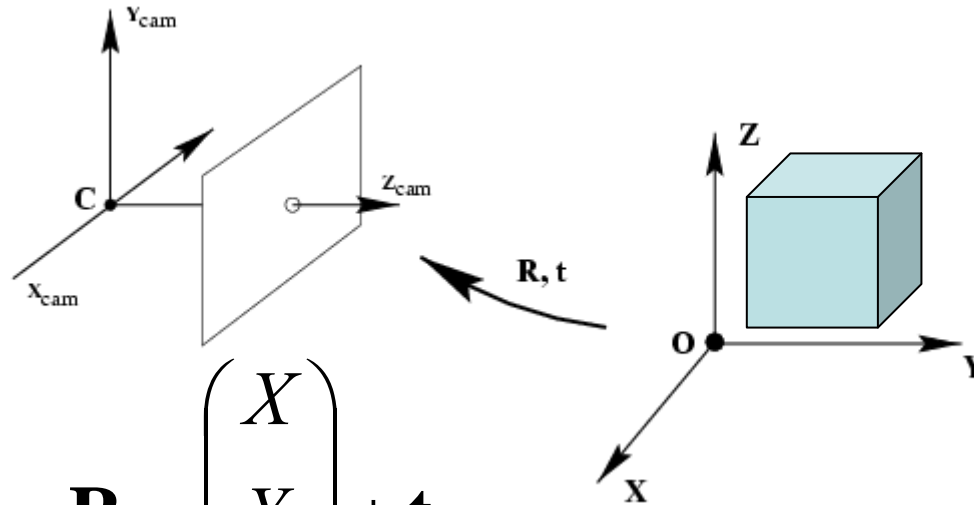
Pin cushion



Barrel

- Radial distortion of the image
 - Caused by imperfect lenses
 - Deviations are most noticeable for rays that pass through the edge of the lens

Camera rotation and translation



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

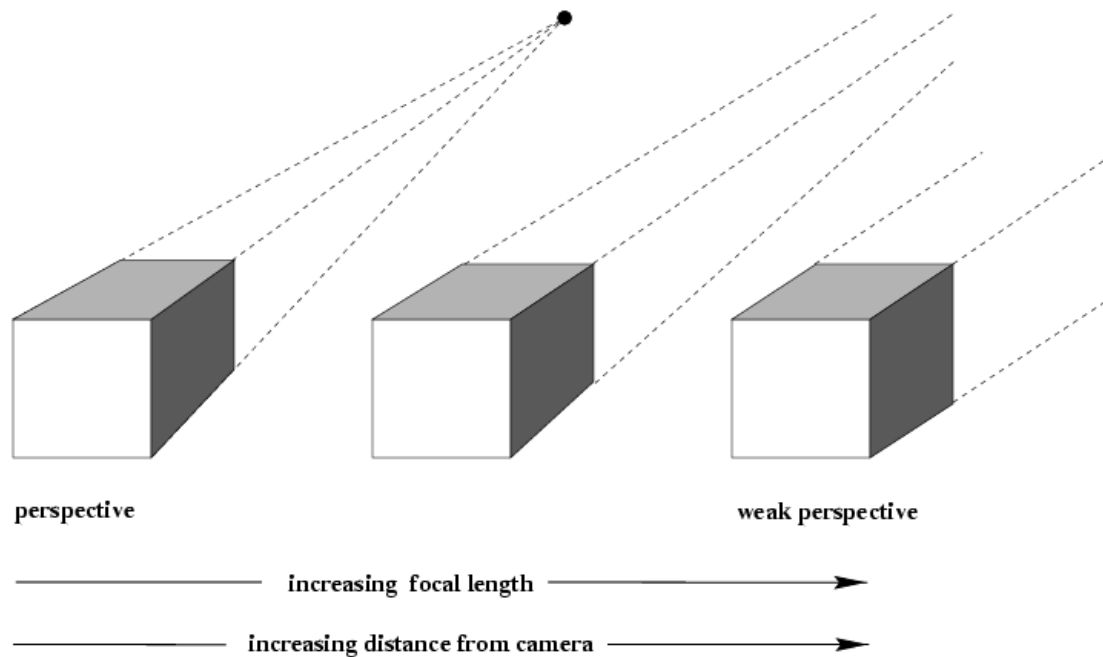
$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{bmatrix} f & 0 & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix} [\mathbf{R} | \mathbf{t}] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

$$\mathbf{x} \sim \mathbf{K} \underbrace{[\mathbf{R} | \mathbf{t}]}_{\text{extrinsic matrix}} \mathbf{X}$$

Two kinds of parameters

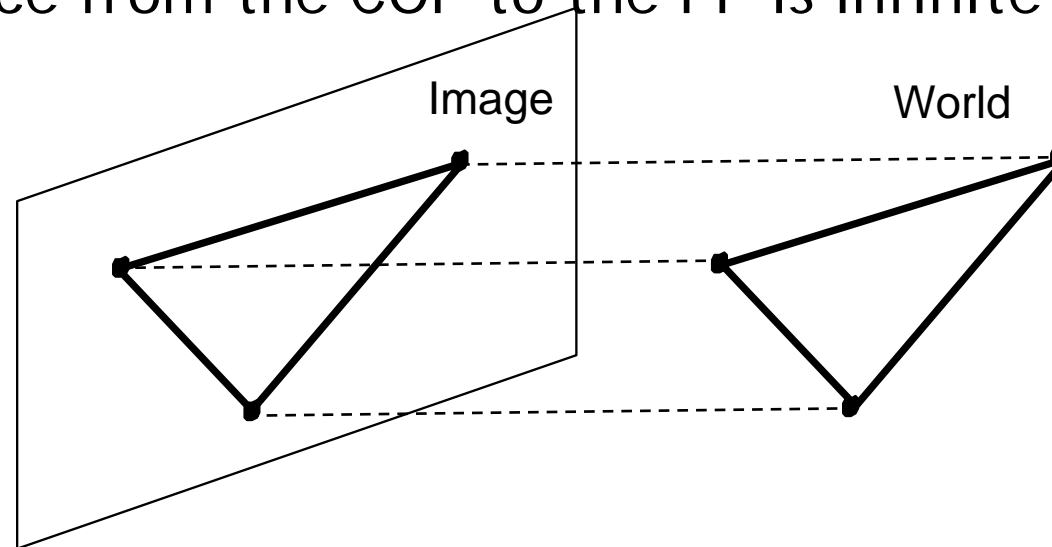
- *internal* or *intrinsic* parameters such as focal length, optical center, aspect ratio:
what kind of camera?
- *external* or *extrinsic* (pose) parameters including rotation and translation:
where is the camera?

Other projection models



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”: $(x, y, z) \rightarrow (x, y)$

Other types of projections

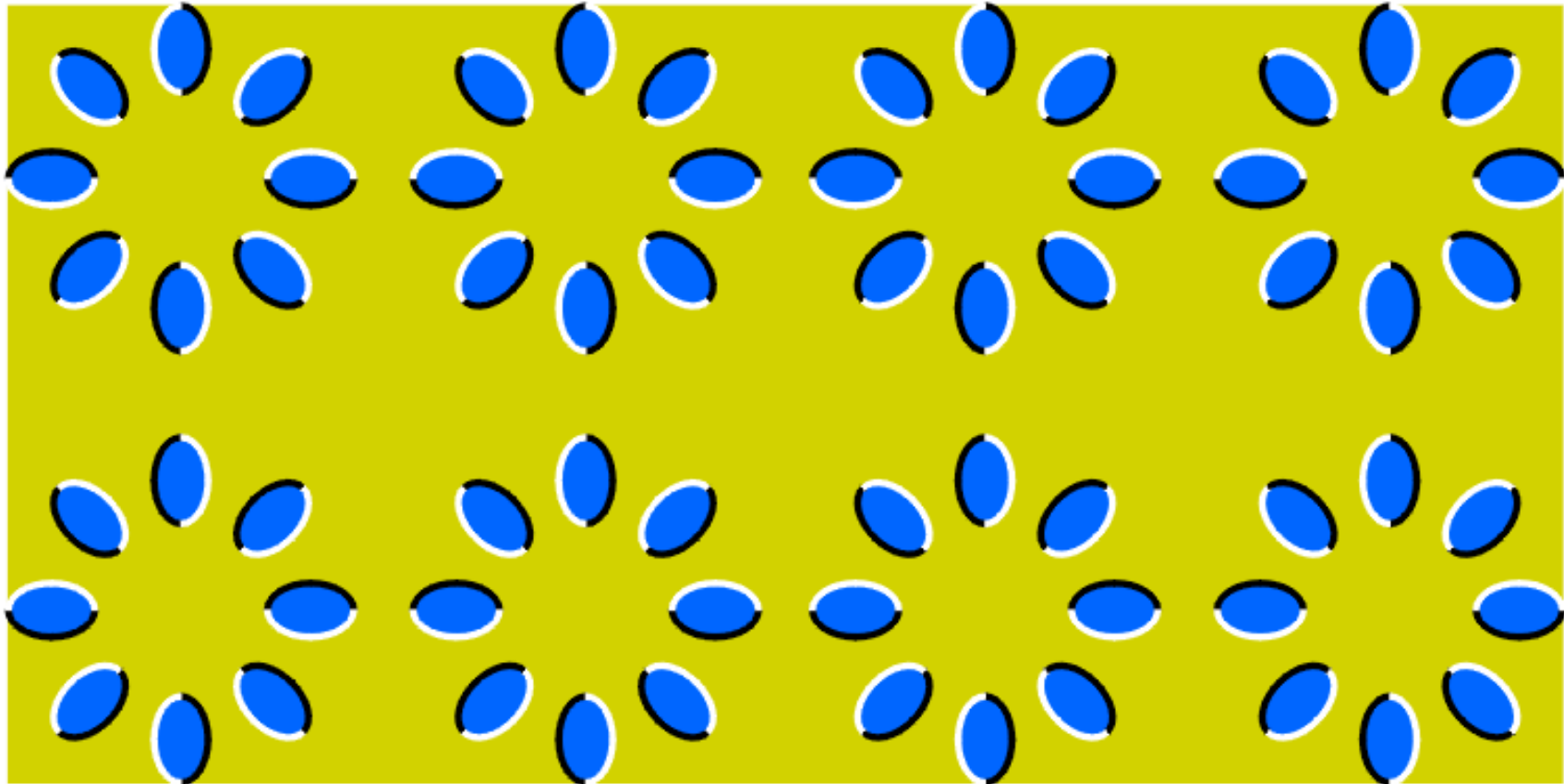
- Scaled orthographic
 - Also called “weak perspective”

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

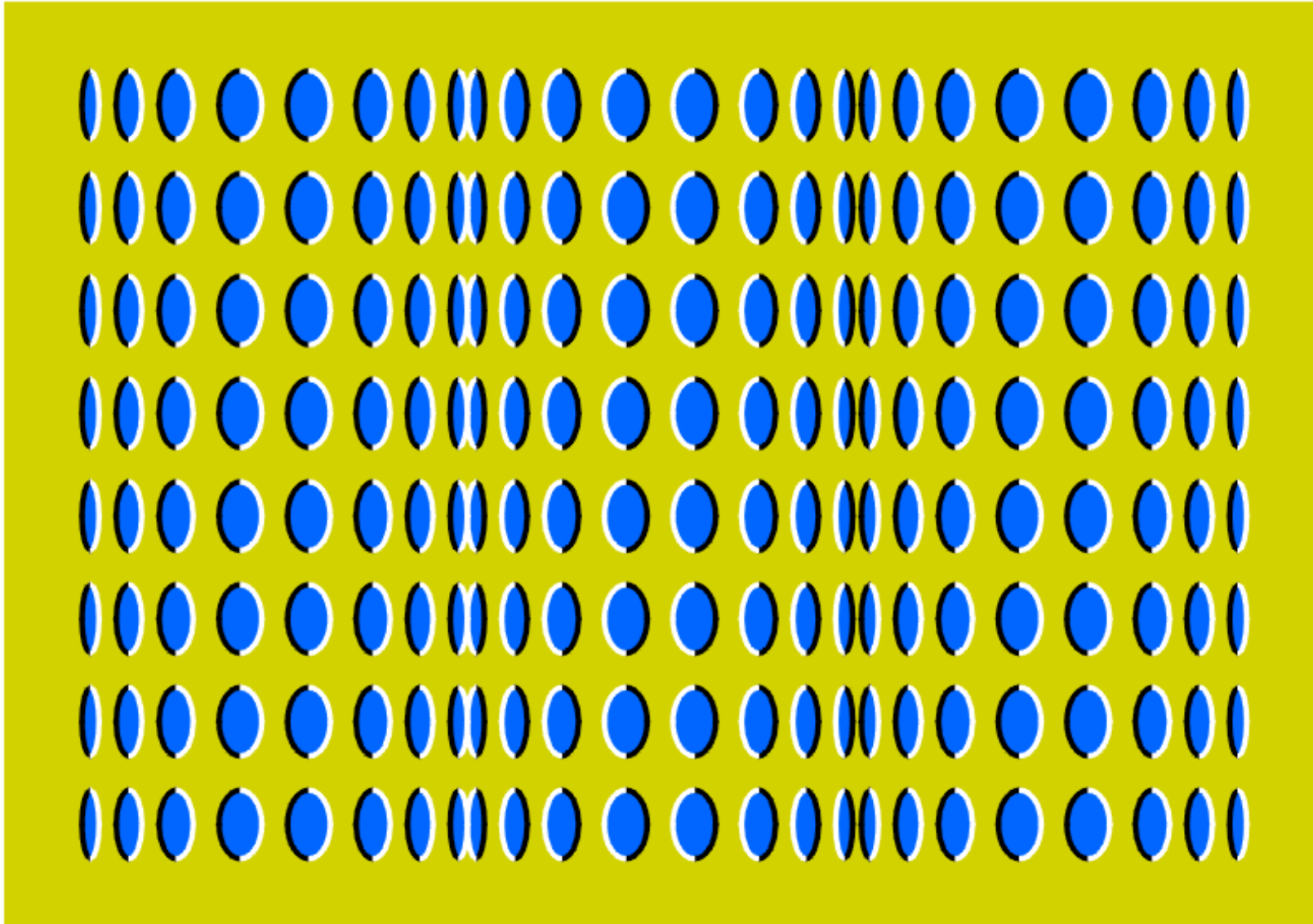
- Affine projection
 - Also called “paraperspective”

$$\begin{bmatrix} a & b & c & d \\ e & f & g & h \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

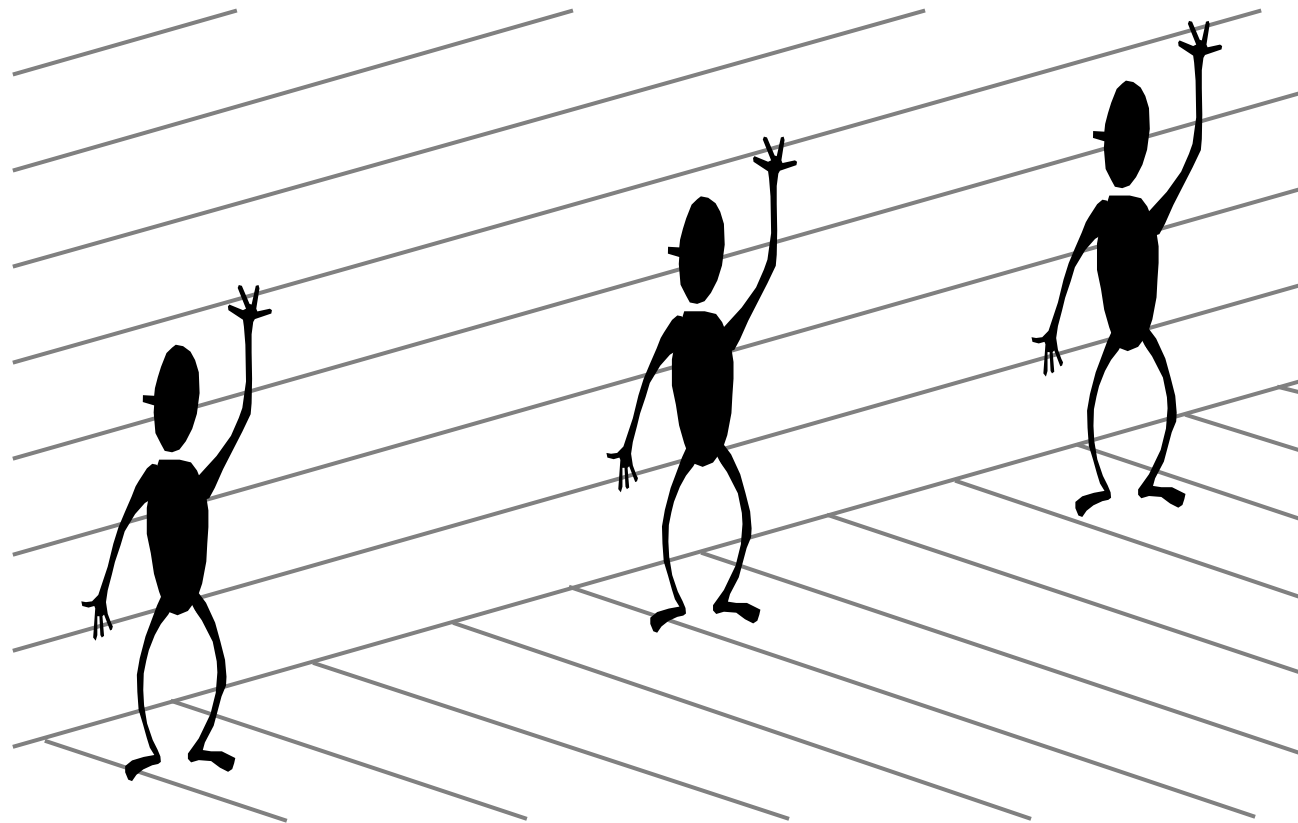
Illusion



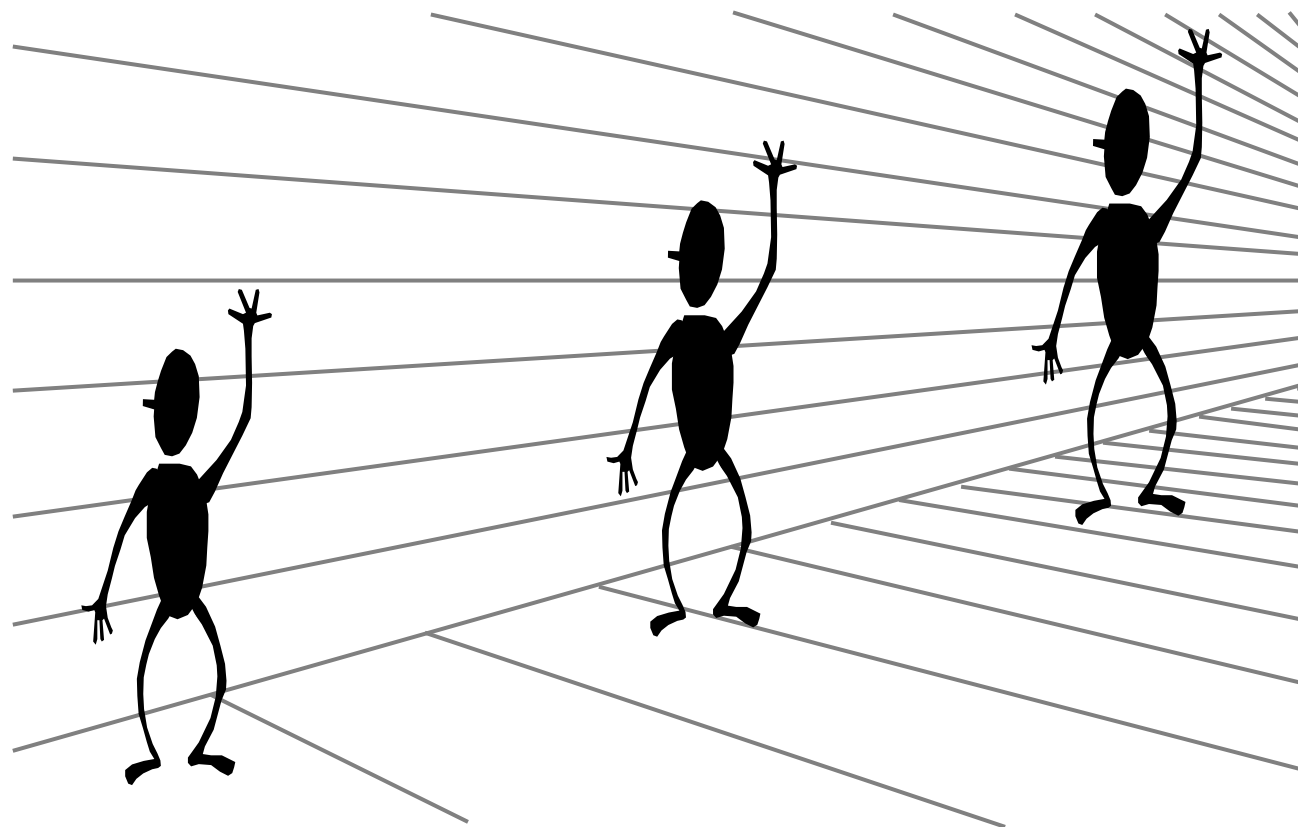
Illusion



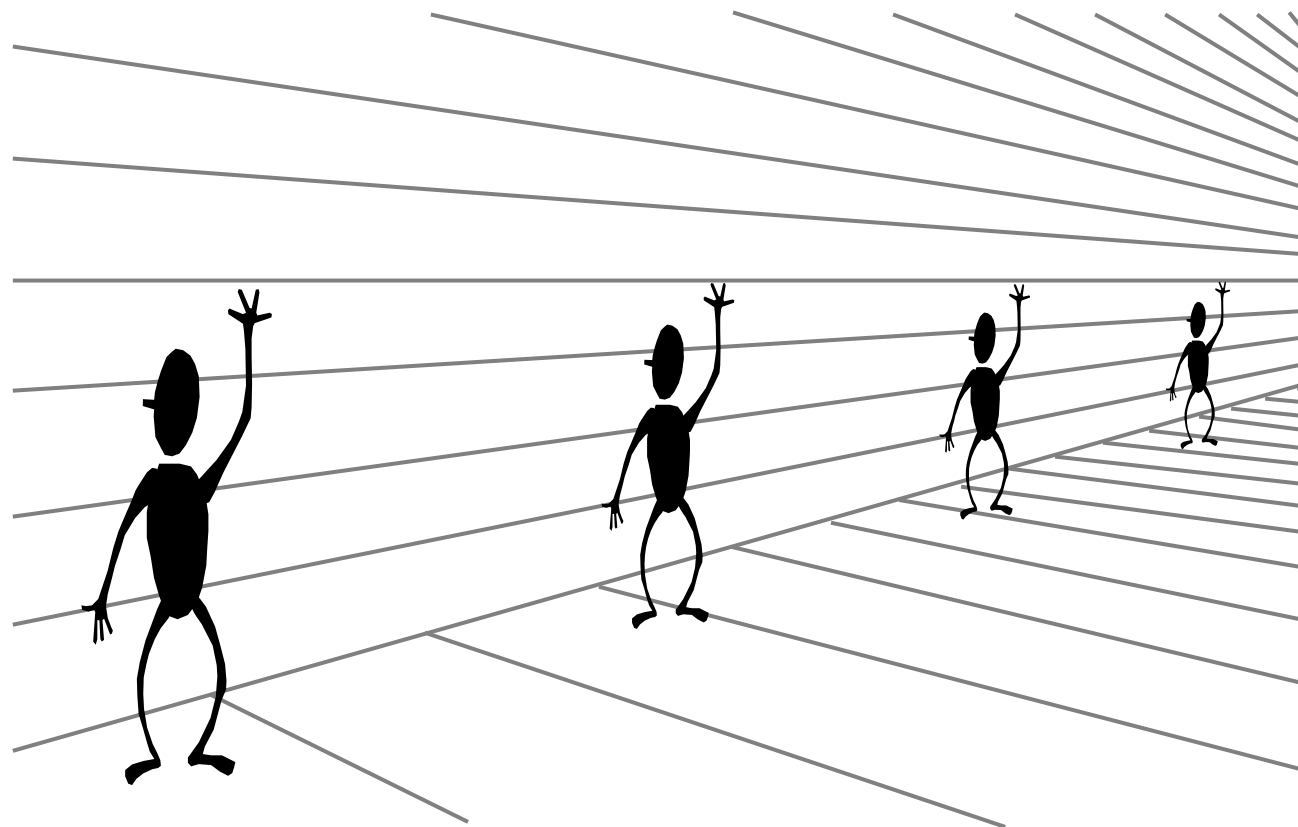
Fun with perspective



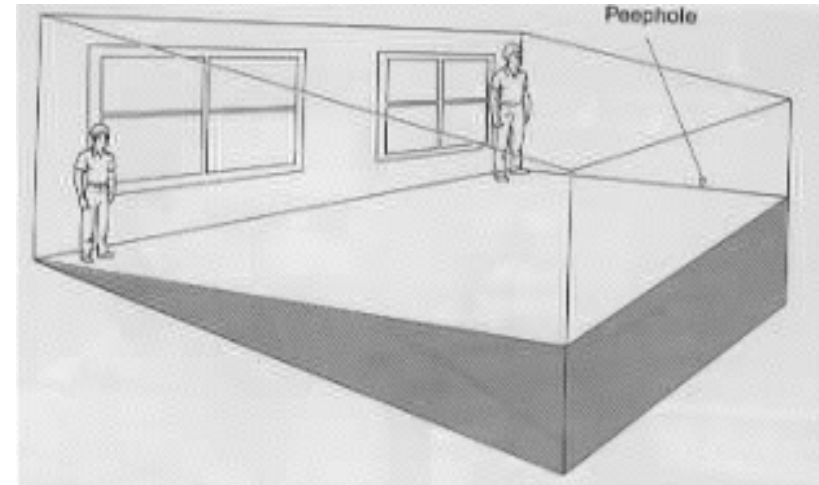
Perspective cues



Perspective cues



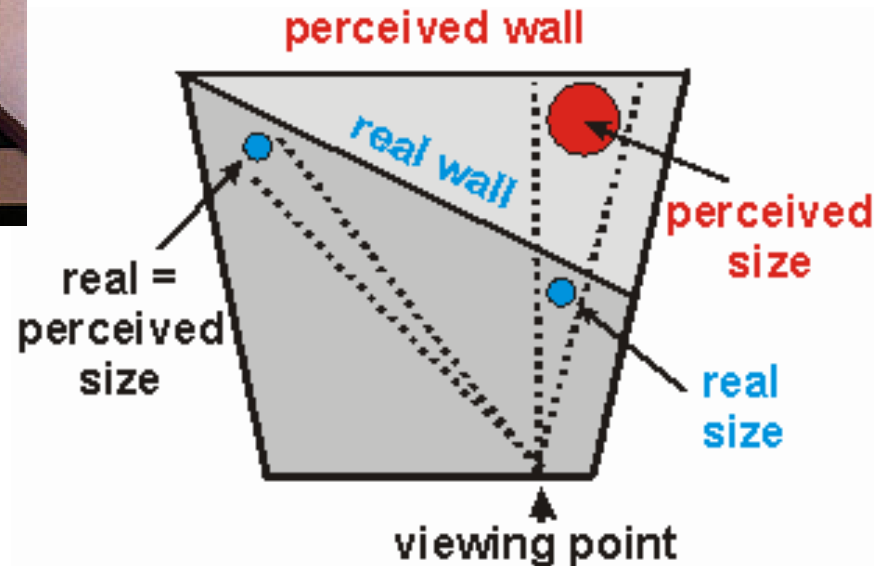
Fun with perspective



[Ames room](#)

[Ames video](#)

[BBC story](#)



Forced perspective in LOTR



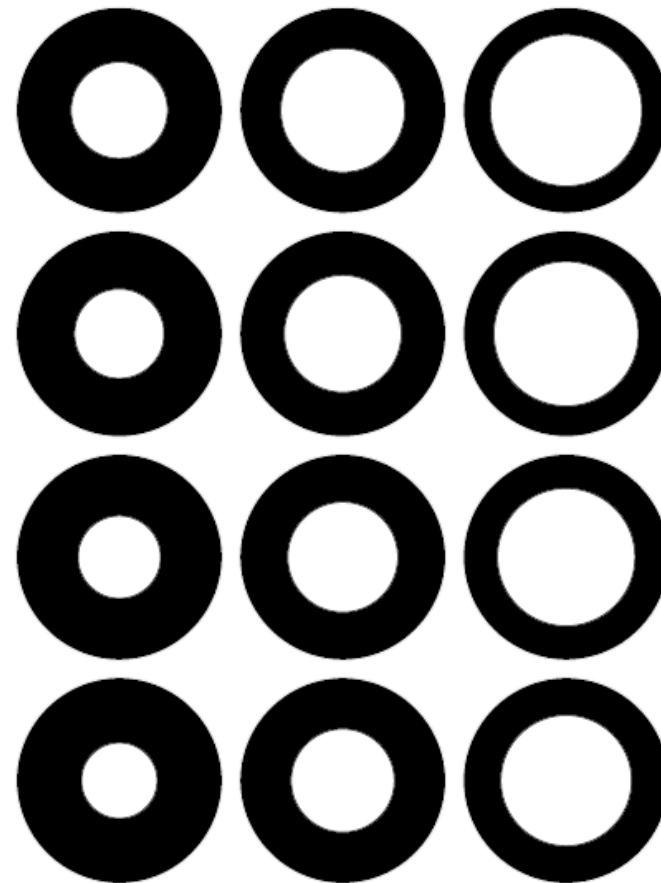
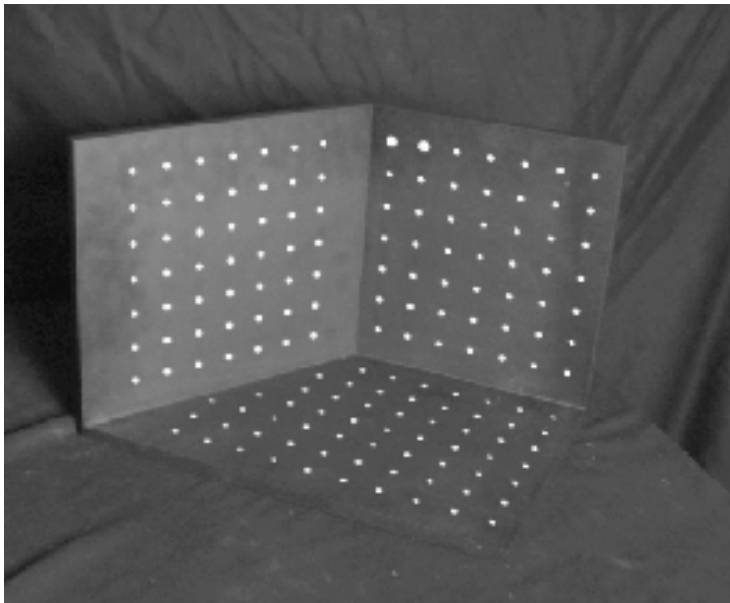
Camera calibration

Camera calibration

- Estimate both intrinsic and extrinsic parameters
- Mainly, two categories:
 1. Photometric calibration: uses reference objects with known geometry
 2. Self calibration: only assumes static scene, e.g. structure from motion

Camera calibration approaches

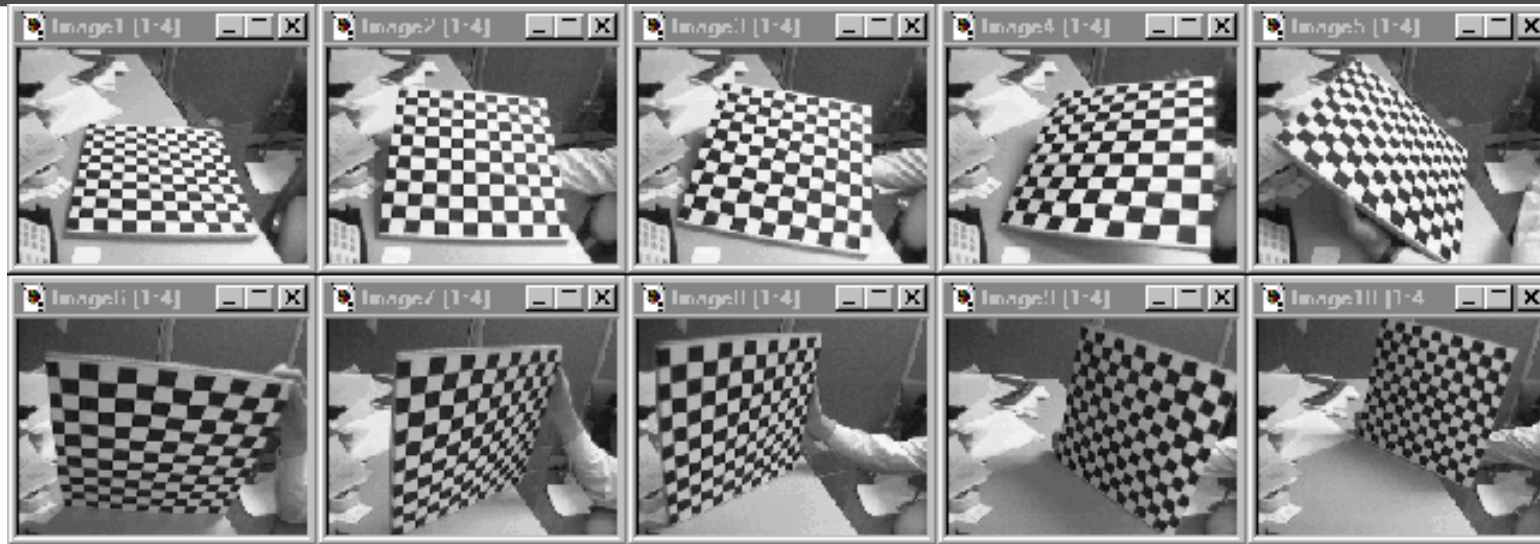
1. linear regression (least squares)
2. nonlinear optimization



Chromaglyphs (HP research)



Multi-plane calibration

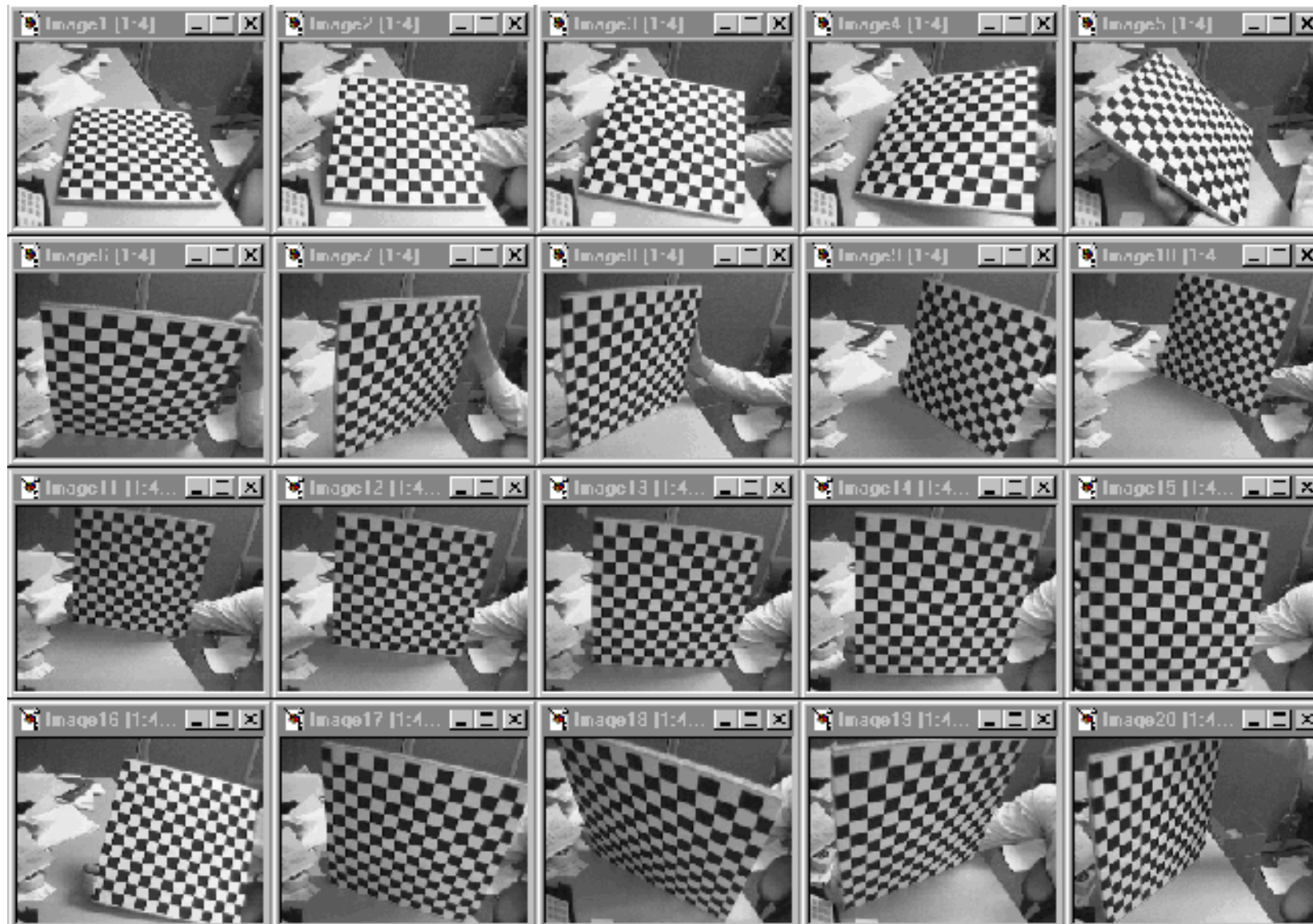


Images courtesy Jean-Yves Bouguet, Intel Corp.

Advantage

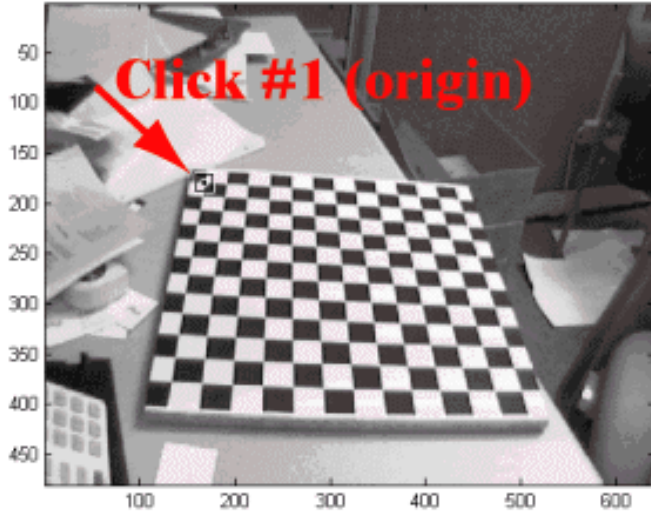
- Only requires a plane
- Don't have to know positions/orientations
- Good code available online!
 - Intel's OpenCV library: <http://www.intel.com/research/mrl/research/opencv/>
 - Matlab version by Jean-Yves Bouguet: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html
 - Zhengyou Zhang's web site: <http://research.microsoft.com/~zhang/Calib/>

Step 1: data acquisition

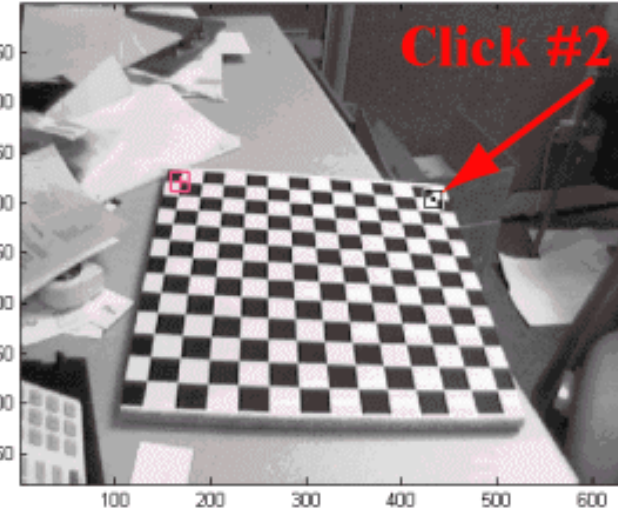


Step 2: specify corner order

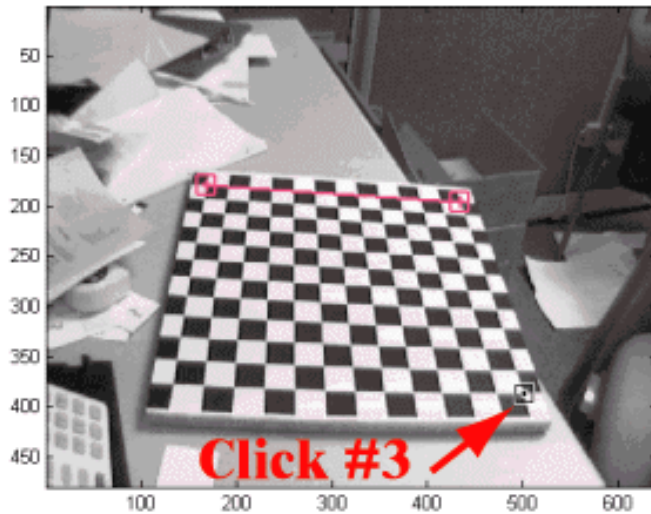
Click on the four extreme corners of the rectangular pattern (first corner = origin)... Image 1



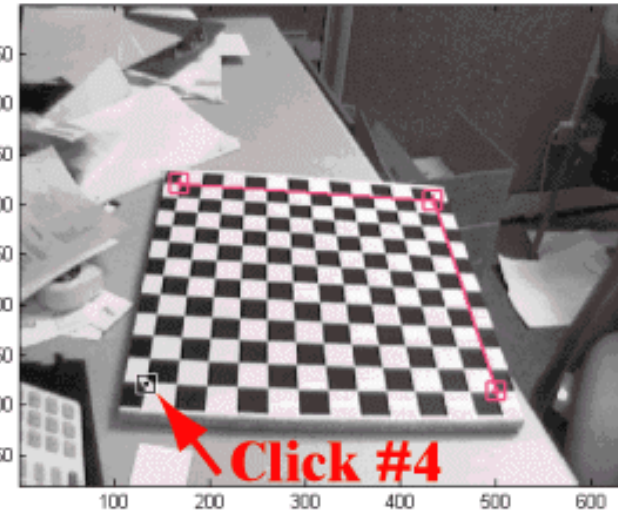
Click on the four extreme corners of the rectangular pattern (first corner = origin)... Image 1



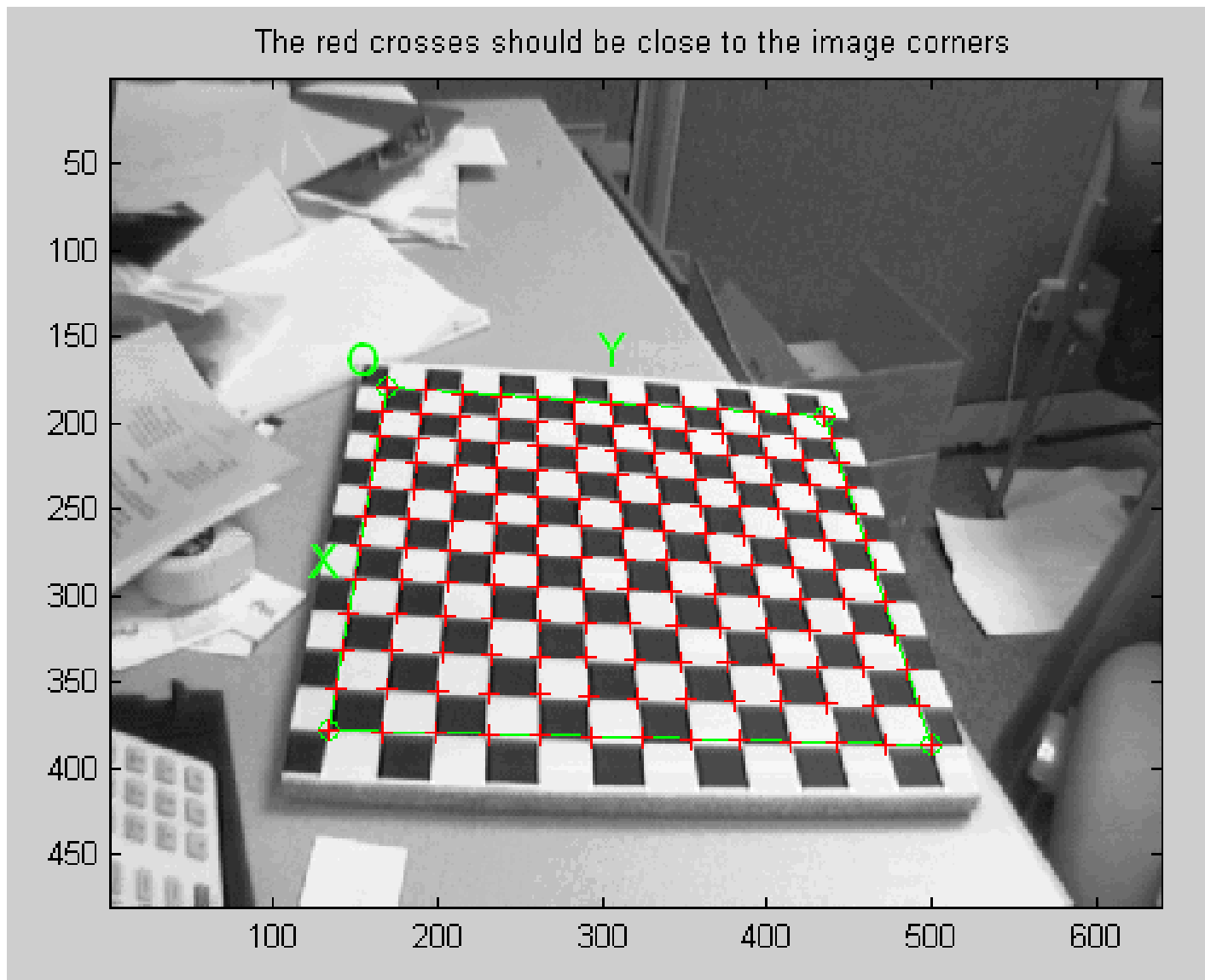
Click on the four extreme corners of the rectangular pattern (first corner = origin)... Image 1



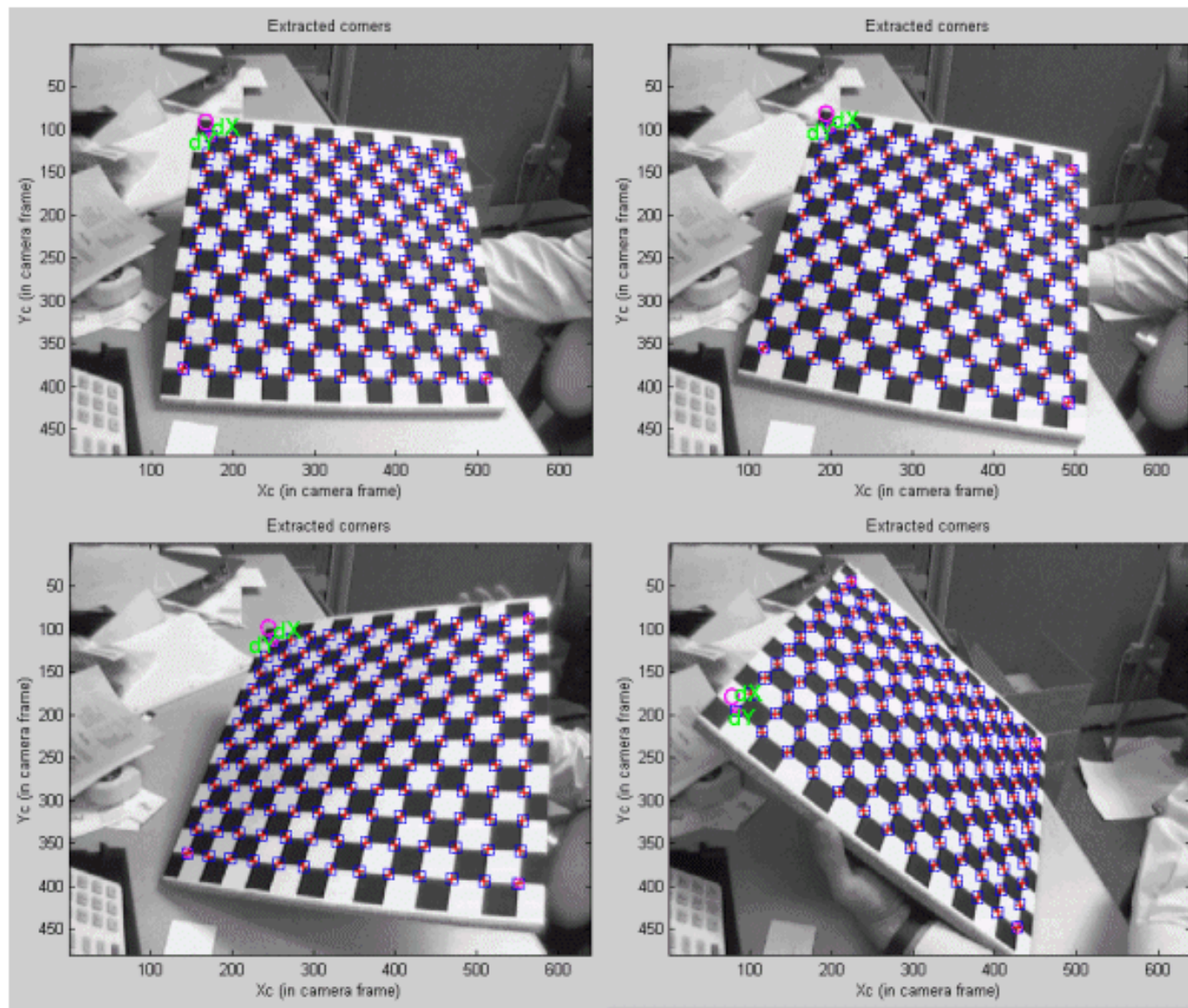
Click on the four extreme corners of the rectangular pattern (first corner = origin)... Image 1



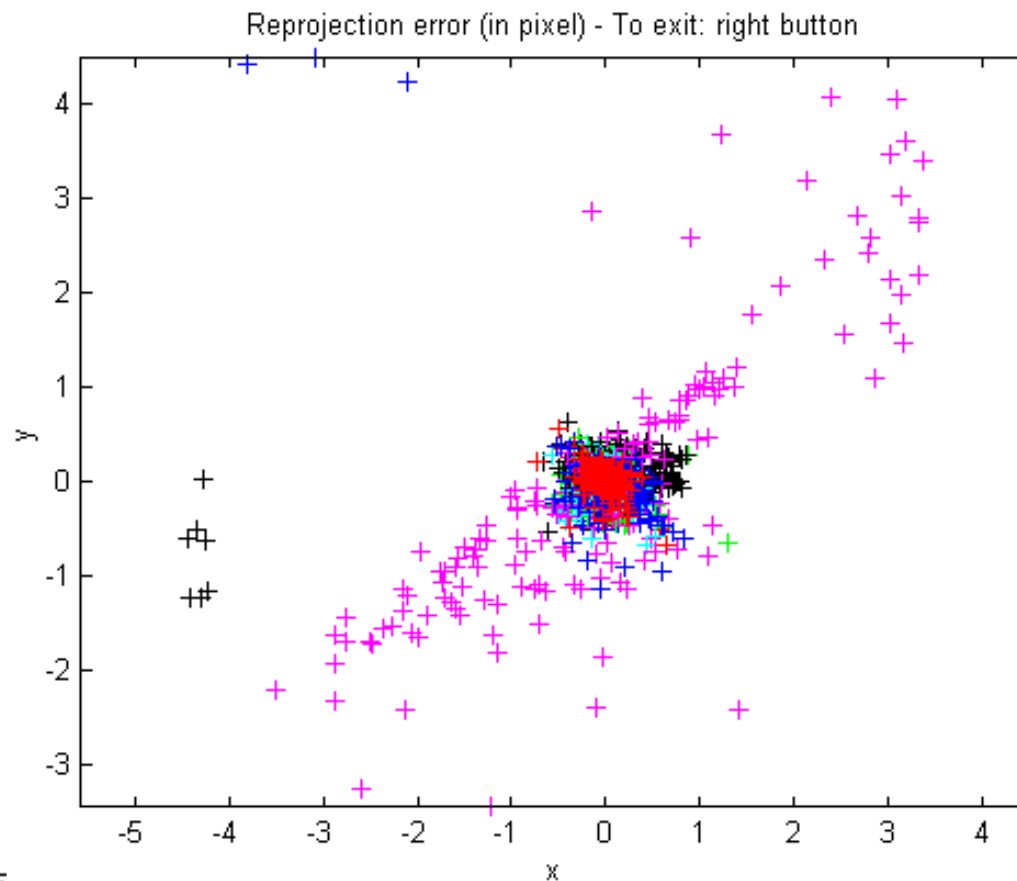
Step 3: corner extraction



Step 3: corner extraction



Step 4: minimize projection error

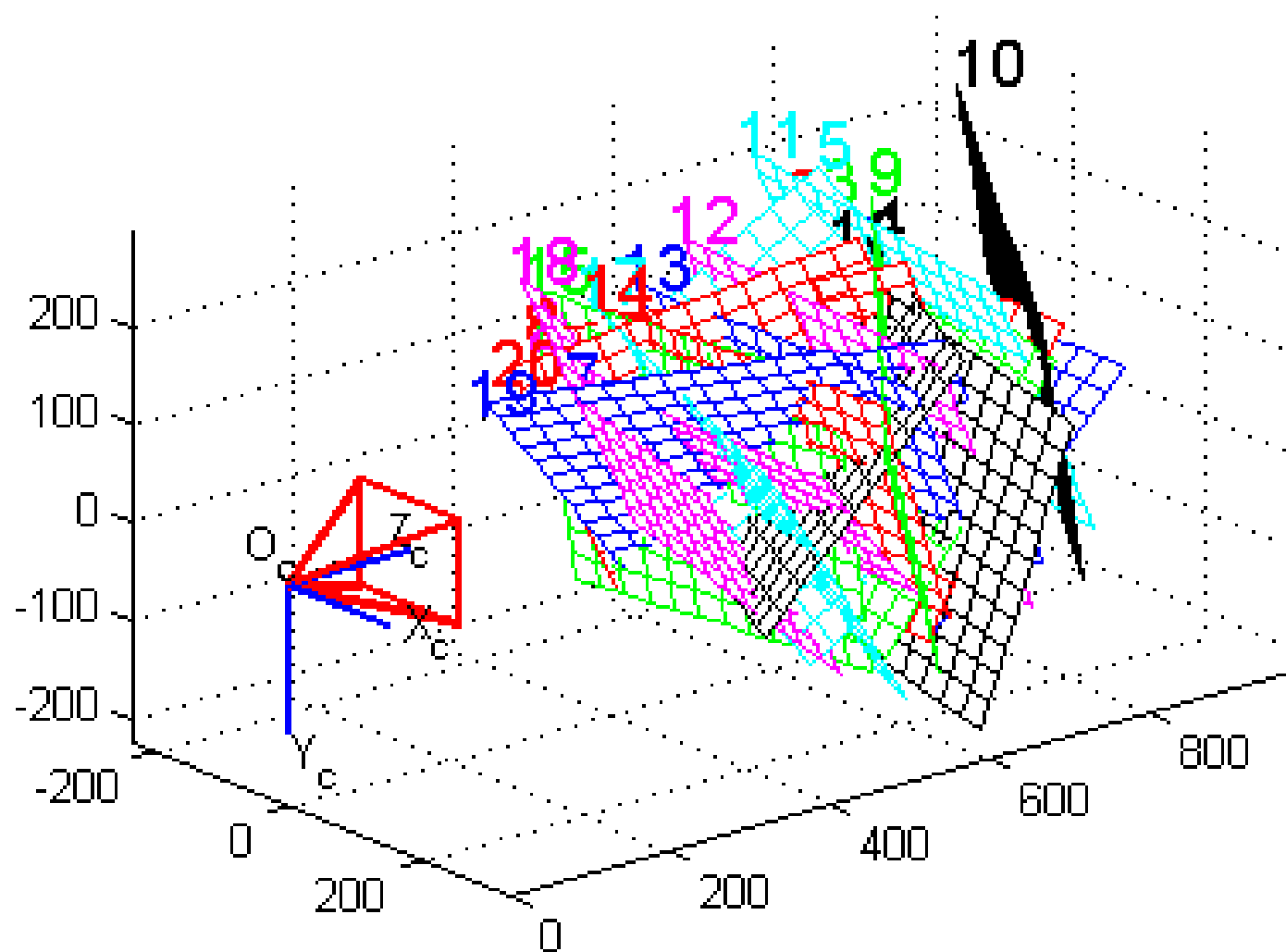


Calibration res

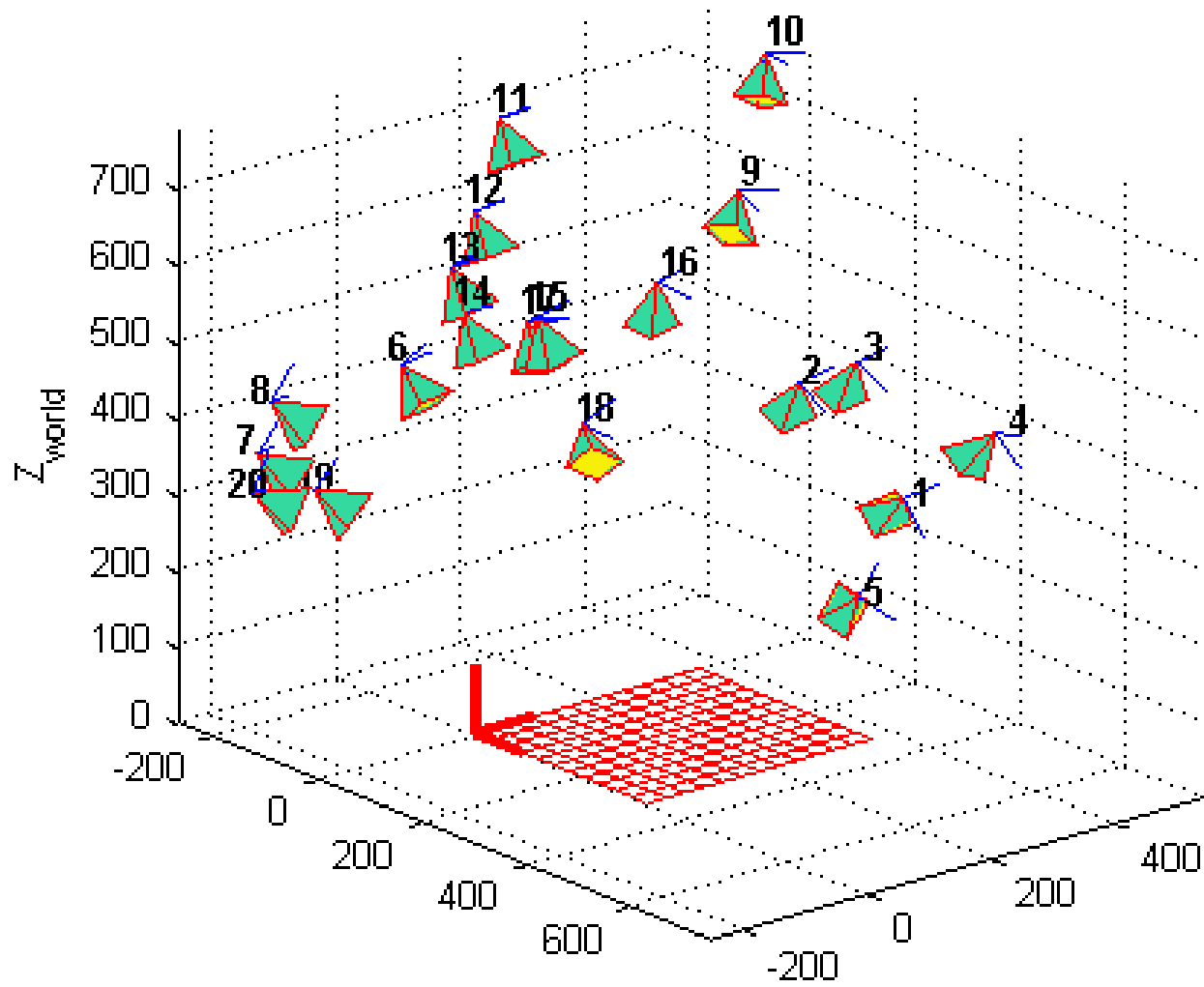
```

Focal Length:      fc = [ 657.46290  657.94673 ] ± [ 0.31819  0.34046 ]
Principal point:   cc = [ 303.13665  242.56935 ] ± [ 0.64682  0.59218 ]
Skew:              alpha_c = [ 0.00000 ] ± [ 0.00000 ] => angle of pixel axes =
Distortion:        kc = [ -0.25403  0.12143  -0.00021  0.00002  0.00000 ]
Pixel error:       err = [ 0.11689  0.11500 ]
    
```

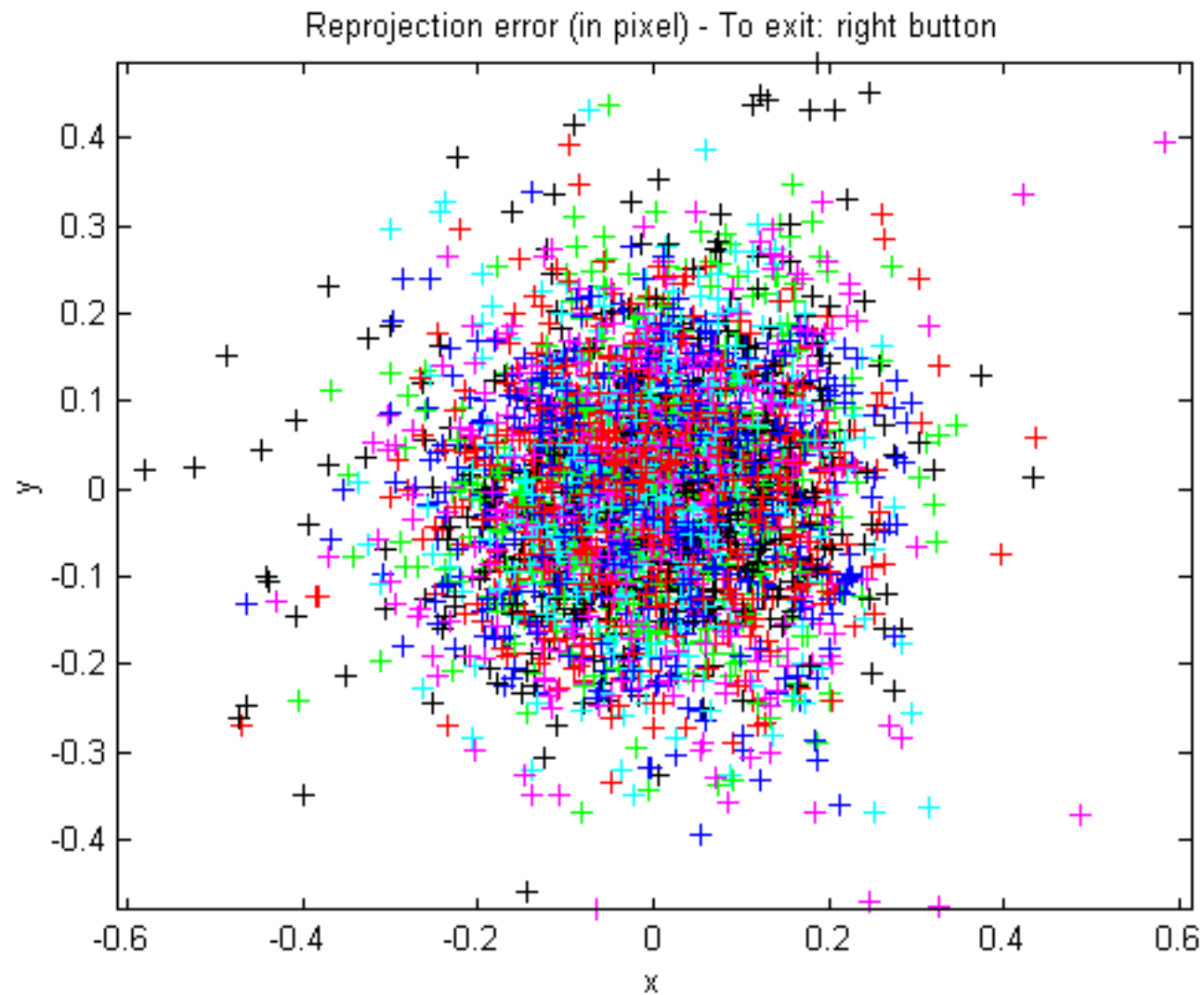

Step 4: camera calibration



Step 4: camera calibration



Step 5: refinement



Optimized parameters

Aspect ratio optimized (`est_aspect_ratio = 1`) -> both components of `fc` are estimated (DEI
 Principal point optimized (`center_optim=1`) - (DEFAULT). To reject principal point, set `center_optim=0`
 Skew not optimized (`est_alpha=0`) - (DEFAULT)
 Distortion not fully estimated (defined by the variable `est_dist`):
 Sixth order distortion not estimated (`est_dist(5)=0`) - (DEFAULT) .

Main calibration optimization procedure - Number of images: 20
 Gradient descent iterations: 1...2...3...4...5...done
 Estimation of uncertainties...done

Calibration results after optimization (with uncertainties):

```
Focal Length:      fc = [ 657.46290  657.94673 ] ± [ 0.31819  0.34046 ]
Principal point:   cc = [ 303.13665  242.56935 ] ± [ 0.64682  0.59218 ]
Skew:              alpha_c = [ 0.00000 ] ± [ 0.00000 ] => angle of pixel axes = 90.000
Distortion:        kc = [ -0.25403  0.12143  -0.00021  0.00002  0.00000 ] ± [ 0.0
Pixel error:       err = [ 0.11689  0.11500 ]
```

Note: The numerical errors are approximately three times the standard deviations (for re

Camera calibration

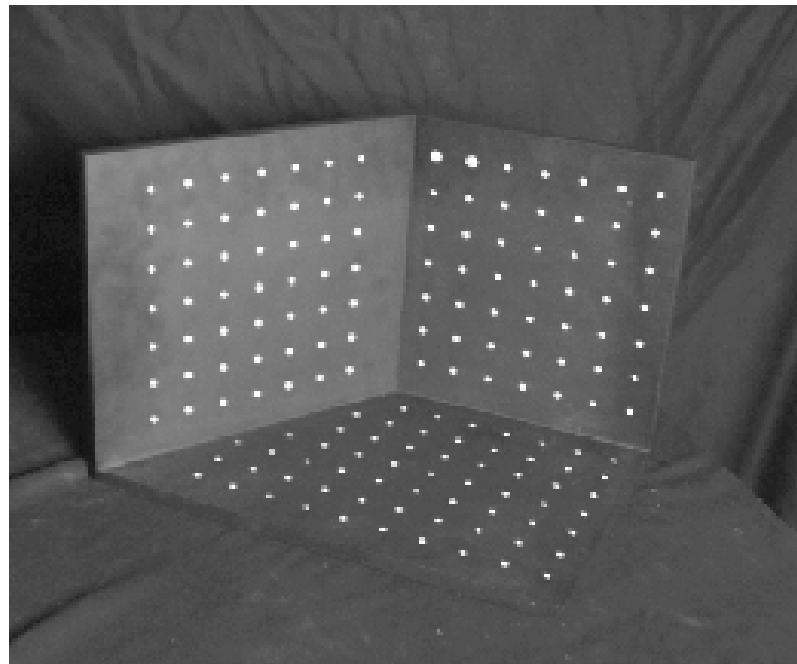
Linear regression

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

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

Linear regression

- 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

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

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

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

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

Linear regression

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

Linear regression

$$\begin{bmatrix}
 X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1 X_1 & -u_1 Y_1 & -u_1 Z_1 & -u_1 \\
 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1 X_1 & -v_1 Y_1 & -v_1 Z_1 & -v_1 \\
 & & & & & & & \vdots & & & & \\
 X_n & Y_n & Z_n & 1 & 0 & 0 & 0 & 0 & -u_n X_n & -u_n Y_n & -u_n Z_n & -u_n \\
 0 & 0 & 0 & 0 & X_n & Y_n & Z_n & 1 & -v_n X_n & -v_n Y_n & -v_n Z_n & -v_n
 \end{bmatrix}
 \begin{bmatrix}
 m_{00} \\
 m_{01} \\
 m_{02} \\
 m_{03} \\
 m_{10} \\
 m_{11} \\
 m_{12} \\
 m_{13} \\
 m_{20} \\
 m_{21} \\
 m_{22}
 \end{bmatrix}
 =
 \begin{bmatrix}
 0 \\
 0 \\
 \vdots \\
 0 \\
 0
 \end{bmatrix}$$

Solve for Projection Matrix M using least-square techniques

Normal equation

Given an overdetermined system

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

the normal equation is that which minimizes the sum of the square differences between left and right sides

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

Linear regression

- Advantages:
 - All specifics of the camera summarized in one matrix
 - Can predict where any world point will map to in the image
- Disadvantages:
 - Doesn't tell us about particular parameters
 - Mixes up internal and external parameters
 - pose specific: move the camera and everything breaks
 - More unknowns than true degrees of freedom

Nonlinear 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 $\{(u_i, v_i)\}$

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

Optimal estimation

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

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

- It is a least square problem (but not necessarily linear least square)
- How do we minimize L ?

Optimal estimation

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

unknown parameters

We could have terms like $f \cos \theta$ in this

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

known constant

- We can use Levenberg-Marquardt method to minimize it

Nonlinear least square methods

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

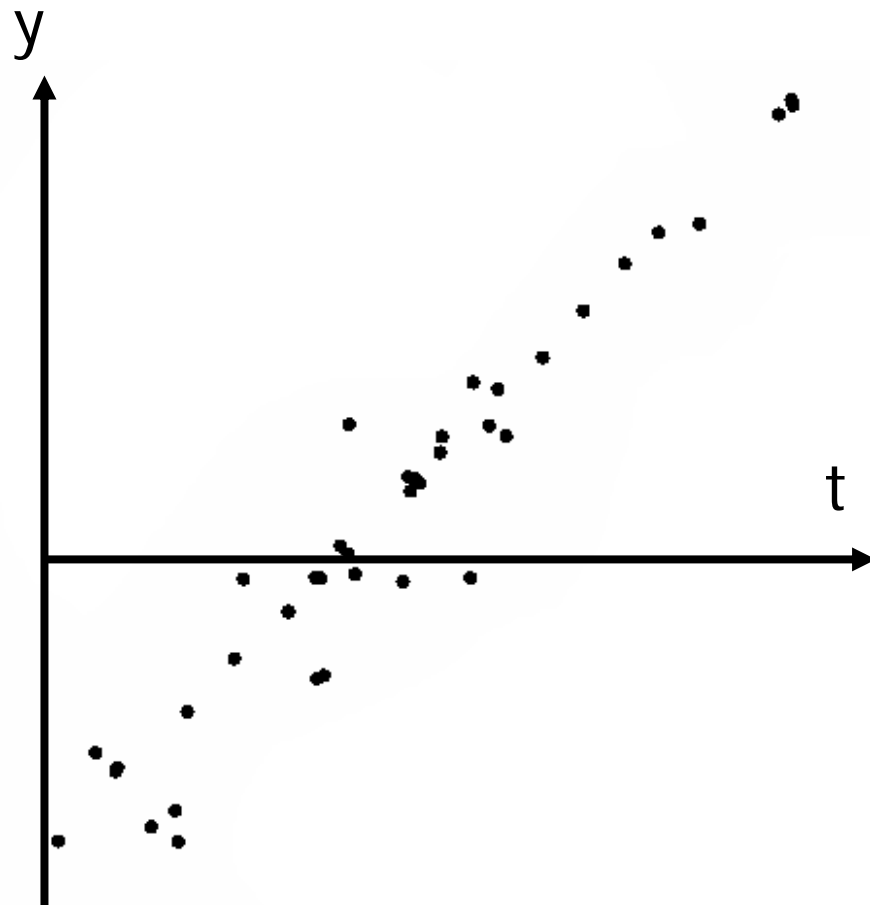
number of data points



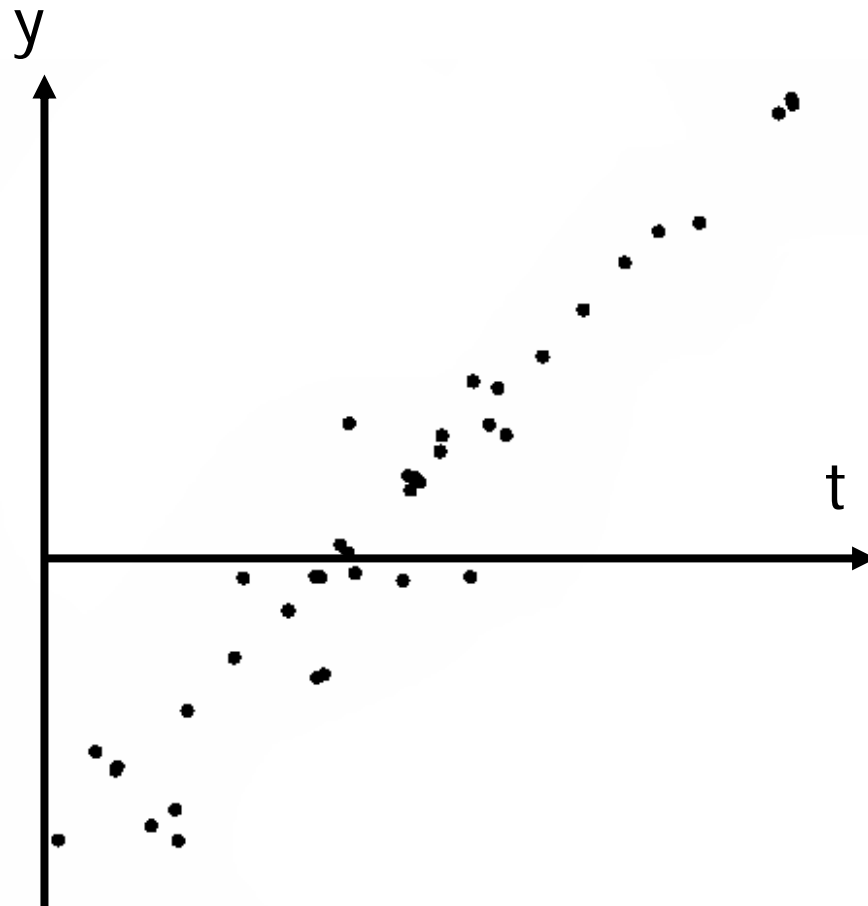
number of parameters



Linear least square fitting



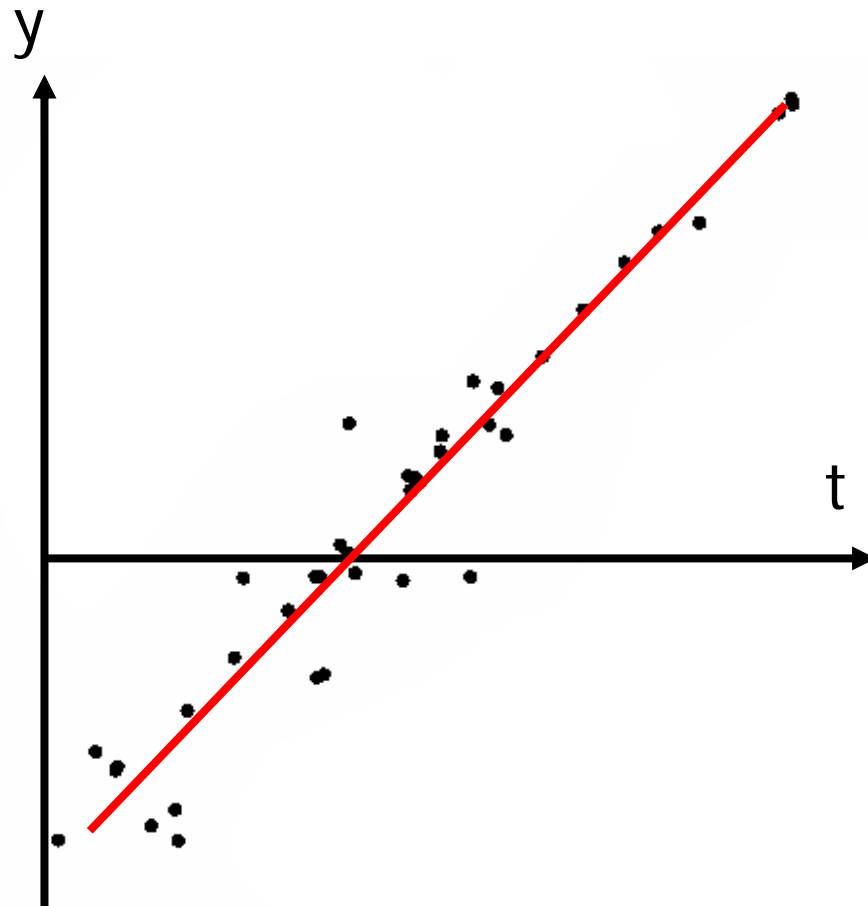
Linear least square fitting



model parameters

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

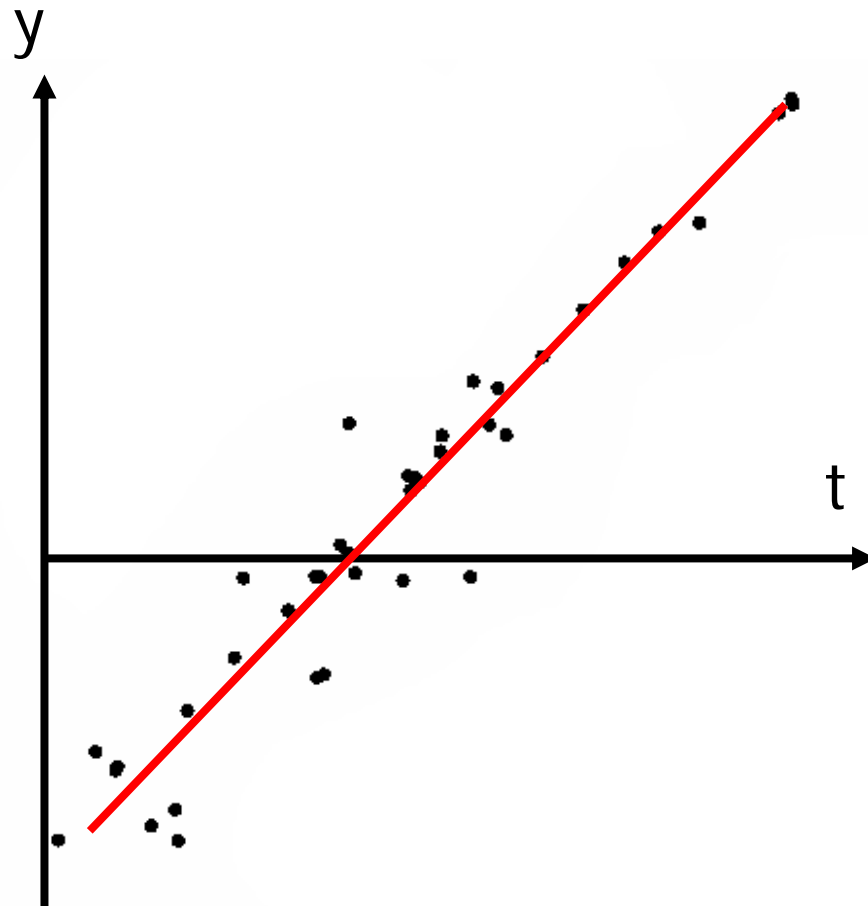
Linear least square fitting



model parameters

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

Linear least square fitting



model parameters

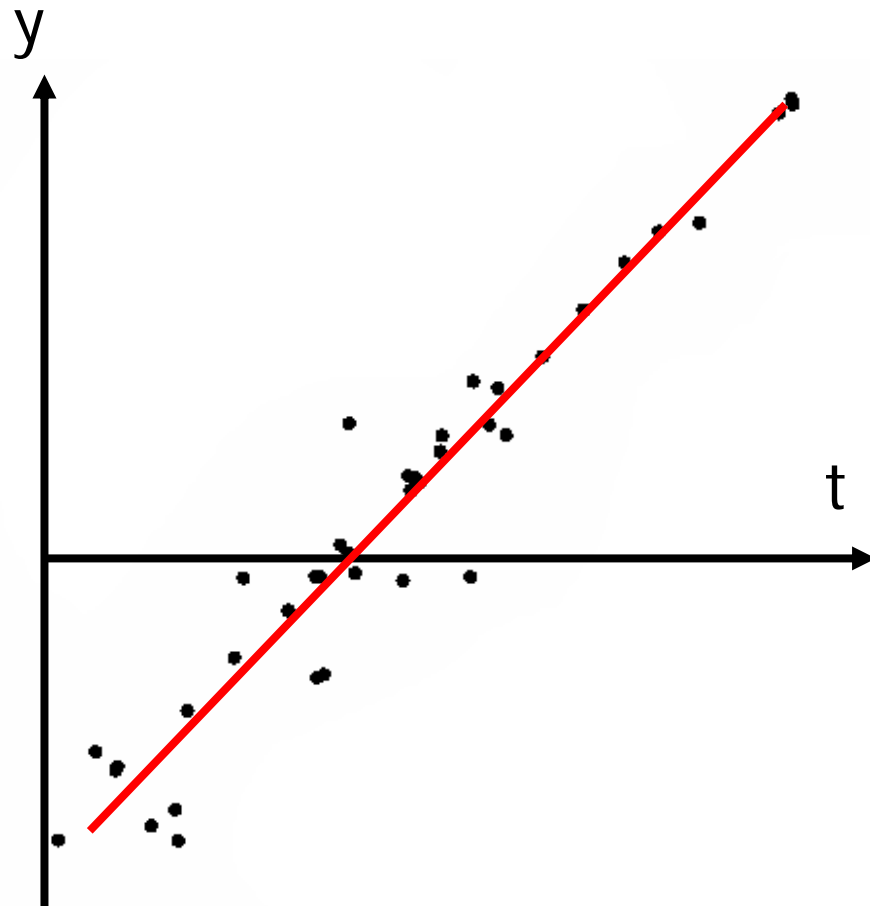
$$y(t) = M(t; \mathbf{x}) = x_0 + x_1 t$$

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

residual

prediction

Linear least square fitting



model parameters

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

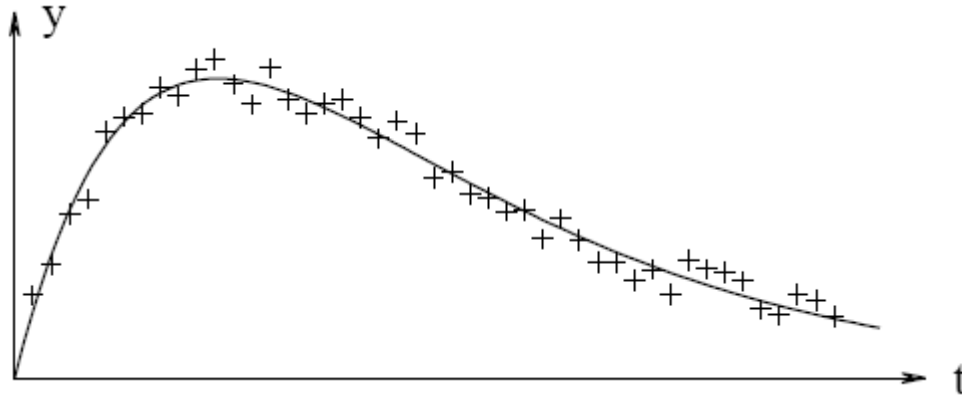
$$f_i(x) = y_i - M(t_i; \mathbf{x})$$

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

residuals $f_i(\mathbf{x}) = y_i - M(t_i; \mathbf{x})$
 $= y_i - (x_3 e^{x_1 t} + x_4 e^{x_2 t})$

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⁾

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

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

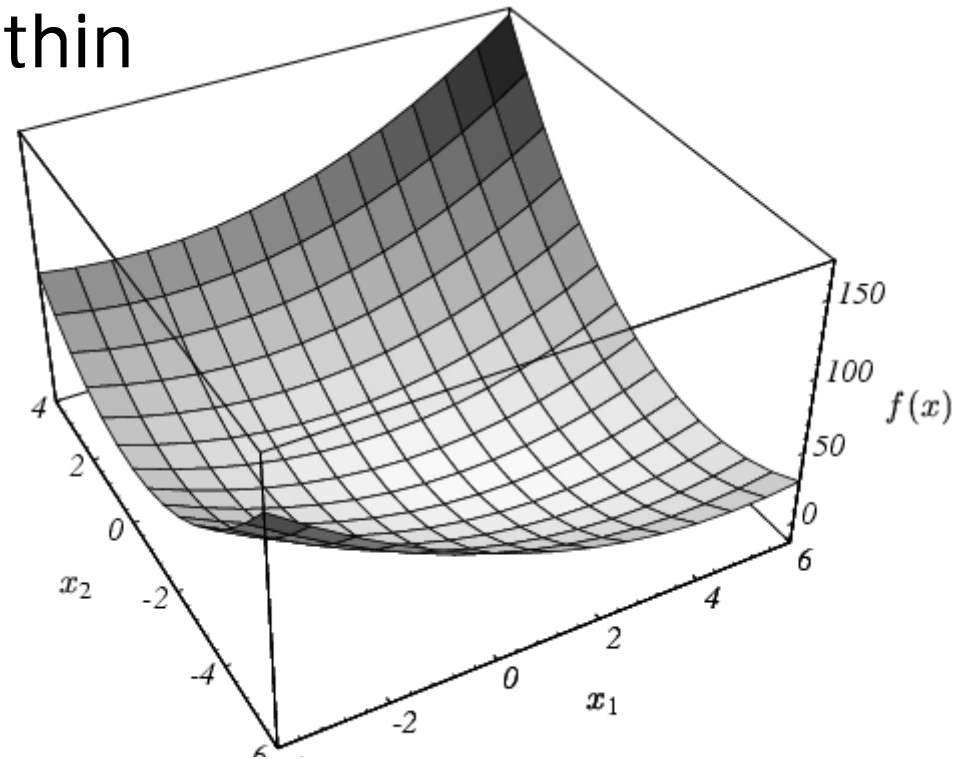
$$\mathbf{g} \equiv \mathbf{F}'(\mathbf{x}) = \begin{bmatrix} \frac{\partial F}{\partial x_1}(\mathbf{x}) \\ \vdots \\ \frac{\partial F}{\partial x_n}(\mathbf{x}) \end{bmatrix},$$

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

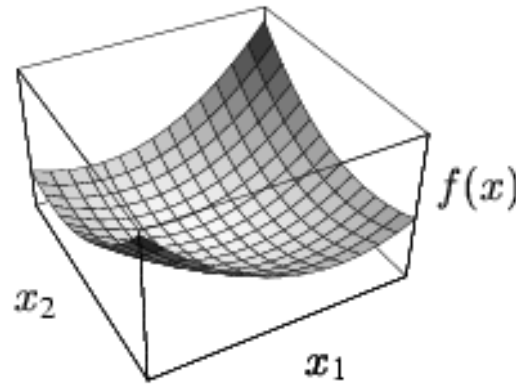


$$f(x) = \frac{1}{2}x^T A x - b^T x + c$$

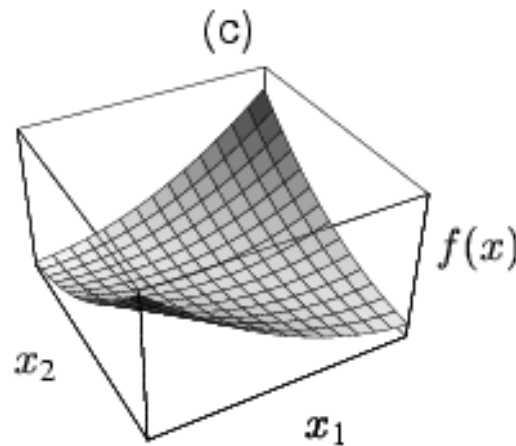
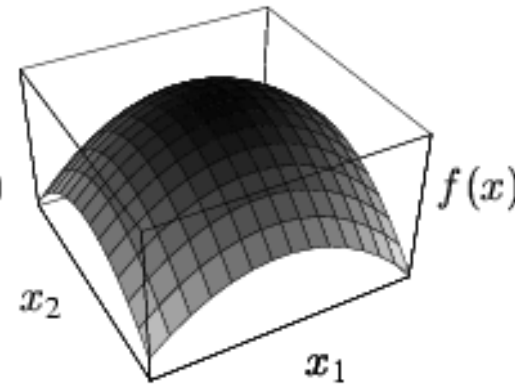
$$A = \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}, \quad b = \begin{bmatrix} 2 \\ -8 \end{bmatrix}, \quad c = 0.$$

Quadratic functions

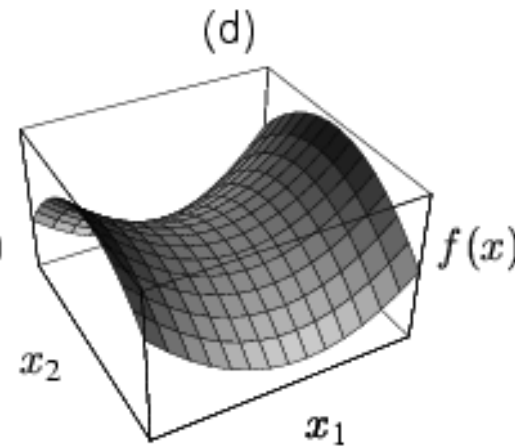
A is positive definite. (a)
 All eigenvalues
 are positive.
 For all x ,
 $x^T A x > 0$.



(b) negative definite



A is singular



A is indefinite

Function minimization

Theorem 1.5. Necessary condition for a local minimizer.

If \mathbf{x}^* is a local minimizer, then

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

Why?

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

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

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

Function minimization

Theorem 1.5. Necessary condition for a local minimizer.

If \mathbf{x}^* is a local minimizer, then

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

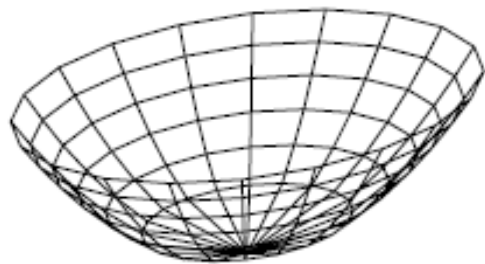
Definition 1.6. Stationary point. If

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

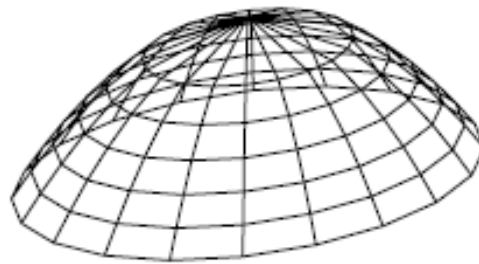
then \mathbf{x}_s is said to be a *stationary point* for F .

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

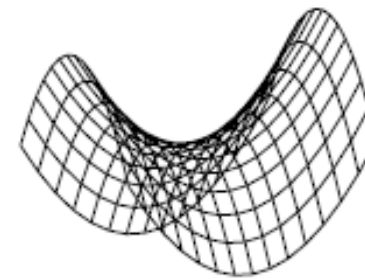
\mathbf{H}_s is *positive definite*.



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.

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

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

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

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

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

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

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

if (no such \mathbf{h} exists)

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

else

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

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

end

Descent direction

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

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

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

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

the decrease of $F(\mathbf{x})$ per
unit along \mathbf{h} direction

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

\mathbf{h}_{sd} is a descent direction because $\mathbf{h}_{\text{sd}}^\top \mathbf{F}'(\mathbf{x}) = -\|\mathbf{F}'(\mathbf{x})\|^2 < 0$

Line search

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

Find α so that

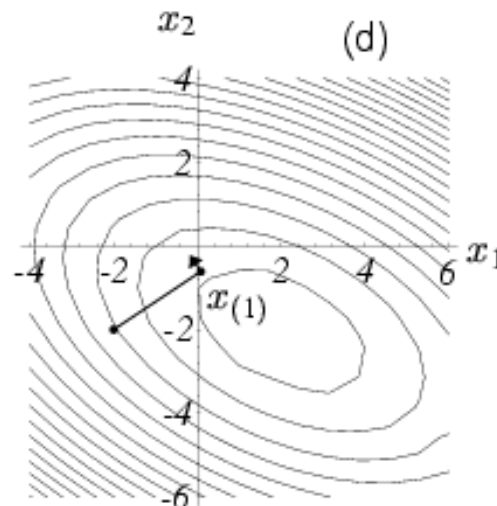
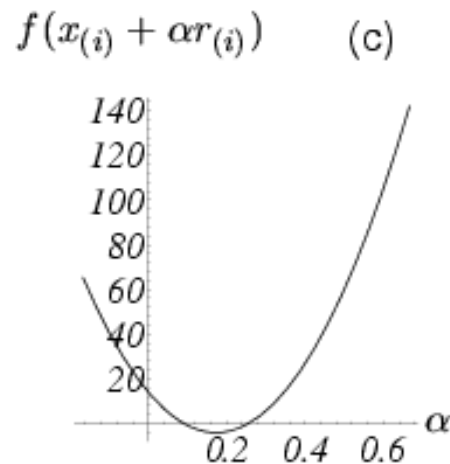
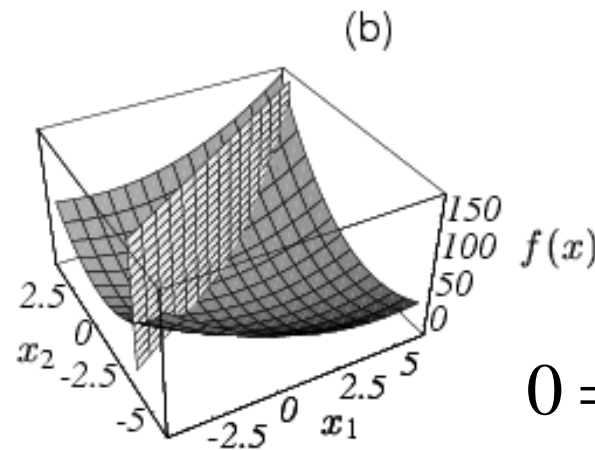
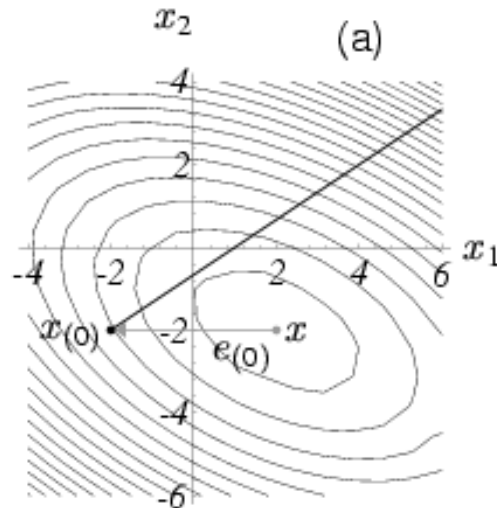
$$\varphi(\alpha) = \mathbf{F}(\mathbf{x}_0 + \alpha\mathbf{h})$$

is minimum

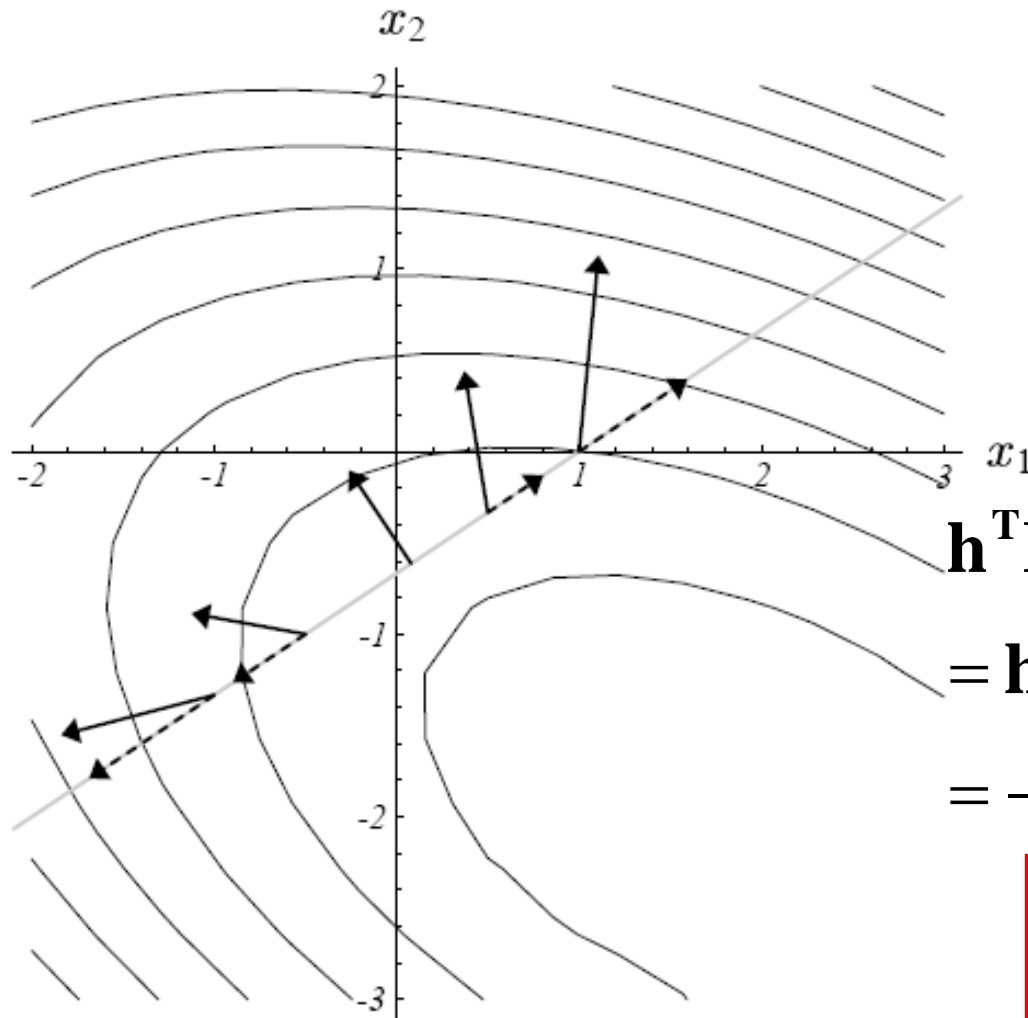
$$0 = \frac{\partial \varphi(\alpha)}{\partial \alpha} = \frac{\partial \mathbf{F}(\mathbf{x}_0 + \alpha\mathbf{h})}{\partial \alpha}$$

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

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



Line search



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

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

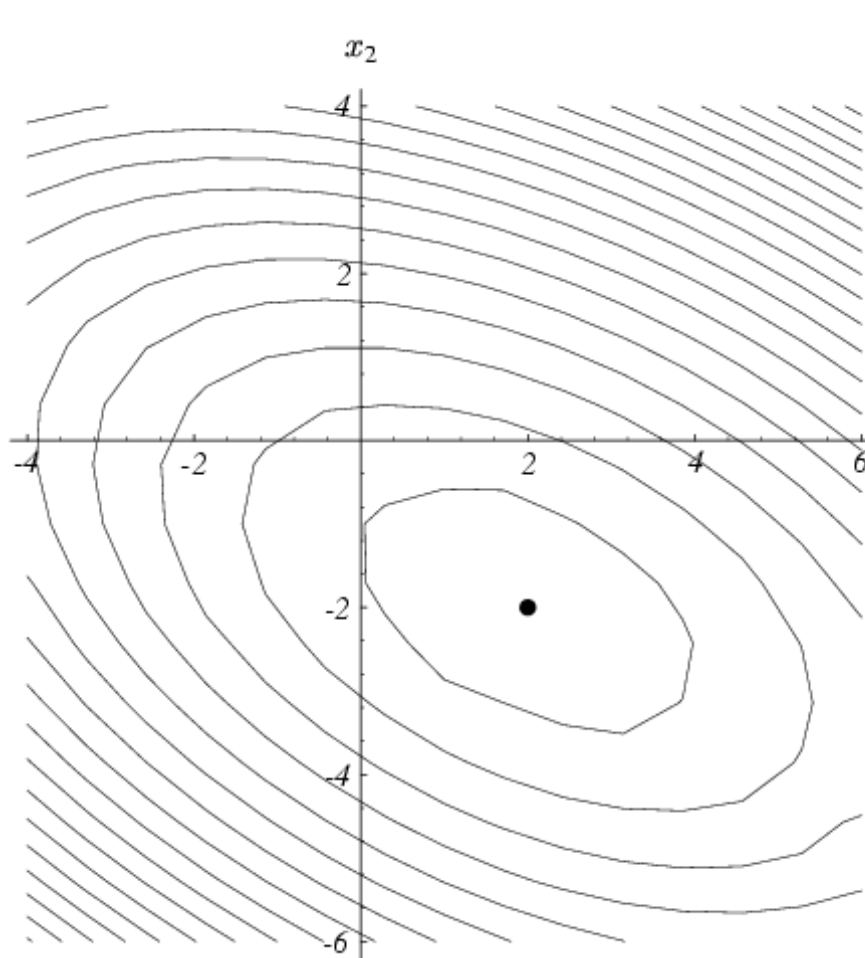
$$\mathbf{h}^T \mathbf{F}'(\mathbf{x}_0 + \alpha \mathbf{h})$$

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

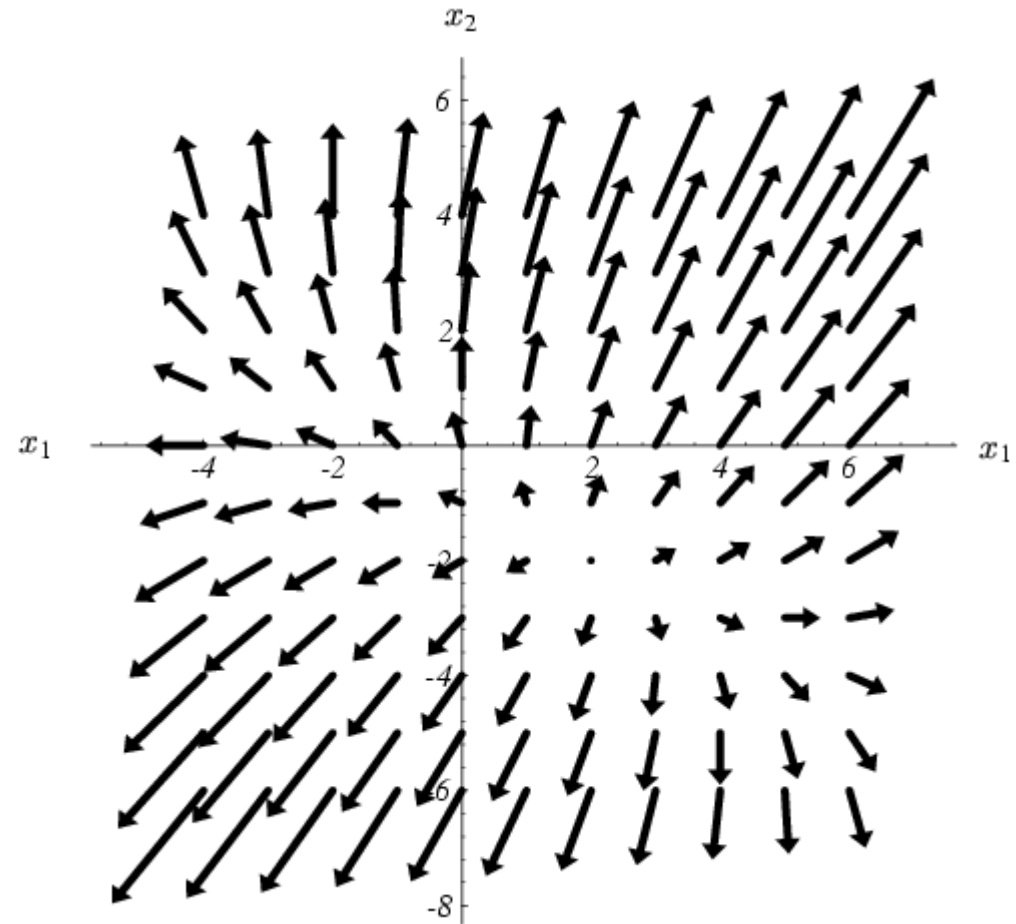
$$= -\mathbf{h}^T \mathbf{h} + \alpha \mathbf{h}^T \mathbf{H} \mathbf{h} = 0$$

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

Steepest descent method

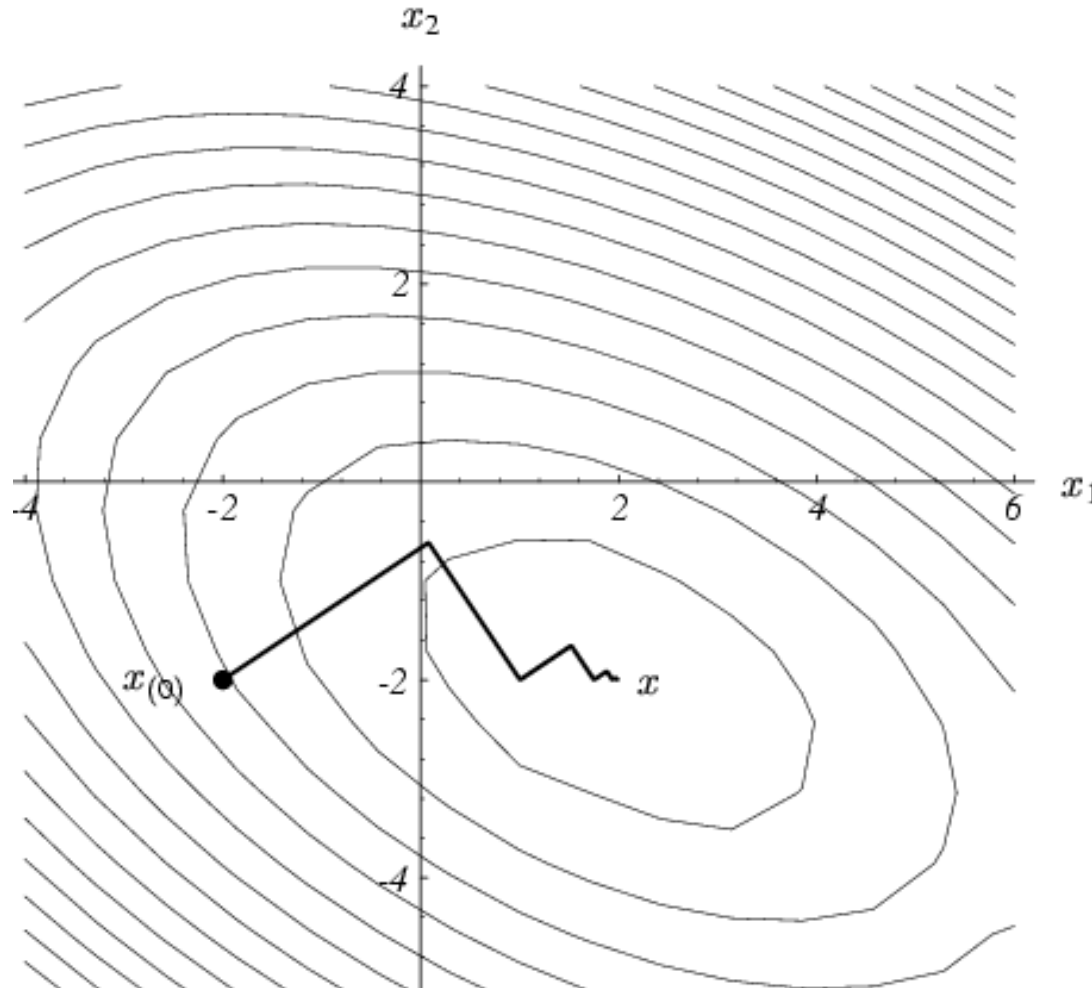


isocontour



gradient

Steepest descent method



It has good performance in the initial stage of the iterative process. Converge very slow with a linear rate.

Newton's method

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

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

$$\begin{aligned}\rightarrow \mathbf{H} \mathbf{h}_n &= -\mathbf{F}'(\mathbf{x}) \quad \text{with } \mathbf{H} = \mathbf{F}''(\mathbf{x}) \\ \mathbf{x} &:= \mathbf{x} + \mathbf{h}_n\end{aligned}$$

Suppose that \mathbf{H} is positive definite

$$\rightarrow \mathbf{u}^\top \mathbf{H} \mathbf{u} > 0 \text{ for all nonzero } \mathbf{u}.$$

$$\rightarrow 0 < \mathbf{h}_n^\top \mathbf{H} \mathbf{h}_n = -\mathbf{h}_n^\top \mathbf{F}'(\mathbf{x}) \quad \mathbf{h}_n \text{ is a descent direction}$$

It has good performance in the final stage of the iterative process, where \mathbf{x} is close to \mathbf{x}^* .

Hybrid method

```
if  $F''(\mathbf{x})$  is positive definite  
     $\mathbf{h} := \mathbf{h}_n$   
else  
     $\mathbf{h} := \mathbf{h}_{sd}$   
 $\mathbf{x} := \mathbf{x} + \alpha \mathbf{h}$ 
```

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.

Nonlinear least square

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

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

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

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

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

 *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 F is not reduced, keep trying larger μ until it does
 - If F is reduced, accept it and reduce μ for the next iteration

How is calibration used?

- Good for recovering intrinsic parameters; It is thus useful for many vision applications
- Since it requires a calibration pattern, it is often necessary to remove or replace the pattern from the footage or utilize it in some ways...

Example of calibration



(a) Background photograph



(b) Camera calibration grid and light probe

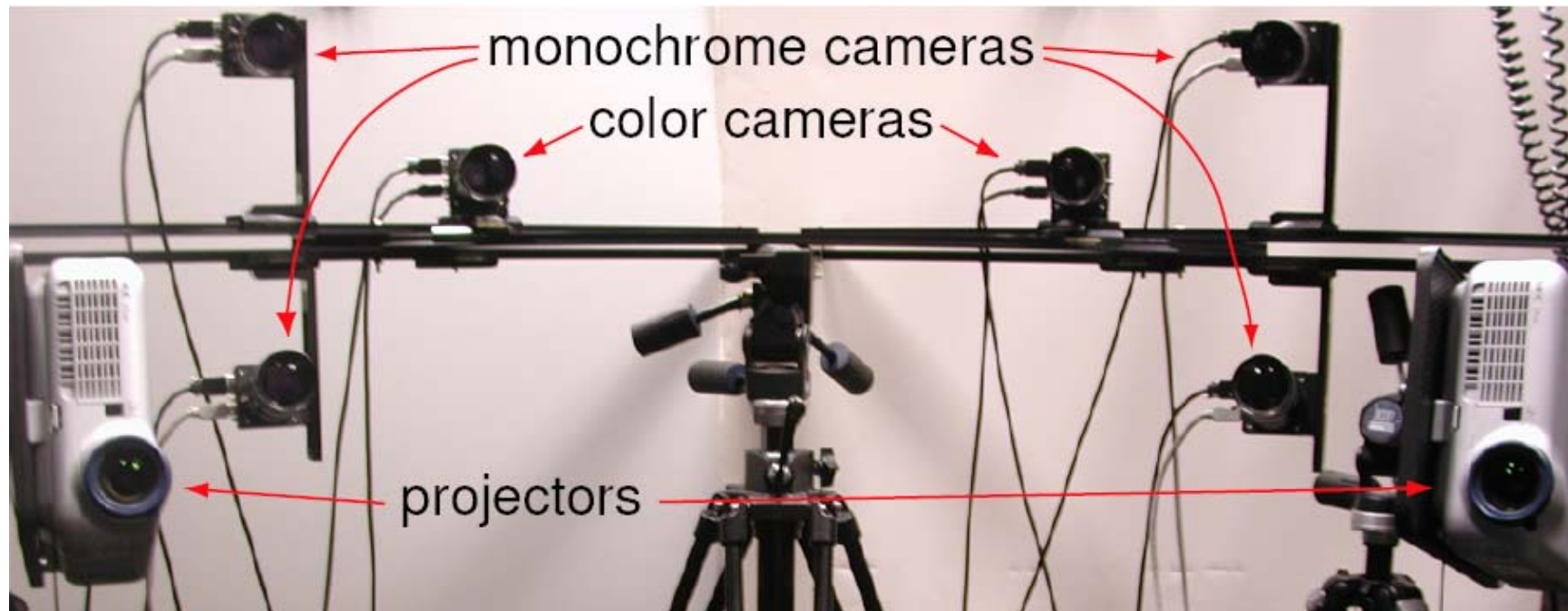


(c) Objects and local scene matched to background



(g) Final result with differential rendering

Example of calibration



Example of calibration

- Videos from GaTech