Camera calibration

Digital Visual Effects, Spring 2005

Yung-Yu Chuang

2005/4/6

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

Announcements

- Project #1 artifacts voting.
- Project #2 camera.

Outline

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

Nonlinear least square methods

Least square

Least Squares Problem

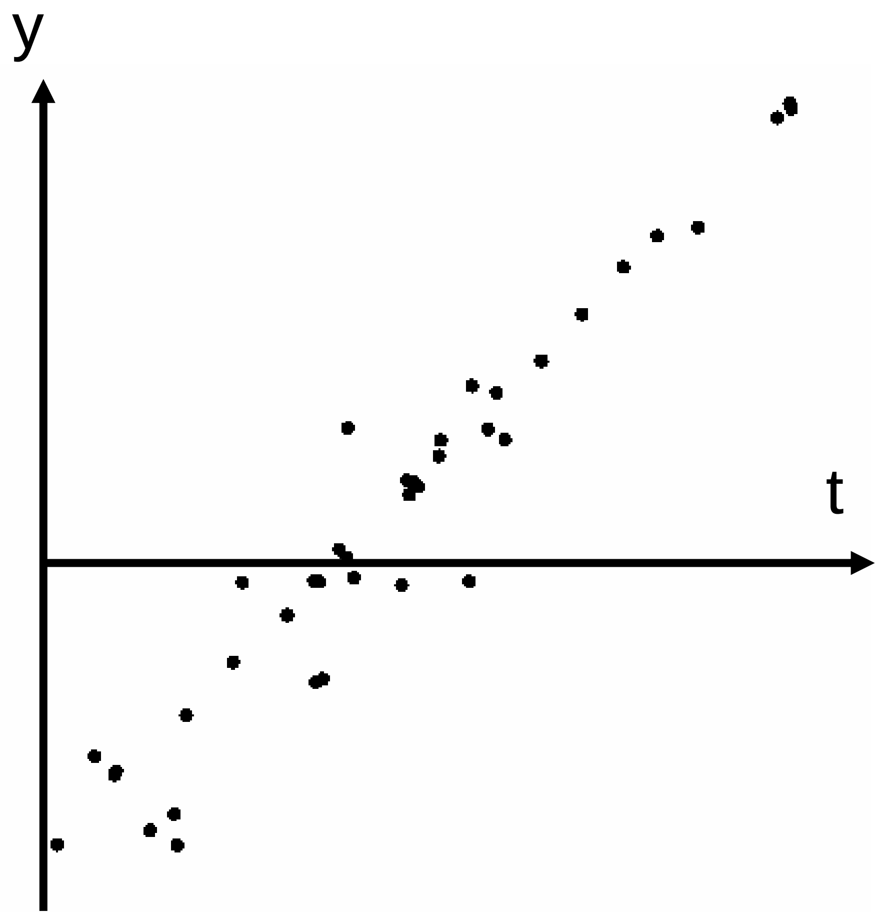
Find $\mathbf{x}^*$, a local minimizer for

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

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

It is widely seen in data fitting.

Linear least square



$$y(t) = M(x,t) = x_0 + x_1 t$$

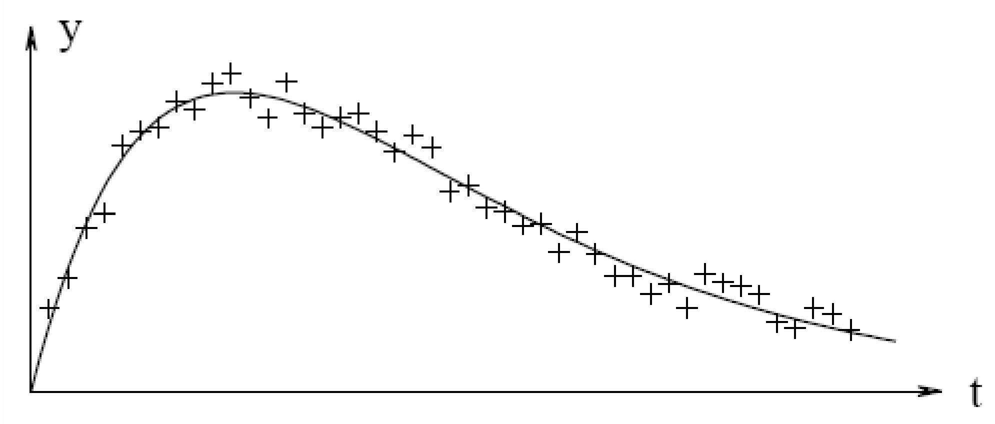
$$f_i(x) = y_i - M(x,t_i)$$

residual

prediction

$M(x,t) = x_0 + x_1 t + x_2 t^3$ is linear, too.

Nonlinear least square



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

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

$$\begin{aligned} \textit{residuals} \quad f_i(\mathbf{x}) &= y_i - M(\mathbf{x}, t_i) \\ &= y_i - x_3 e^{x_1 t_i} - x_4 e^{x_2 t_i} \end{aligned}$$

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

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

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 g is the *gradient*,

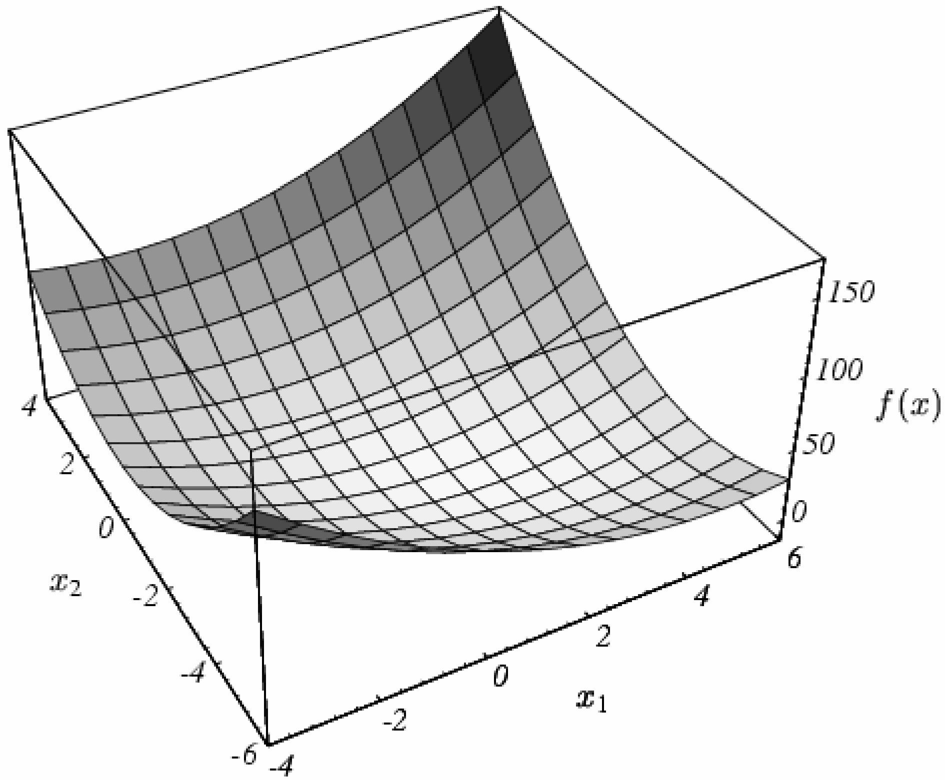
$$\mathbf{g} \equiv \mathbf{F}'(\mathbf{x}) = \begin{bmatrix} \frac{\partial F}{\partial x_1}(\mathbf{x}) \\ \vdots \\ \frac{\partial F}{\partial x_n}(\mathbf{x}) \end{bmatrix},$$

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

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

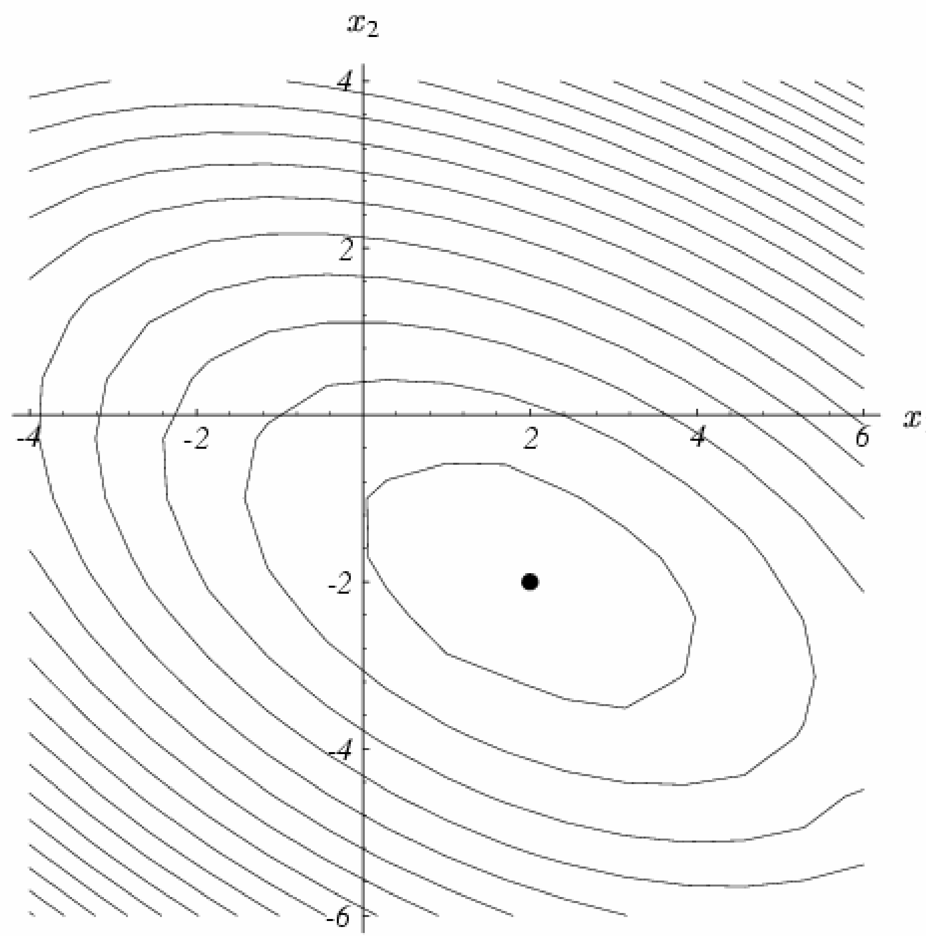
Quadratic functions

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

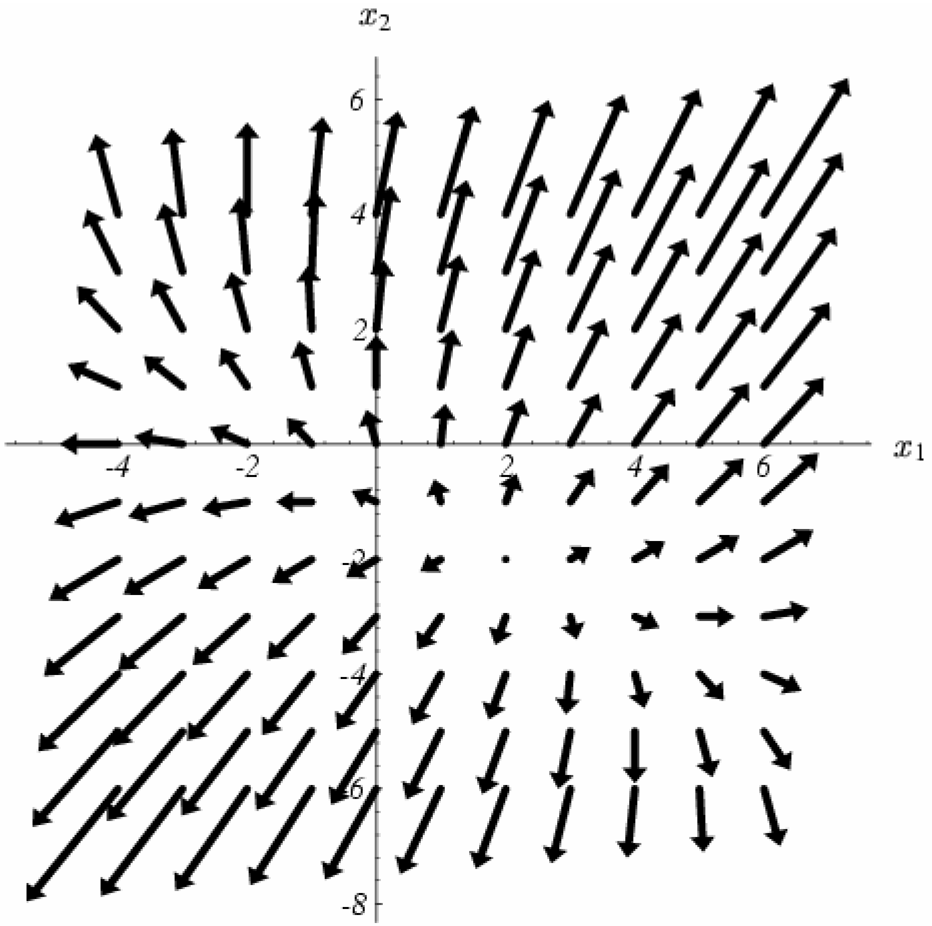


$$A = \begin{bmatrix} 3 & 2 \\ 2 & 6 \end{bmatrix}, \qquad b = \begin{bmatrix} 2 \\ -8 \end{bmatrix}, \qquad c = 0.$$

Quadratic functions

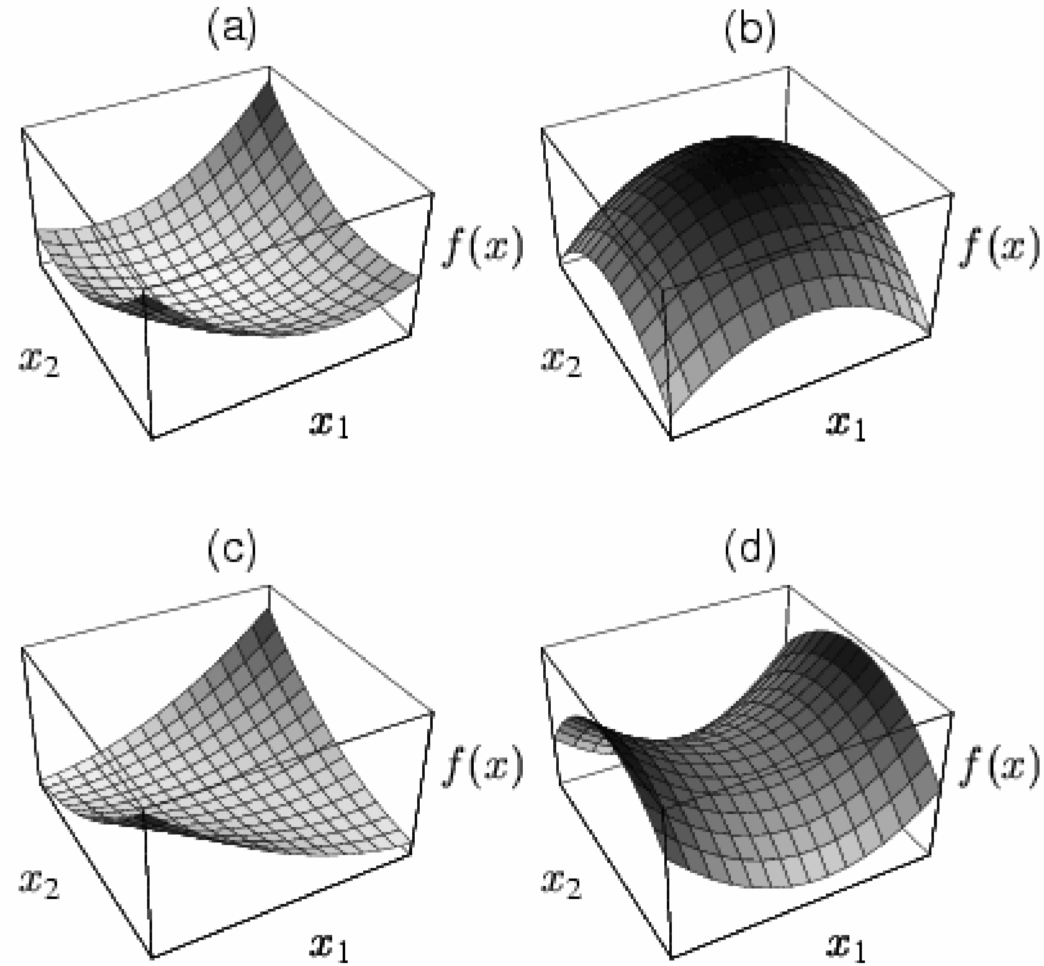


isocontour



gradient

Quadratic functions



Descent methods

1. Find a descent direction $\mathbf{h}_\mathrm{d}$
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** $(k < k_{\max})$
  $\mathbf{h}_\mathrm{d} :=$ search_direction($\mathbf{x}$) {From $\mathbf{x}$ and downhill}
  if (no such $\mathbf{h}$ exists)
   found := **true** {$\mathbf{x}$ is stationary}
  else
   $\alpha :=$ step_length($\mathbf{x}, \mathbf{h}_\mathrm{d}$) {from $\mathbf{x}$ in direction $\mathbf{h}_\mathrm{d}$}
   $\mathbf{x} := \mathbf{x} + \alpha\mathbf{h}_\mathrm{d}$; $k := k+1$ {next iterate}
end

Descent direction

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

We say that $\mathbf{h}$ is a *descent direction* if $F(\mathbf{x}+\alpha\mathbf{h})$ is a decreasing function of α at $\alpha = 0$. This leads to the following definition.

Definition Descent direction.

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

If no such $\mathbf{h}$ exists, then $\mathbf{F}'(\mathbf{x}) = \mathbf{0}$, showing that in this case $\mathbf{x}$ is stationary.

Steepest descent method

From (2.5) we see that when we perform a step $\alpha\mathbf{h}$ with positive α, then the relative gain in function value satisfies

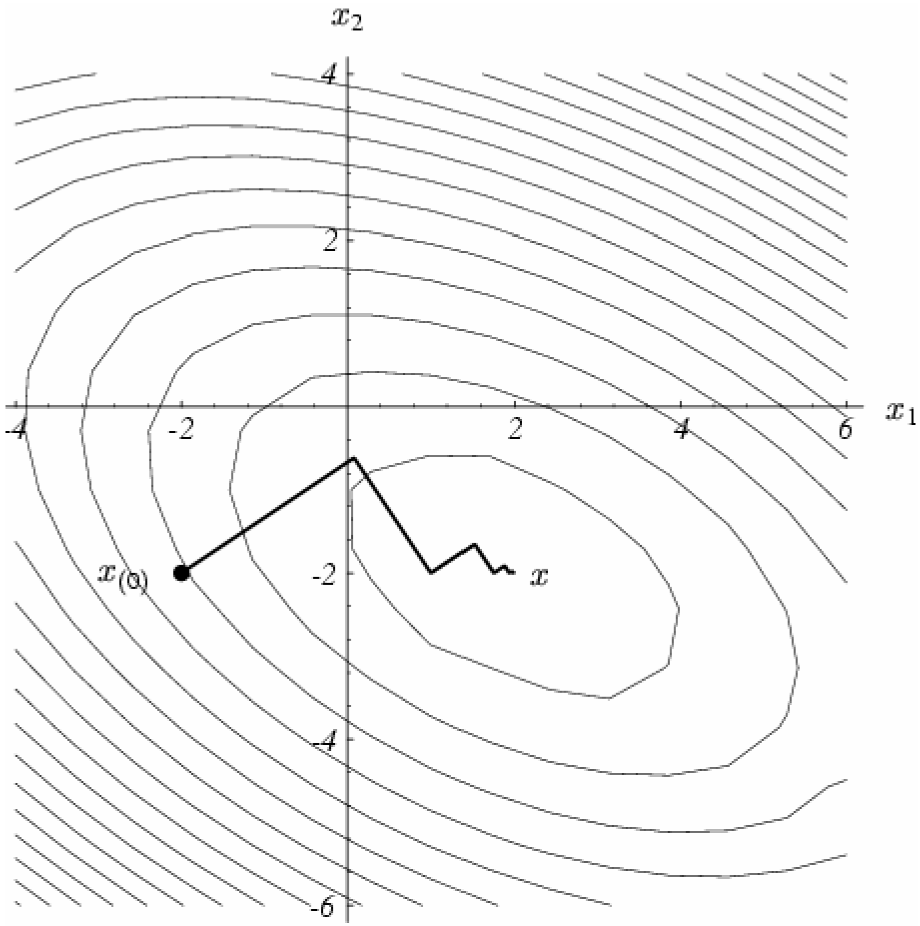
$$\lim_{\alpha\to 0} \frac{F(\mathbf{x}) - F(\mathbf{x}+\alpha\mathbf{h})}{\alpha\|\mathbf{h}\|} = -\frac{1}{\|\mathbf{h}\|}\,\mathbf{h}^\top \mathbf{F}'(\mathbf{x}) = -\|\mathbf{F}'(\mathbf{x})\|\cos\theta\ ,$$

where θ is the angle between the vectors $\mathbf{h}$ and $\mathbf{F}'(\mathbf{x})$. This shows that we get the greatest gain rate if $\theta = \pi$, ie if we use the steepest descent direction $\mathbf{h}_{\text{sd}}$ given by

$$\mathbf{h}_{\text{sd}} = -\mathbf{F}'(\mathbf{x})\ . \tag{2.8}$$

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

Steepest descent method



Newton's method

We can derive this method from the condition that $\mathbf{x}^*$ is a stationary point. According to Definition 1.6 it satisfies $\mathbf{F}'(\mathbf{x}^*) = \mathbf{0}$. This is a nonlinear system of equations, and from the Taylor expansion

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

we derive *Newton's method:* Find $\mathbf{h}_n$ as the solutions to

$$\mathbf{H}\,\mathbf{h}_n = -\mathbf{F}'(\mathbf{x}) \quad \text{with } \mathbf{H} = \mathbf{F}''(\mathbf{x})\,, \tag{2.9a}$$

Suppose that $\mathbf{H}$ is positive definite, then it is nonsingular (implying that (2.9a) has a unique solution), and $\mathbf{u}^\top \mathbf{H}\,\mathbf{u} > 0$ for all nonzero $\mathbf{u}$. Thus, by multiplying with $\mathbf{h}_n^\top$ on both sides of (2.9a) we get

$$0 < \mathbf{h}_n^\top \mathbf{H}\,\mathbf{h}_n = -\mathbf{h}_n^\top \mathbf{F}'(\mathbf{x})\,, \tag{2.10}$$

It has good performance in the final stage of the iterative process.

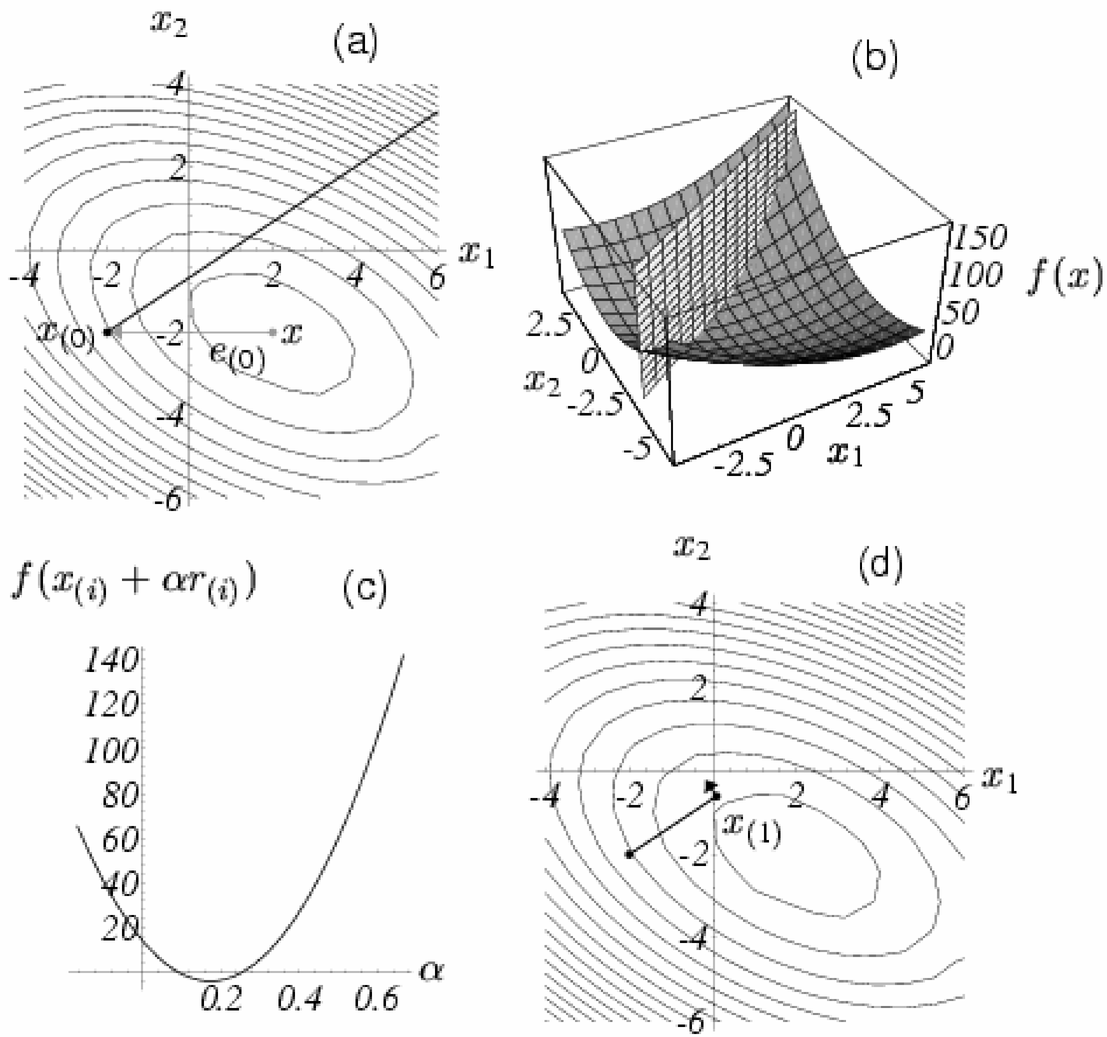
Hybrid method

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

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

Line search

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



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 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 $\varepsilon\varepsilon^T$ 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

If a covariance matrix $\mathbf{\Sigma_x}$ for the measured vector $\mathbf{x}$ is available, it can be incorporated into the LM algorithm by minimizing the squared $\mathbf{\Sigma_x}^{-1}$-norm $\epsilon^T \mathbf{\Sigma_x}^{-1} \epsilon$ instead of the Euclidean $\epsilon^T \epsilon$. Accordingly, the minimum is found by solving a weighted least squares problem defined by the *weighted normal equations*

$$\mathbf{J}^T \mathbf{\Sigma_x}^{-1} \mathbf{J} \delta_{\mathbf{p}} = \mathbf{J}^T \mathbf{\Sigma_x}^{-1} \epsilon. \tag{4}$$

Algorithm:

$k := 0$; $\nu := 2$; $\mathbf{p} := \mathbf{p}_0$;

$\mathbf{A} := \mathbf{J}^T\mathbf{J}$; $\epsilon_{\mathbf{p}} := \mathbf{x} - f(\mathbf{p})$; $\mathbf{g} := \mathbf{J}^T\epsilon_{\mathbf{p}}$;

stop:=$(||\mathbf{g}||_\infty \leq \varepsilon_1)$; $\mu := \tau * \max_{i=1,\ldots,m}(A_{ii})$;

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

    $k := k + 1$;

    repeat

        $\boxed{\text{Solve } (\mathbf{A} + \mu\mathbf{I})\delta_{\mathbf{p}} = \mathbf{g};}$

        if $(||\delta_{\mathbf{p}}|| \leq \varepsilon_2||\mathbf{p}||)$

            stop:=true;

        else

            $\mathbf{p}_{new} := \mathbf{p} + \delta_{\mathbf{p}}$;

            $\rho := (||\epsilon_{\mathbf{p}}||^2 - ||\mathbf{x} - f(\mathbf{p}_{new})||^2)/(\delta_{\mathbf{p}}^T(\mu\delta_{\mathbf{p}} + \mathbf{g}))$;

            if $\rho > 0$

                $\mathbf{p} = \mathbf{p}_{new}$;

                $\mathbf{A} := \mathbf{J}^T\mathbf{J}$; $\epsilon_{\mathbf{p}} := \mathbf{x} - f(\mathbf{p})$; $\mathbf{g} := \mathbf{J}^T\epsilon_{\mathbf{p}}$;

                stop:=$(||\mathbf{g}||_\infty \leq \varepsilon_1)$;

                $\mu := \mu * \max(\frac{1}{3}, 1 - (2\rho - 1)^3)$; $\nu := 2$;

            else

                $\mu := \mu * \nu$; $\nu := 2 * \nu$;

            endif

        endif

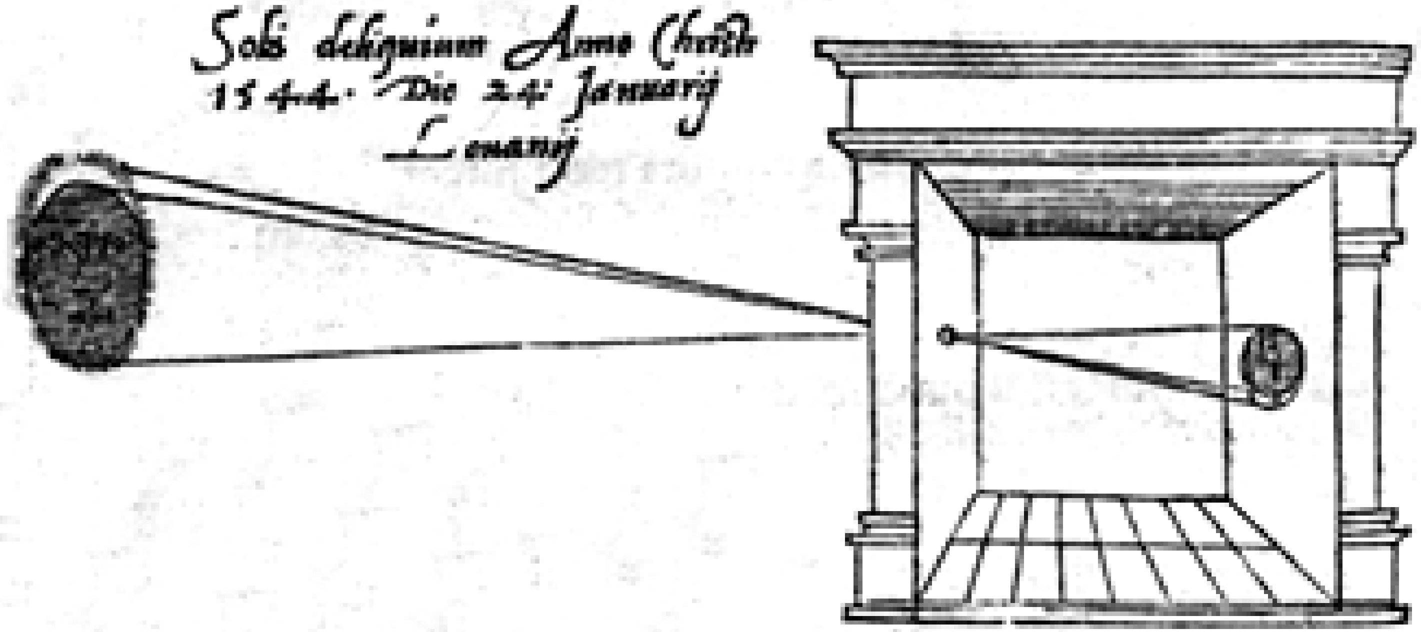
    until ($\rho > 0$) or (stop)

endwhile

Camera projection models

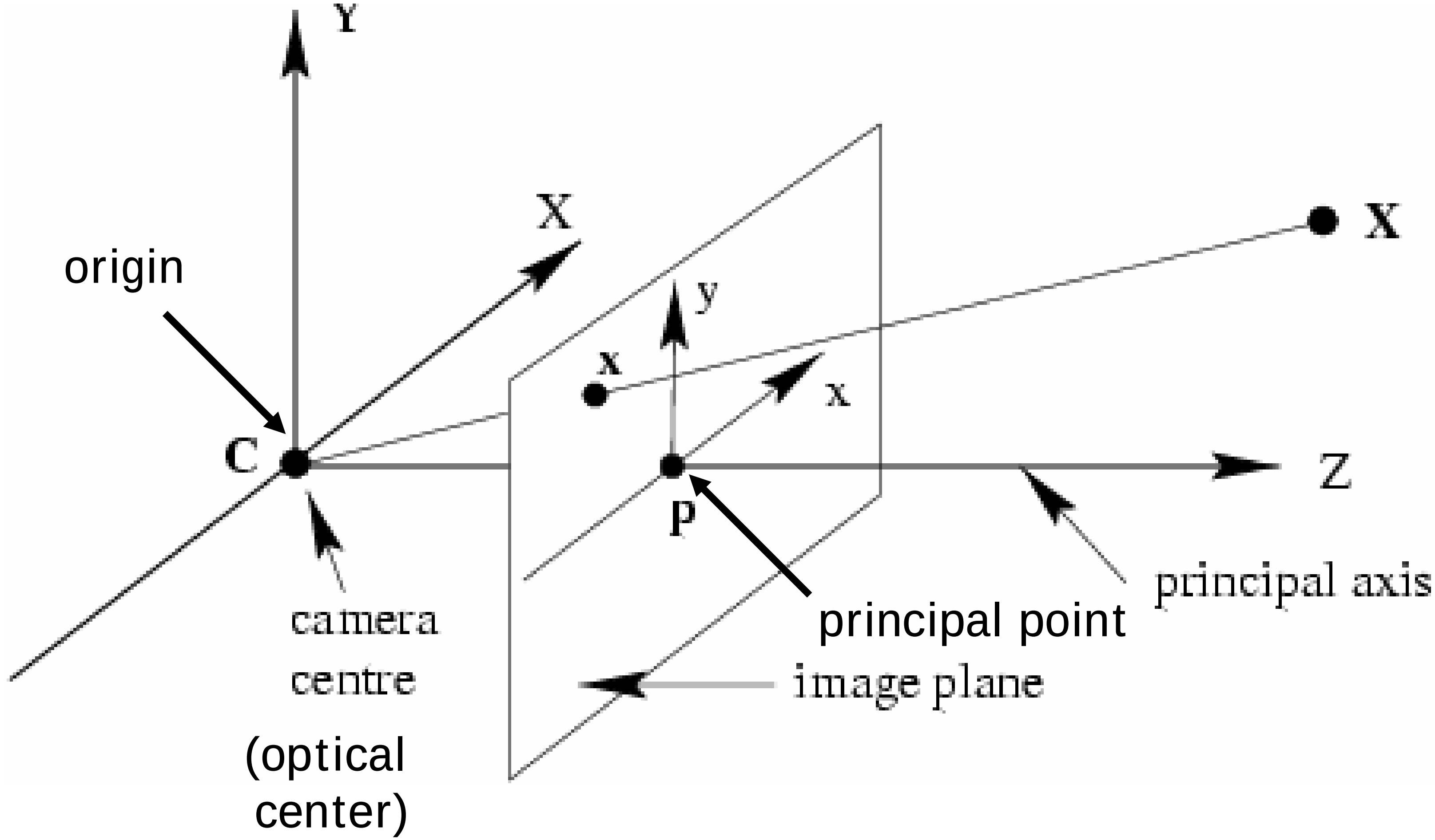
Pinhole camera

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

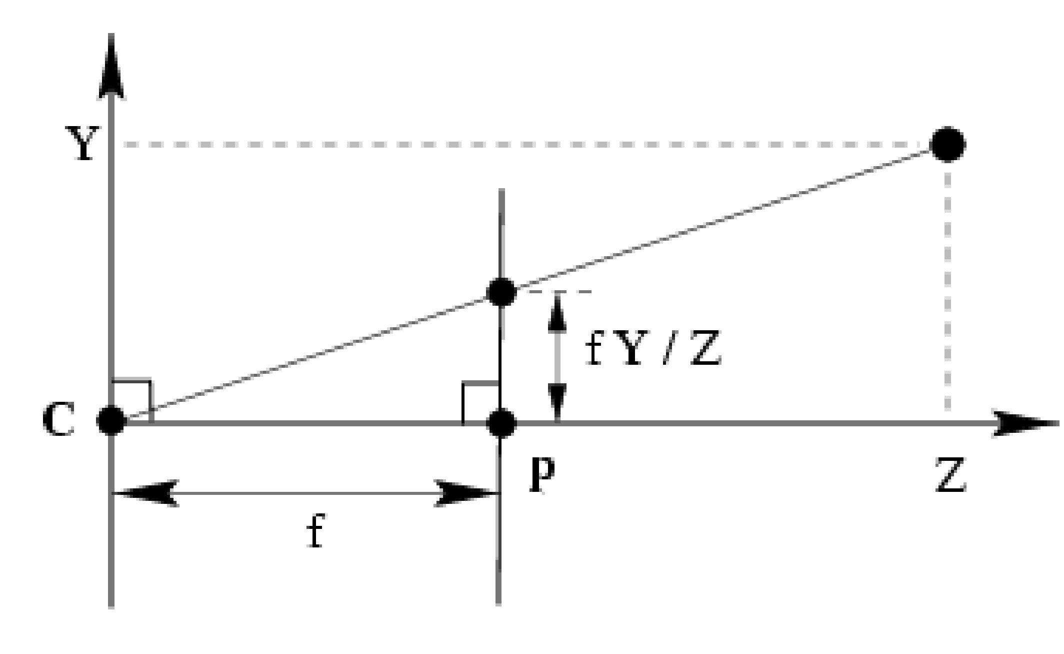


Sic nos exactè Anno .1544. Louanii eclipsim Solis
obseruauimus, inuenimusq; deficere paulò plus q̃ dex-

Pinhole camera model



Pinhole camera model

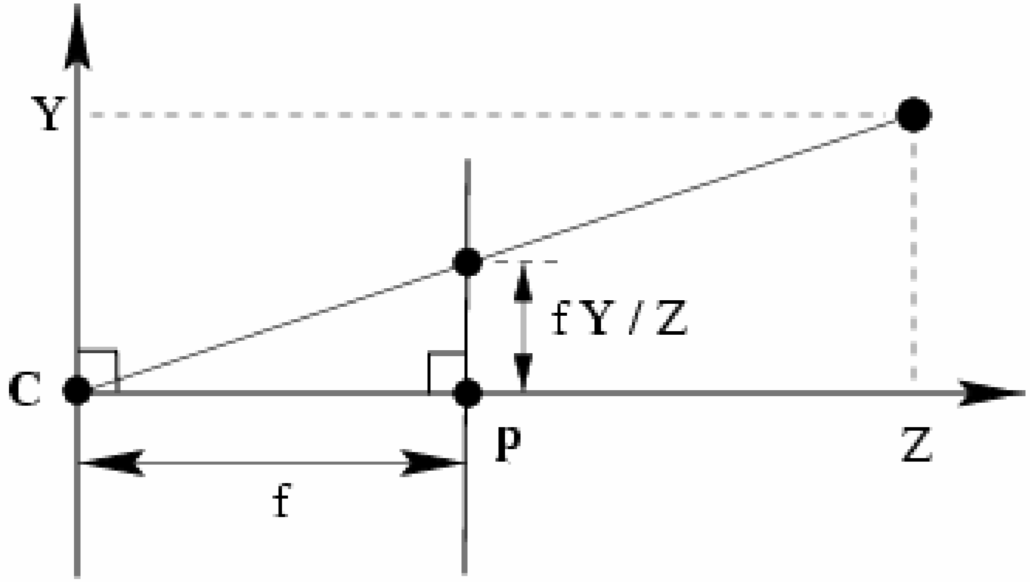


$$x = \frac{fX}{Z}$$

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

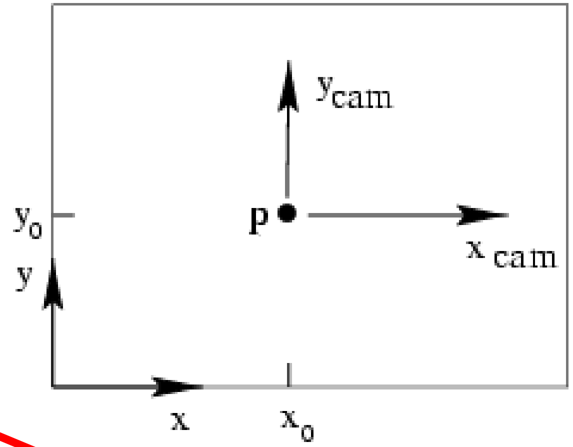
$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{pmatrix} fX \\ fY \\ Z \end{pmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

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

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

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

Intrinsic matrix

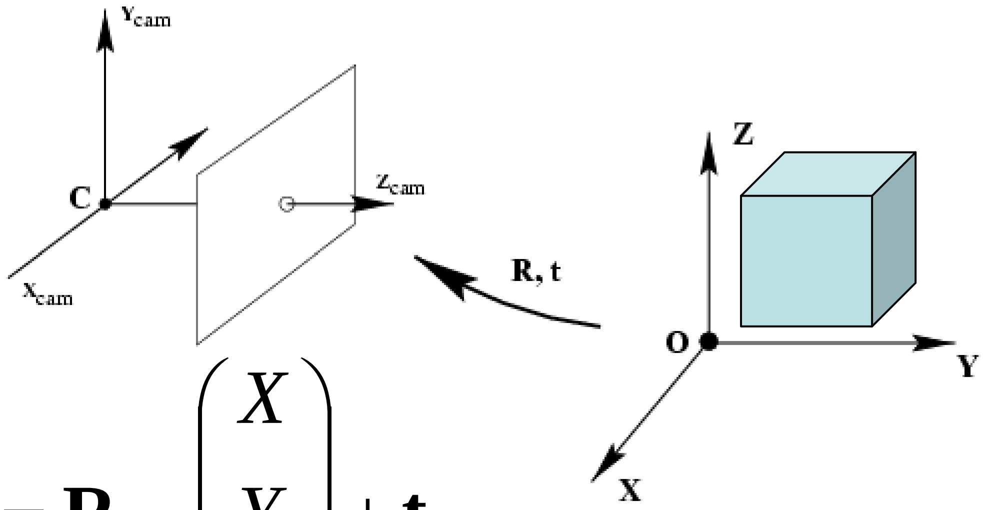
Is this form of K good enough?

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

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

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

Camera rotation and translation



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

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

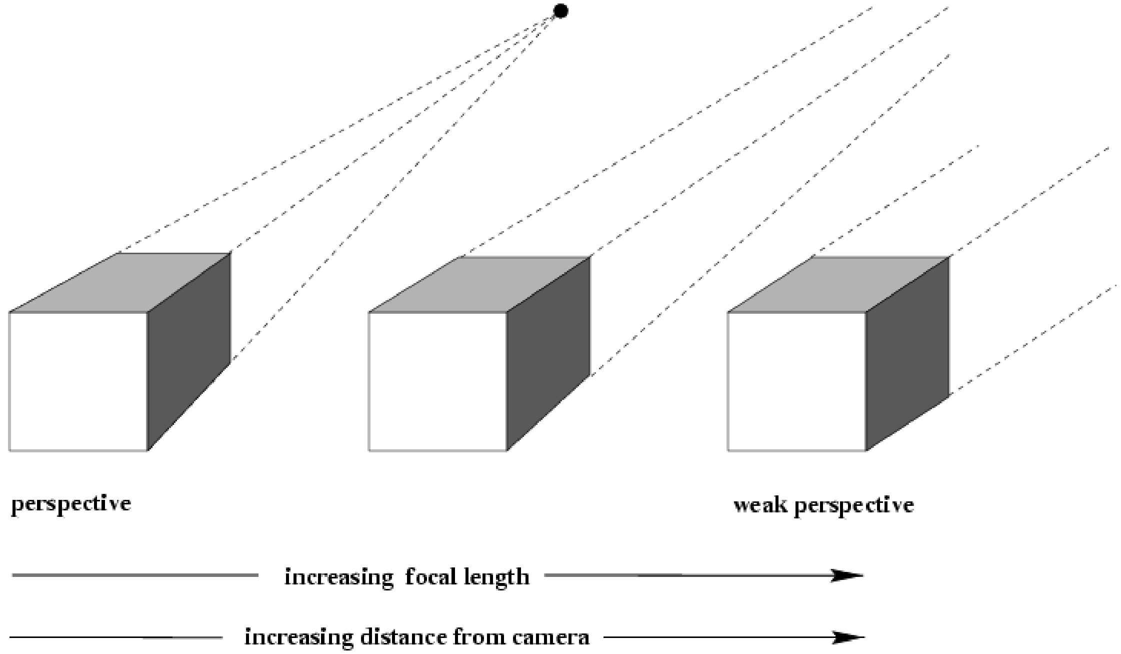
$$\mathbf{x} \sim \mathbf{K}\underbrace{[\mathbf{R}|\mathbf{t}]}\mathbf{X}$$

extrinsic matrix

Two kinds of parameters

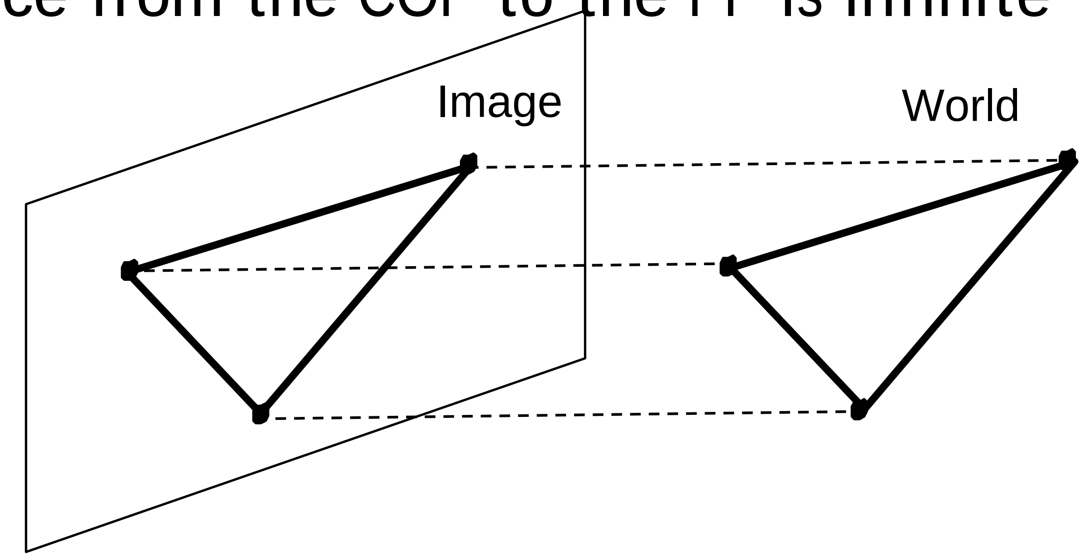
- *internal* or *intrinsic* parameters such as focal length, optical center, aspect ratio: *what kind of camera?*
- *external* or *extrinsic* (pose) parameters including rotation and translation: *where is the camera?*

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 projection

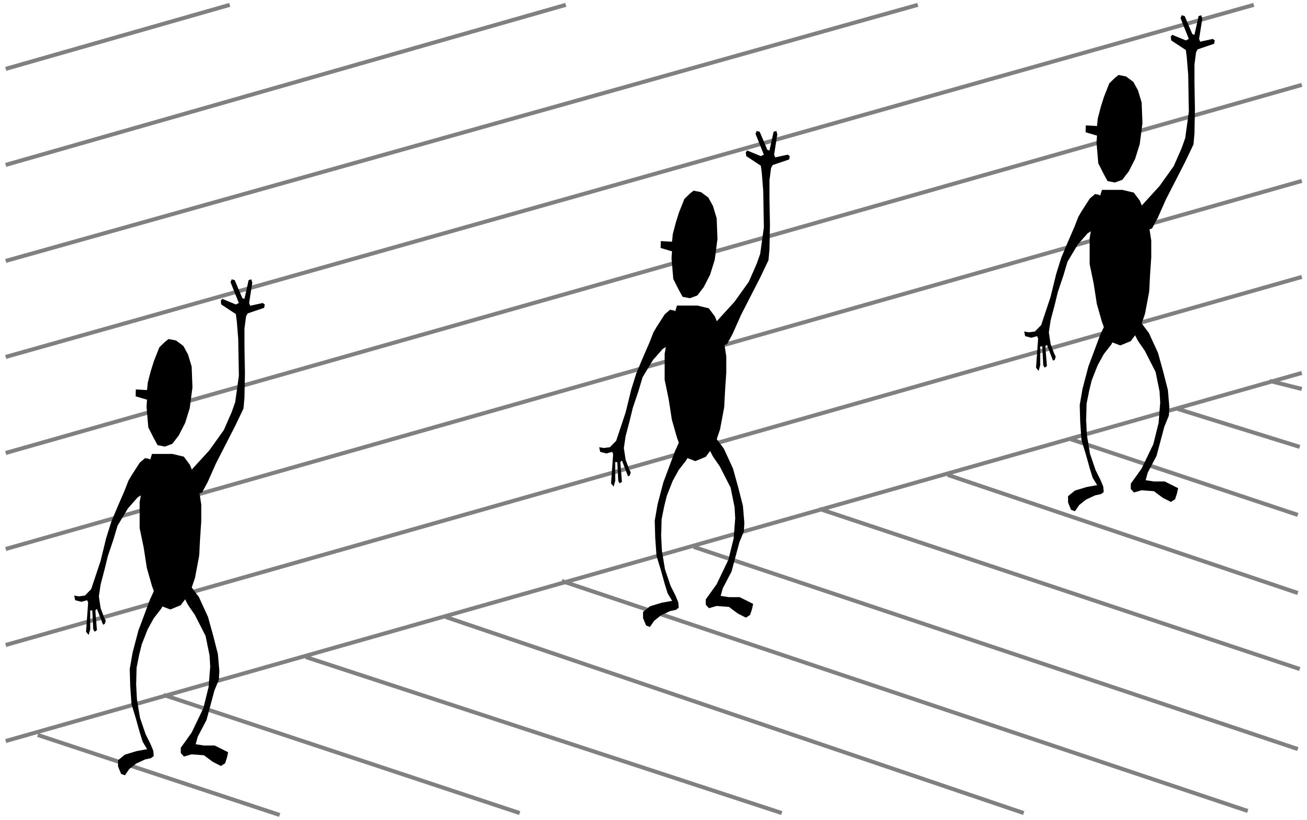
- Scaled orthographic
  - Also called “weak perspective”

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

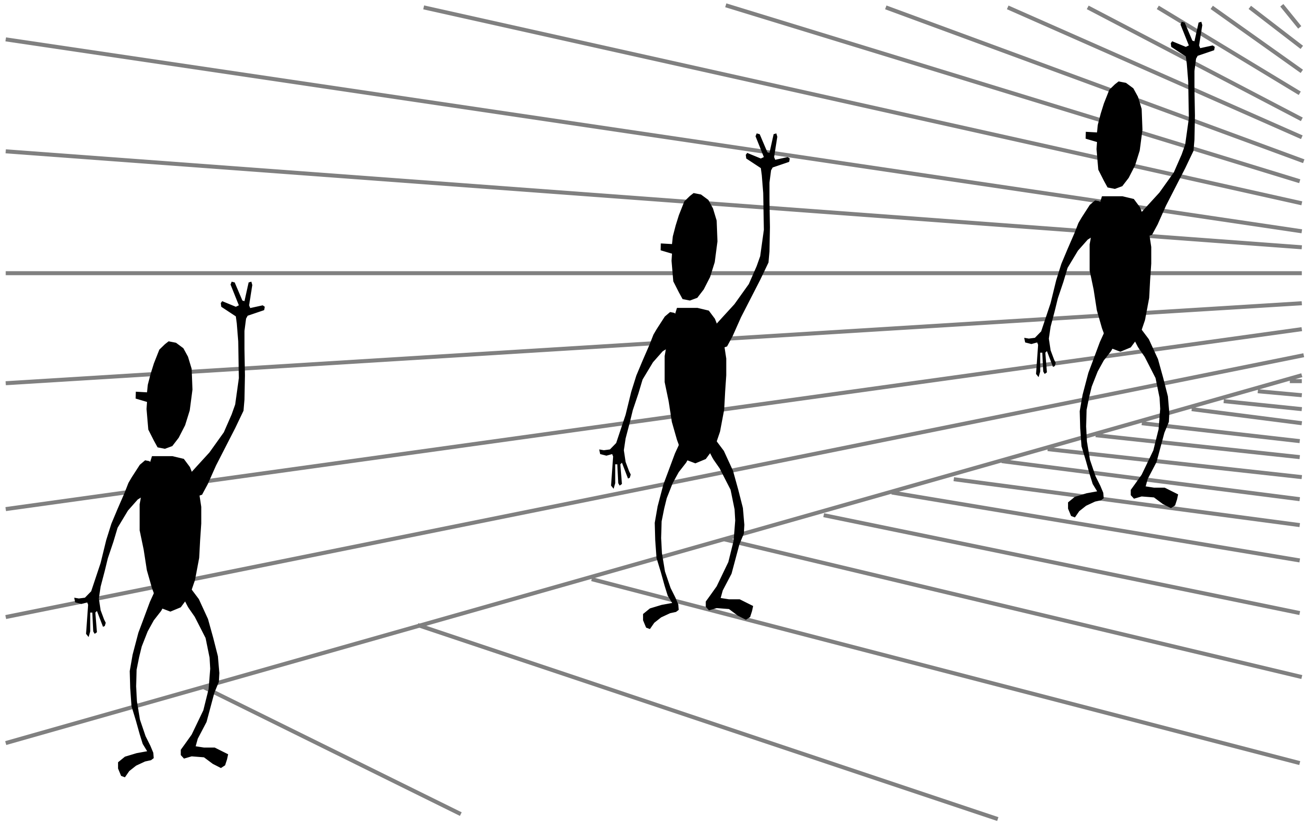
- Affine projection
  - Also called “paraperspective”

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

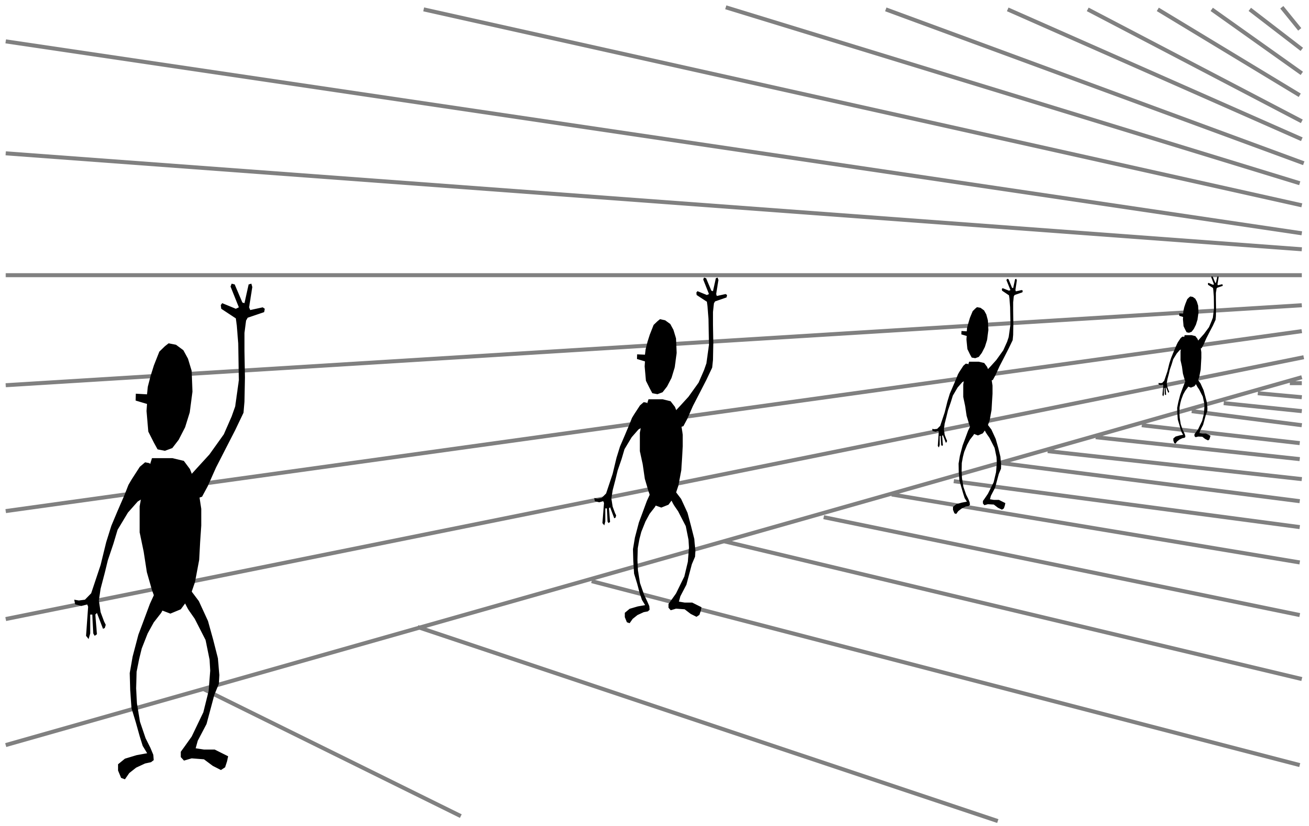
Fun with perspective



Perspective cues

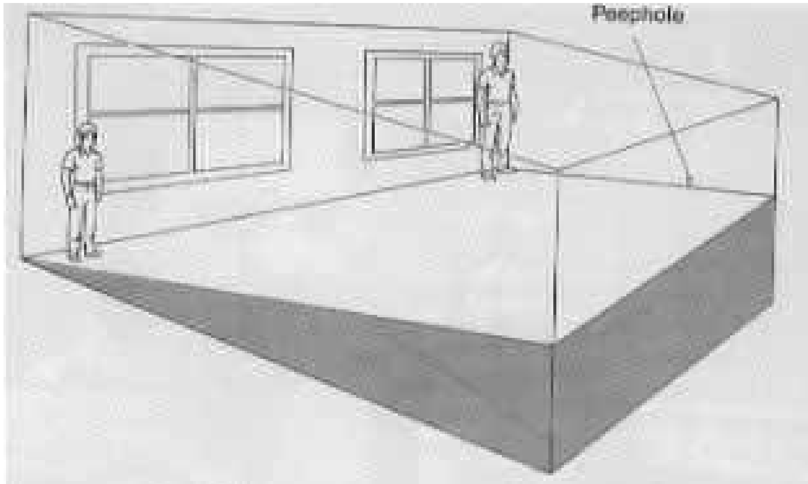


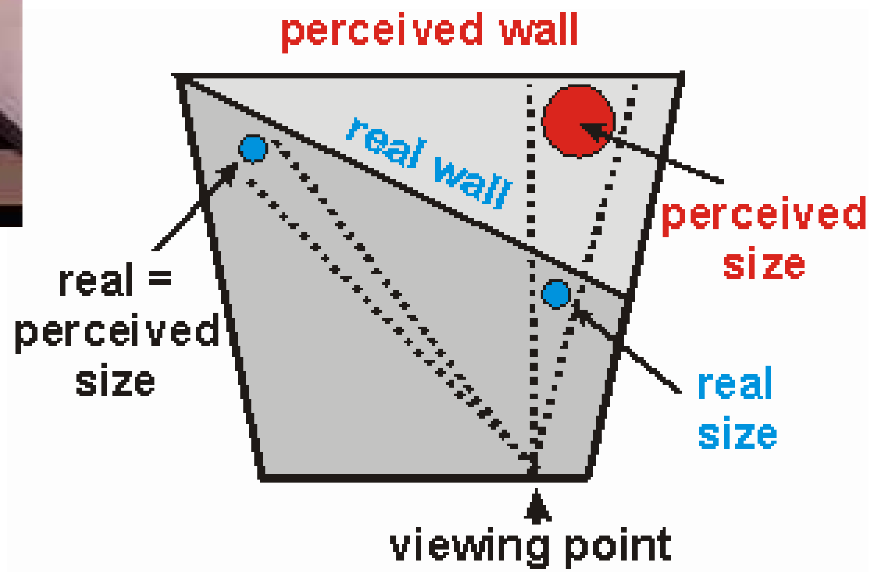
Perspective cues



Fun with perspective

Ames room





Forced perspective in LOTR

Camera calibration

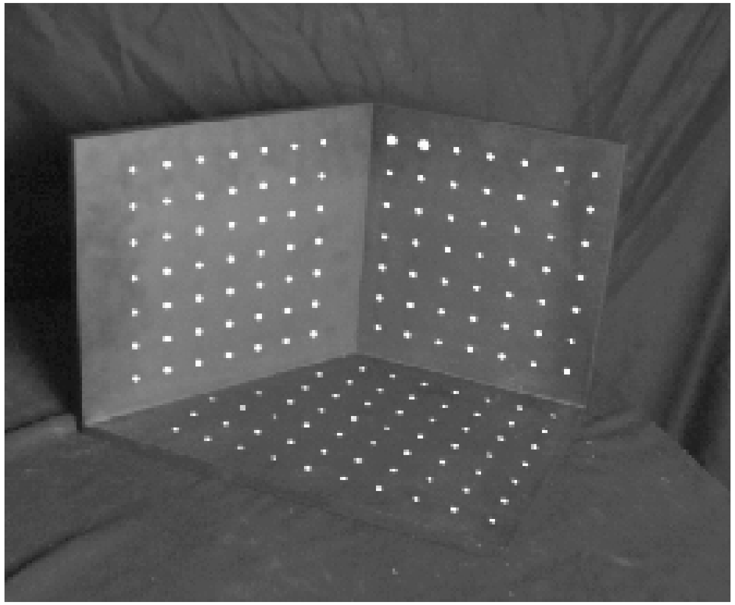
Camera calibration

- Estimate both intrinsic and extrinsic parameters
- Mainly, two categories:

1. Photometric calibration: use reference objects with known geometry
2. Self calibration: only assume static scene, e.g. structure from motion

Camera calibration approaches

1. linear regression (least squares)
2. nonlinear optinization
3. multiple planar patterns



Chromaglyphs (HP research)

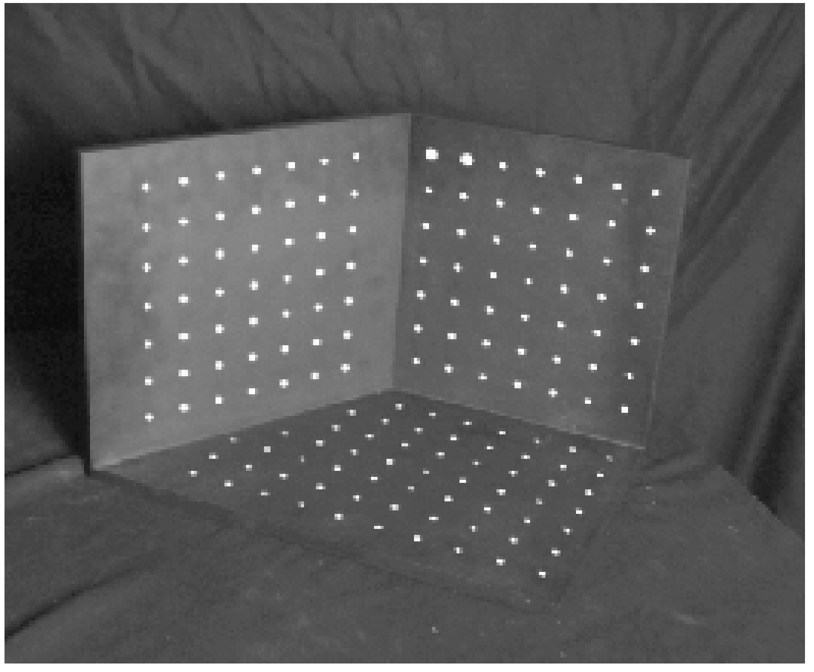
Linear regression

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

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

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

DigiVFX

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

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

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

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

Solve for Projection Matrix M using least-square techniques

Normal equation

Given an overdetermined system

$$\mathbf{A}\mathbf{x} = \mathbf{b}$$

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

$$\mathbf{A}^{\mathrm{T}}\mathbf{A}\mathbf{x} = \mathbf{A}^{\mathrm{T}}\mathbf{b}$$

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

Nonlinear optimization

- Feature measurement equations

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

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

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

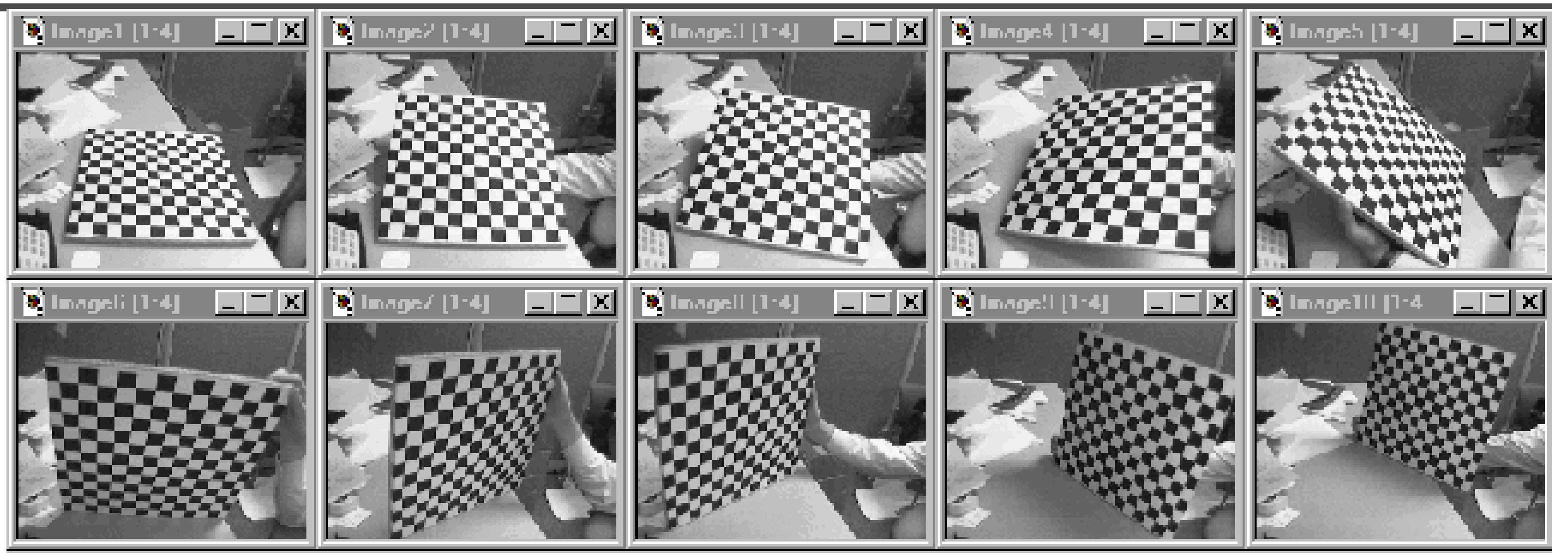
Optimal estimation

- Log likelihood of *M* given $\{(u_i, v_i)\}$

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

- How do we minimize *C*?
- Non-linear regression (least squares), because $\hat{u}_i$ and v_i are non-linear functions of *M*
- We can use Levenberg-Marquardt method to minimize it

Multi-plane calibration

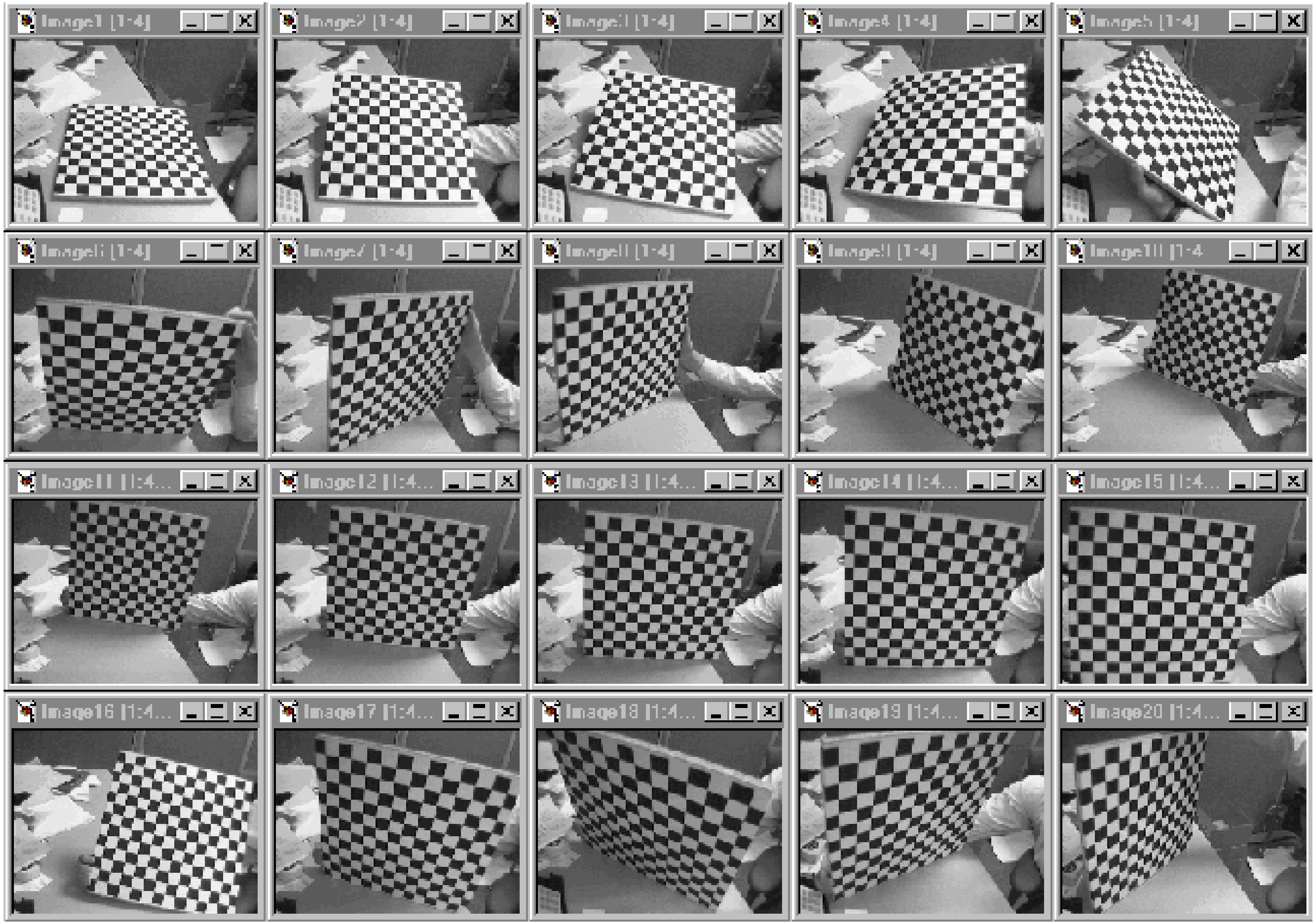


Images courtesy Jean-Yves Bouguet, Intel Corp.

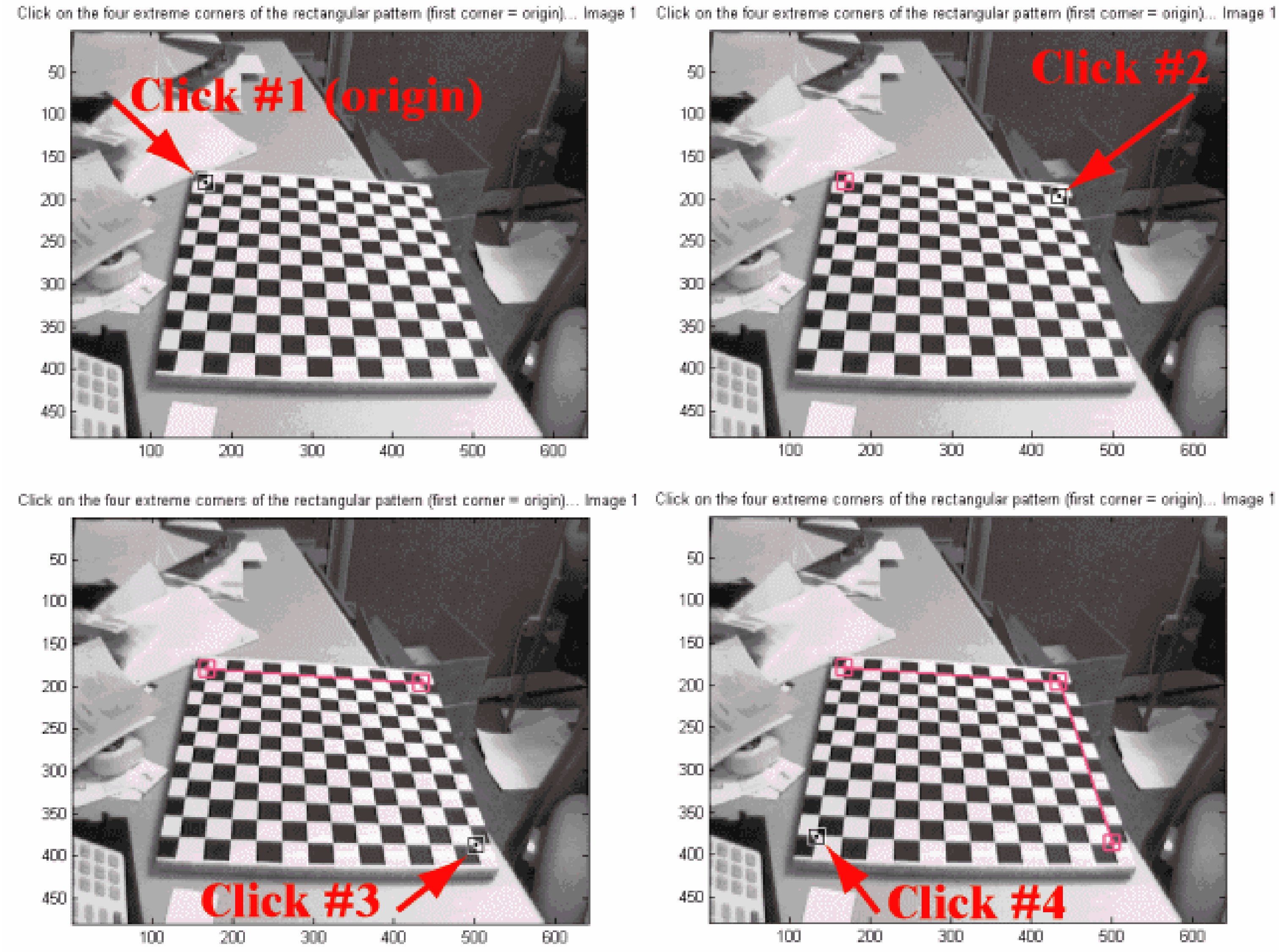
Advantage

- 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 Bouget: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html
  - Zhengyou Zhang’s web site: http://research.microsoft.com/~zhang/Calib/

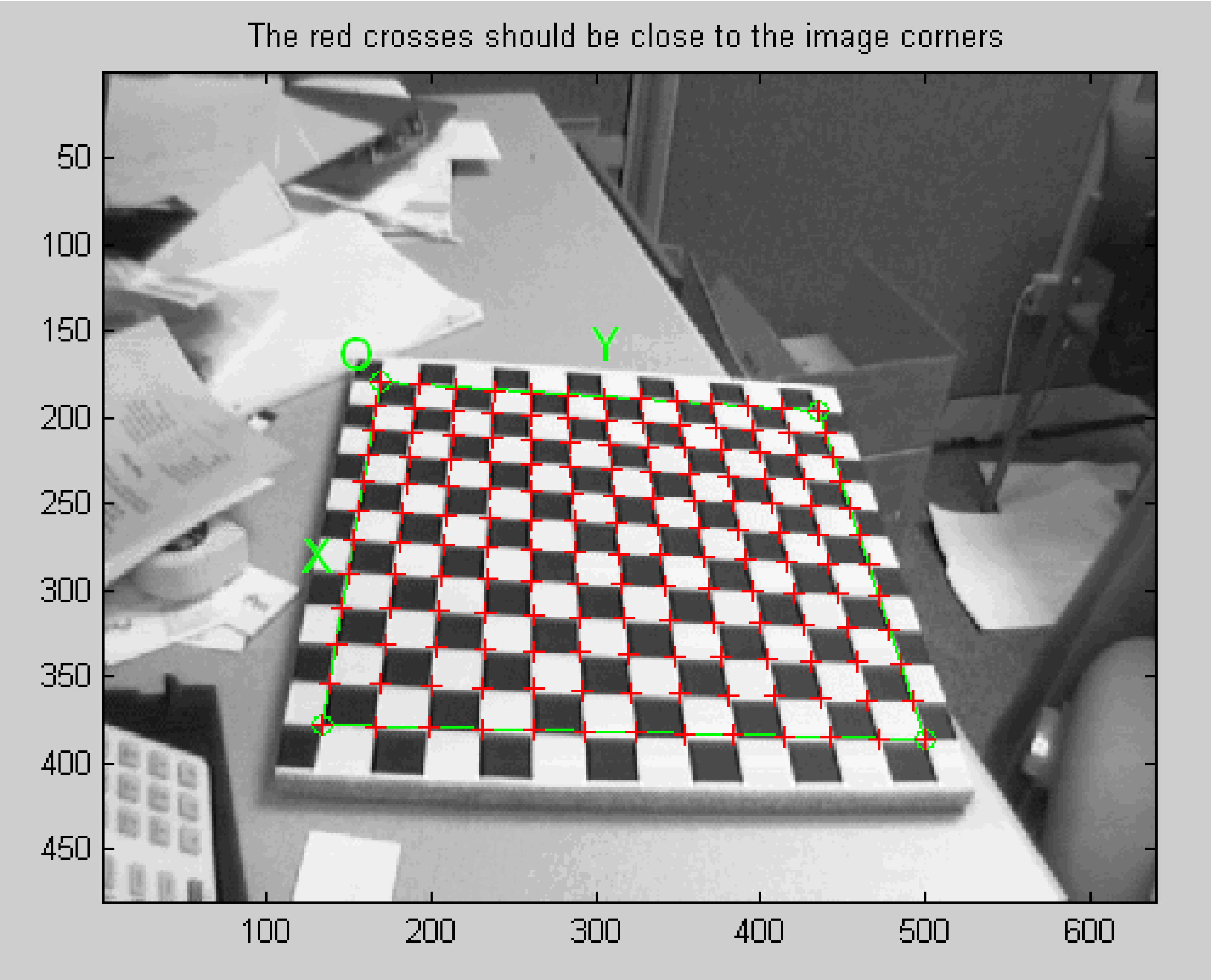
Step 1: data acquisition



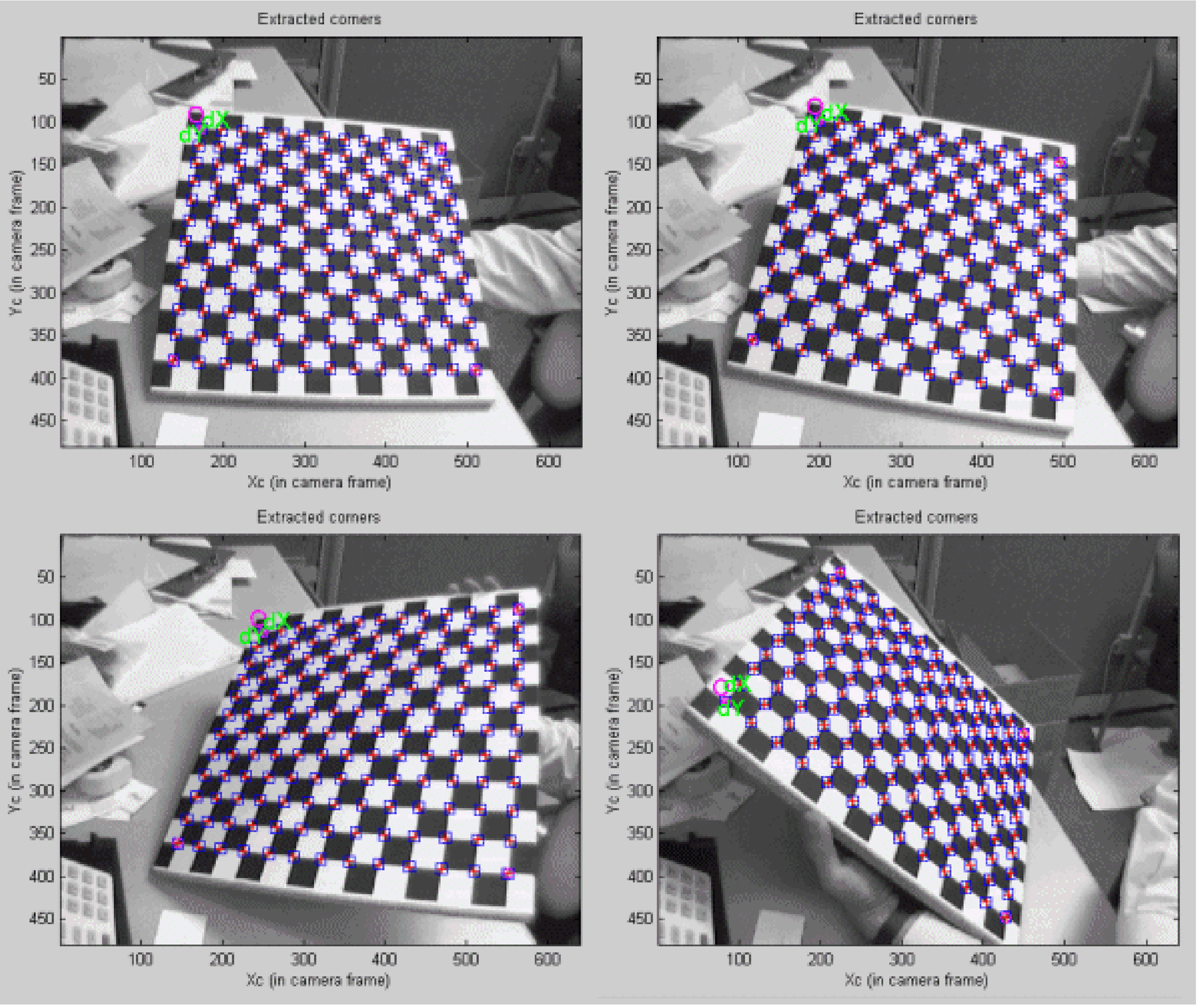
Step 2: specify corner order



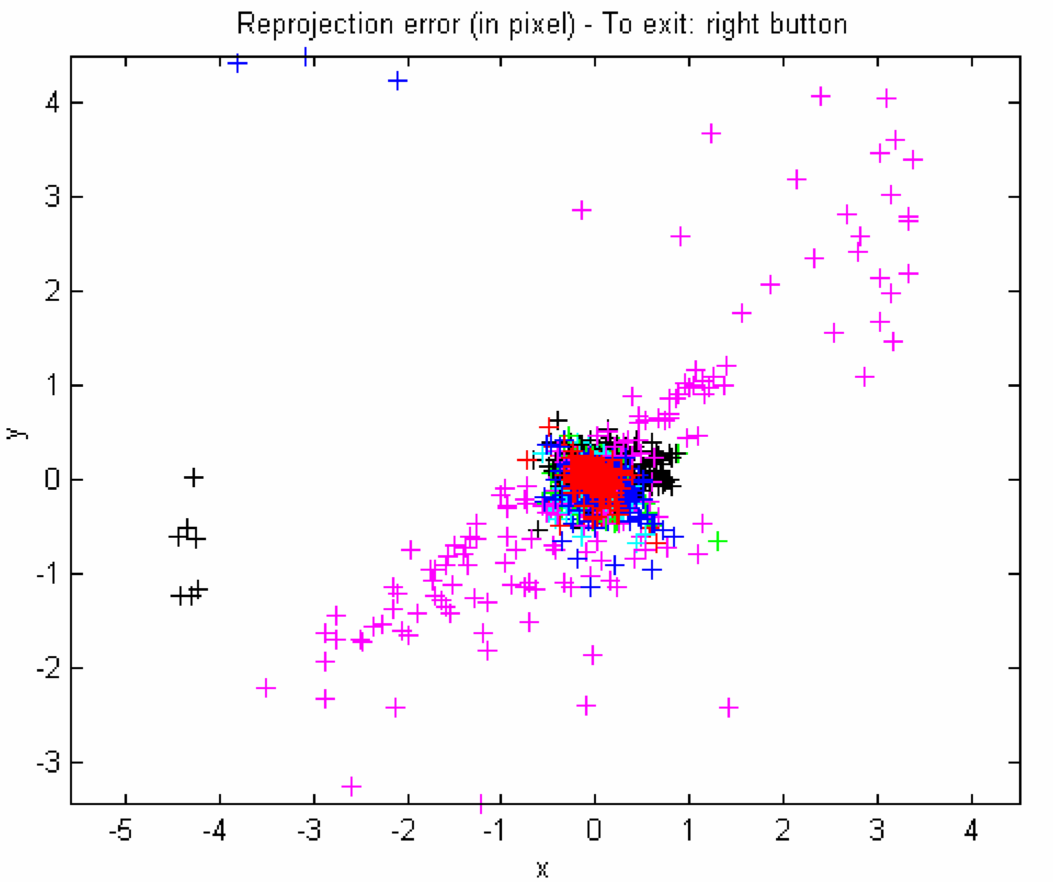
Step 3: corner extraction



Step 3: corner extraction



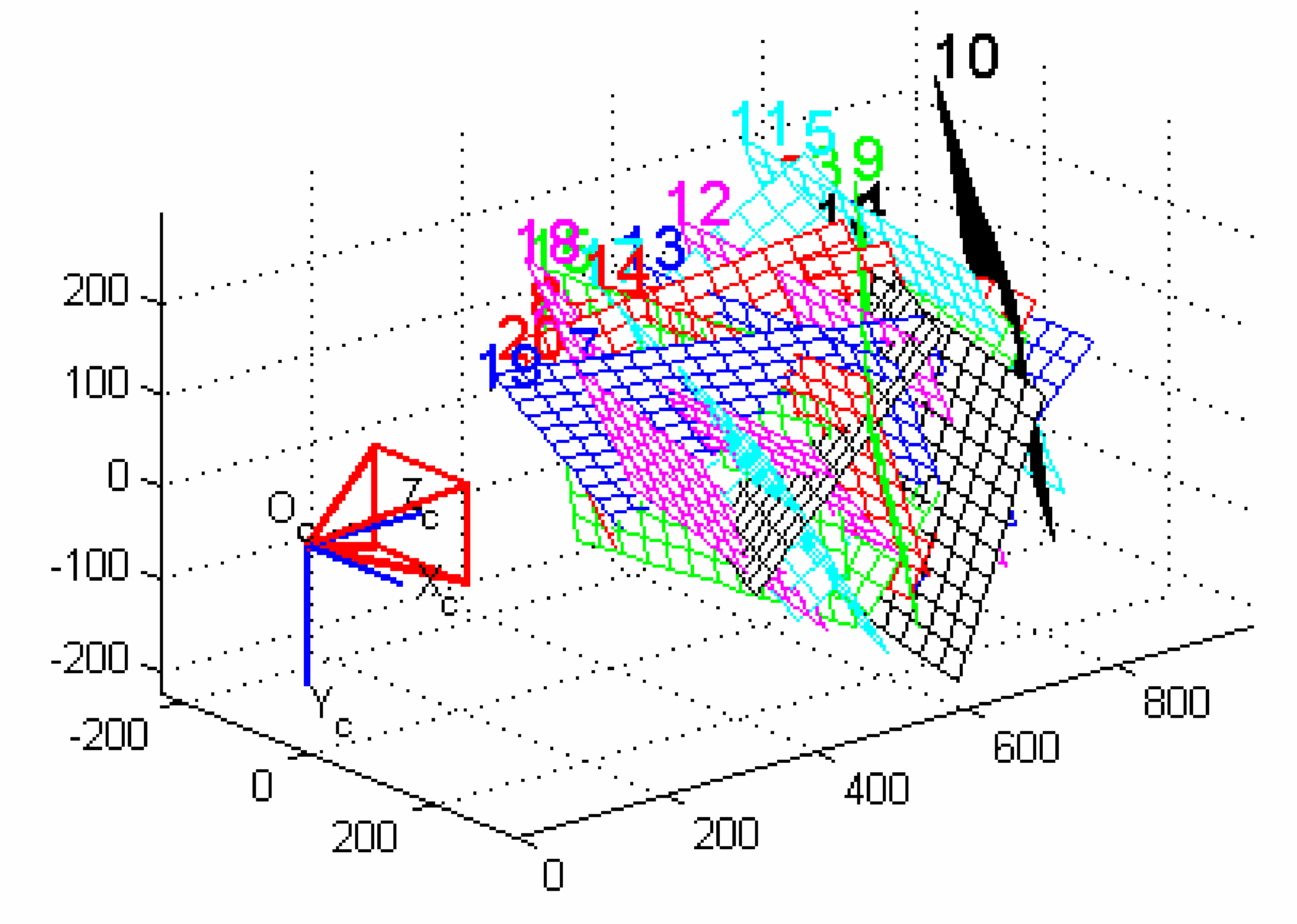
Step 4: minimize projection error



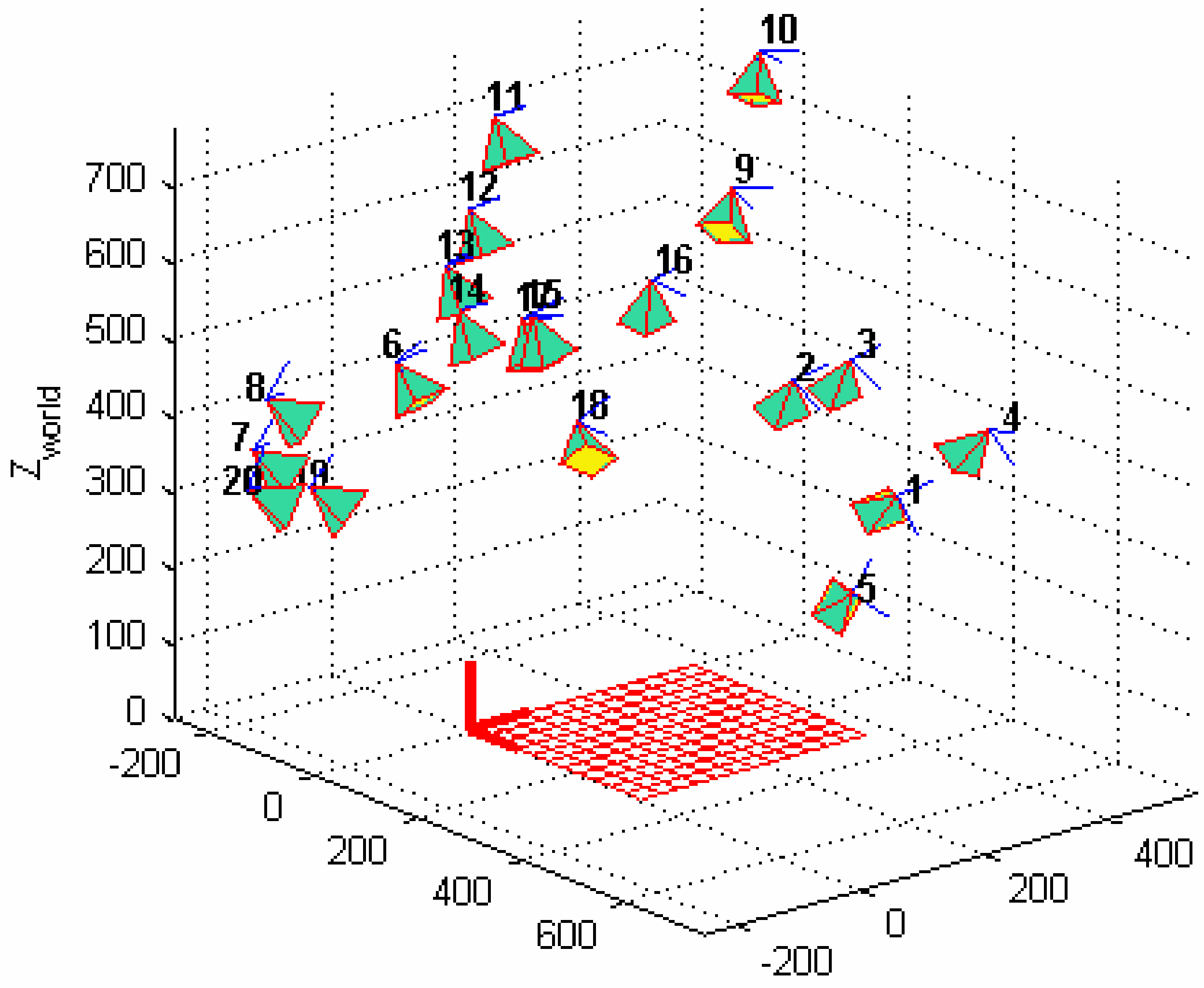

```
Calibration res

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 ]
```

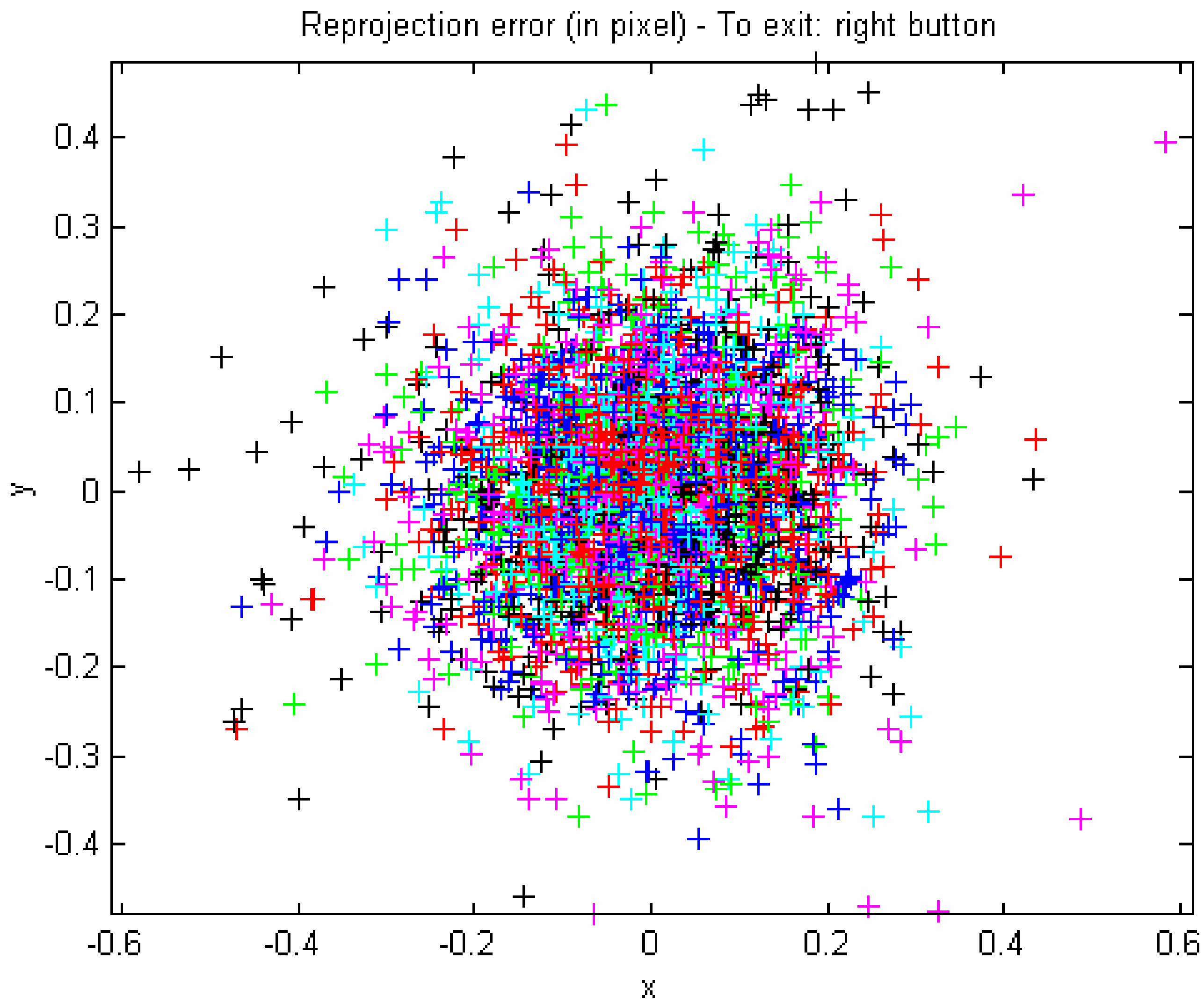
Step 4: camera calibration



Step 4: camera calibration



Step 5: refinement



Bundle adjustment

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

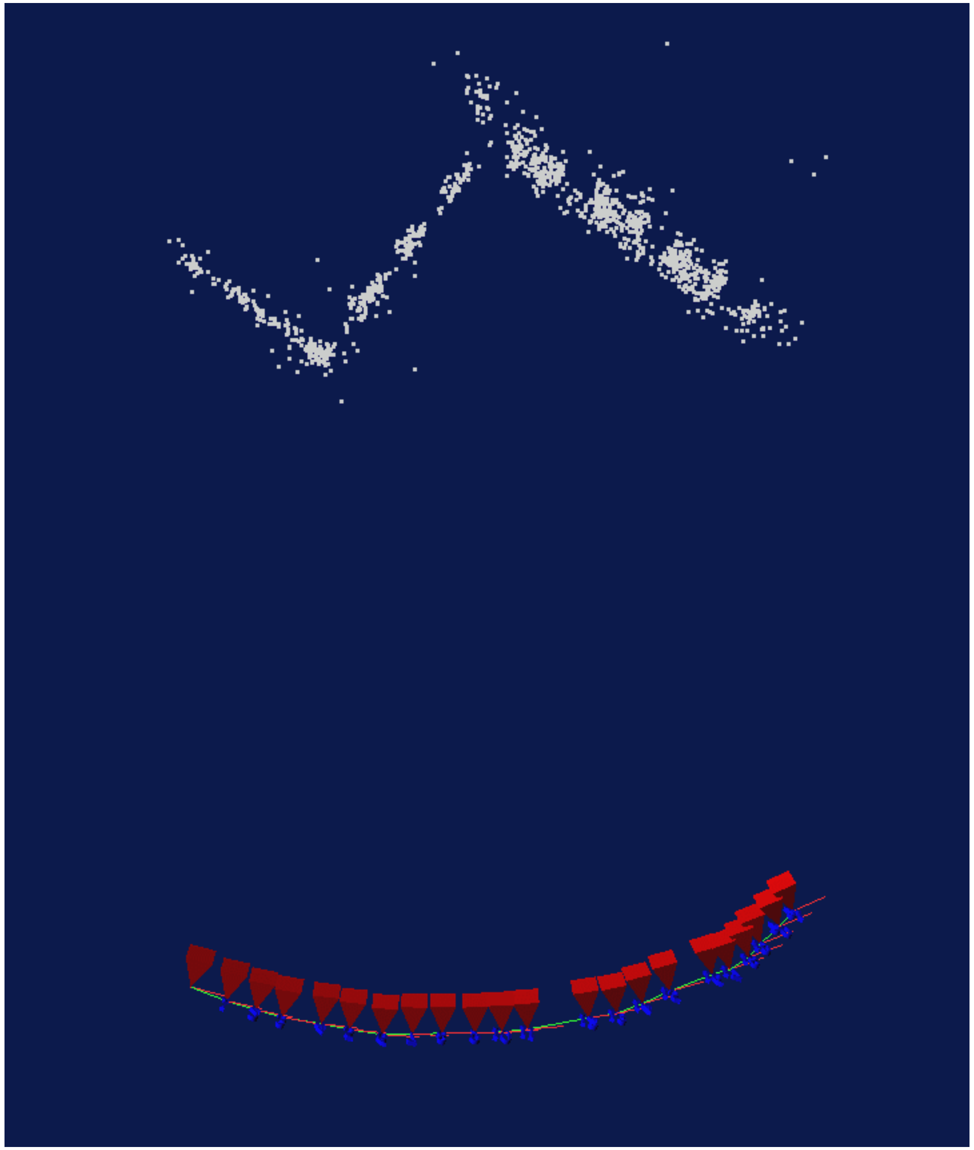
- n 3D points are seen in m views
- x_{ij} is the projection of the i-th point on image j
- a_j is the parameters for the j-th camera
- b_i is the parameters for the i-th point
- BA attempts to minimize the projection error

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

predicted projection

Euclidean distance

Bundle adjustment



Algorithm:

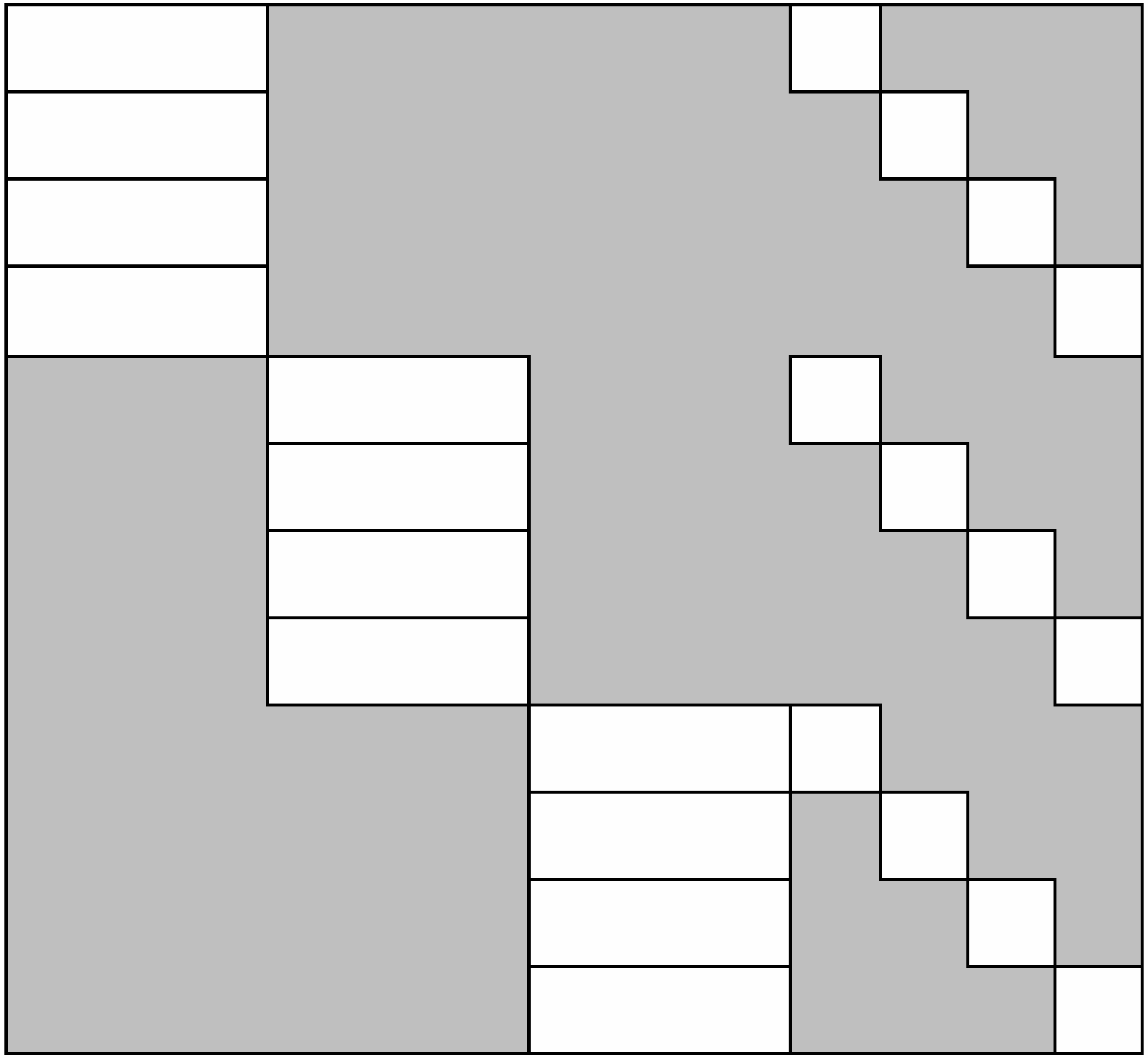
```
k := 0; ν := 2; p := p_0;
A := J^T J; ε_p := x − f(p); g := J^T ε_p;
stop:=(||g||_∞ ≤ ε_1); μ := τ * max_{i=1,...,m}(A_ii);
while (not stop) and (k < k_max)
    k := k + 1;
    repeat
        Solve (A + μI)δ_p = g;
        if (||δ_p|| ≤ ε_2||p||)
            stop:=true;
        else
            p_new := p + δ_p;
            ρ := (||ε_p||^2 − ||x − f(p_new)||^2)/(δ_p^T(μδ_p + g));
            if ρ > 0
                p = p_new;
                A := J^T J; ε_p := x − f(p); g := J^T ε_p;
                stop:=(||g||_∞ ≤ ε_1);
                μ := μ * max(1/3, 1 − (2ρ − 1)^3); ν := 2;
            else
                μ := μ * ν; ν := 2 * ν;
            endif
        endif
    until (ρ > 0) or (stop)
endwhile
```

Bundle adjustment

3 views and 4 points

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

Typical Jacobian



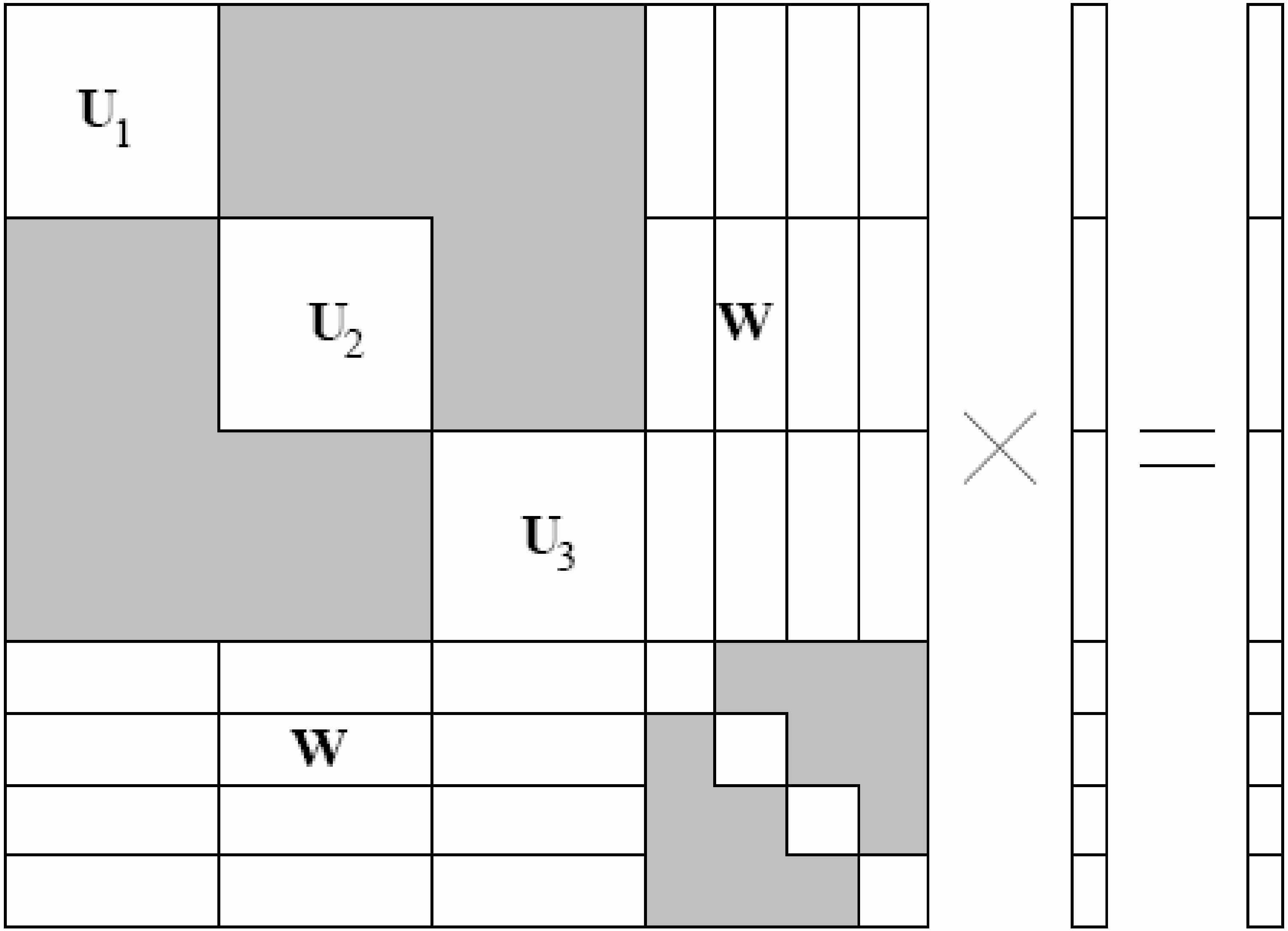
Bundle adjustment

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

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

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

Block structure of normal equation



Bundle adjustment

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

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

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

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

Recognising panoramas

- Parameterise each camera by rotation and focal length

$$\mathbf{R}_i = e^{[\boldsymbol{\theta}_i]_\times}, \ [\boldsymbol{\theta}_i]_\times = \begin{bmatrix} 0 & -\theta_{i3} & \theta_{i2} \\ \theta_{i3} & 0 & -\theta_{i1} \\ -\theta_{i2} & \theta_{i1} & 0 \end{bmatrix}$$

$$\mathbf{K}_i = \begin{bmatrix} f_i & 0 & 0 \\ 0 & f_i & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- This gives pairwise homographies

$$\tilde{\mathbf{u}}_i = \mathbf{H}_{ij}\tilde{\mathbf{u}}_j, \ \mathbf{H}_{ij} = \mathbf{K}_i\mathbf{R}_i\mathbf{R}_j^T\mathbf{K}_j^{-1}$$

Error function

- Sum of squared projection errors

$$e = \sum_{i=1}^{n} \sum_{j \epsilon \mathcal{I}(i)} \sum_{k \epsilon \mathcal{F}(i,j)} f(\mathbf{r}_{ij}^{k})^2$$

  - n = #images
  - I(i) = set of image matches to image i
  - F(i, j) = set of feature matches between images i,j
  - r_{ij}^{k} = residual of k^{th} feature match between images i,j

- Robust error function

$$f(\mathbf{x}) = \begin{cases} |\mathbf{x}|, & \text{if } |\mathbf{x}| < x_{max} \\ x_{max}, & \text{if } |\mathbf{x}| \geq x_{max} \end{cases}$$

A sparse BA software using LM

- **sba** is a generic C implementation for bundle adjustment using Levenberg-Marquardt method. It is available at http://www.ics.forth.gr/~lourakis/sba.
- You can use this library for your project #2.

MatchMove

Reference

- Manolis Lourakis and Antonis Argyros, The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm, FORTH-ICS/TR-320 2004.
- K. Madsen, H.B. Nielsen, O. Timgleff, Methods for Non-Linear Least Squares Problems, 2004.
- Zhengyou Zhang, A Flexible New Techniques for Camera Calibration, MSR-TR-98-71, 1998.
- Bill Triggs, Philip McLauchlan, Richard Hartley and Andrew Fitzgibbon, Bundle Adjustment - A Modern Symthesis, Proceedings of the International Workshop on Vision Algorithms: Theory and Practice, pp298-372, 1999.