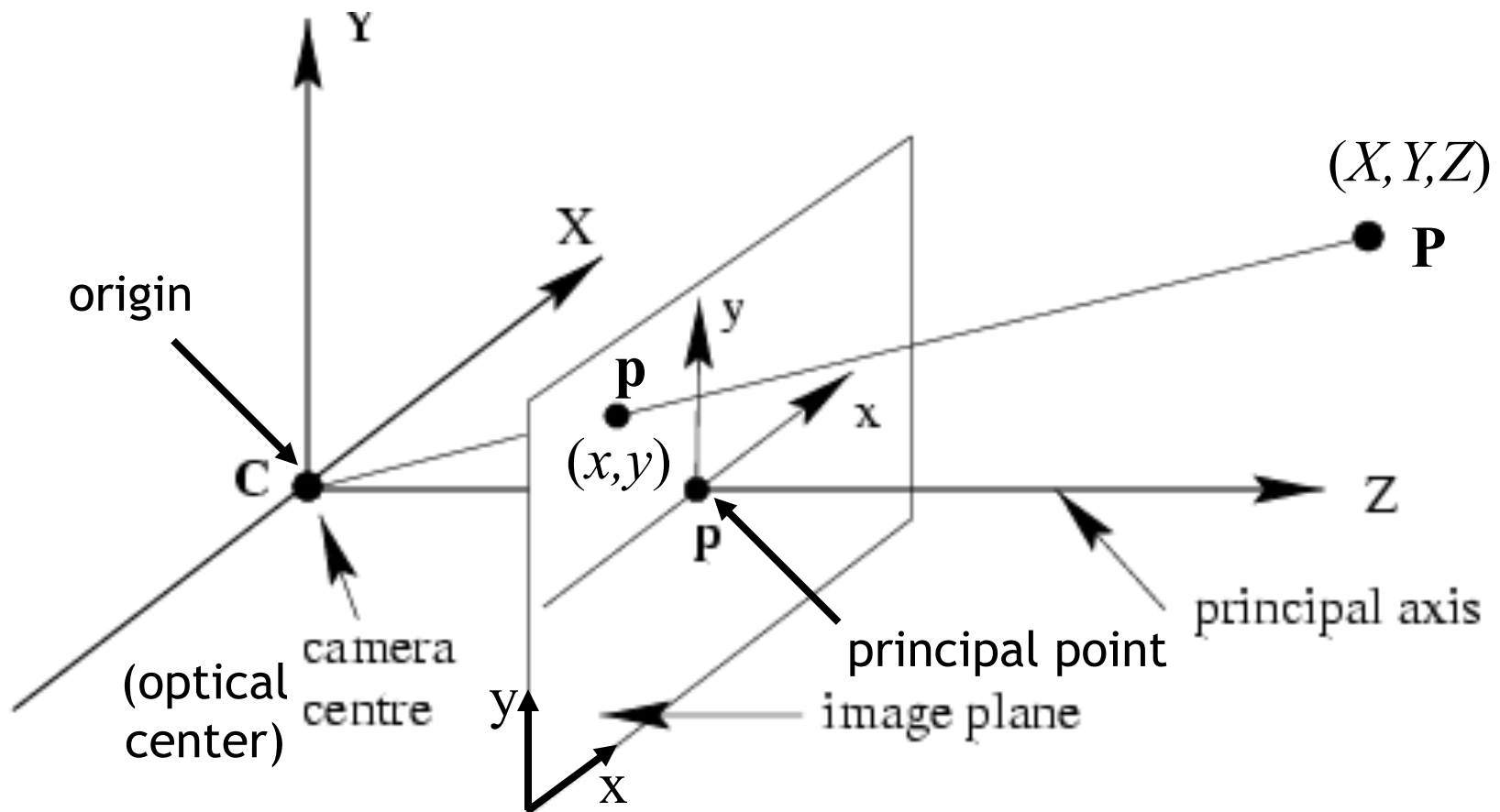


Last Lecture



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

Camera calibration

Camera calibration

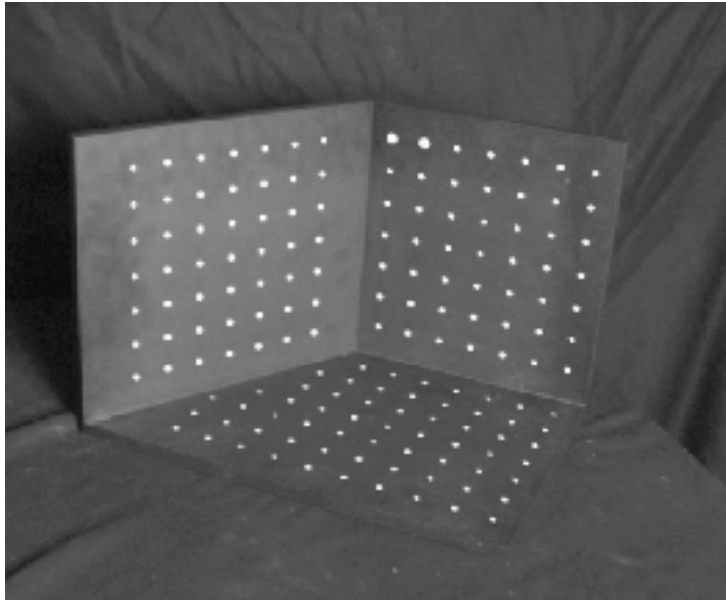
$$\begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \sim \begin{bmatrix} f & 0 & x_c \\ 0 & f & y_c \\ 0 & 0 & 1 \end{bmatrix} [\mathbf{R} | \mathbf{t}] \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix} = \begin{bmatrix} a & b & c & d \\ e & f & g & h \\ i & j & k & l \end{bmatrix} \begin{pmatrix} X \\ Y \\ Z \\ 1 \end{pmatrix}$$

- Estimate both intrinsic and extrinsic parameters
- Mainly, two categories:
 1. Using objects with known geometry as reference
 2. Self calibration (structure from motion)

One app of camera pose application

- Virtual gaming
 - http://www.livestream.com/emtech/video?clipId=pla_74103098-95fb-4704-99f0-d07339dc16a1

Camera calibration approaches



- 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)

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

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

Linear regression

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

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

Linear regression

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

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

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

Linear regression

$$\begin{bmatrix} X_1 & Y_1 & Z_1 & 1 & 0 & 0 & 0 & 0 & -u_1 X_1 & -u_1 Y_1 & -u_1 Z_1 \\ 0 & 0 & 0 & 0 & X_1 & Y_1 & Z_1 & 1 & -v_1 X_1 & -v_1 Y_1 & -v_1 Z_1 \\ & & & & & & \vdots & & & & \\ X_N & Y_N & Z_N & 1 & 0 & 0 & 0 & 0 & -u_N X_N & -u_N Y_N & -u_N Z_N \\ 0 & 0 & 0 & 0 & X_N & Y_N & Z_N & 1 & -v_N X_N & -v_N Y_N & -v_N Z_N \end{bmatrix} \begin{bmatrix} m_{00} \\ m_{01} \\ m_{02} \\ m_{03} \\ m_{10} \\ m_{11} \\ m_{12} \\ m_{13} \\ m_{20} \\ m_{21} \\ m_{22} \end{bmatrix} = \begin{bmatrix} u_i \\ v_i \end{bmatrix}$$

Solve for Projection Matrix M using least-square techniques

Normal equation (Geometric Interpretation)

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

$$\min \|\mathbf{Ax} - \mathbf{b}\|^2$$

$$\Leftrightarrow \mathbf{A}^T (\mathbf{Ax} - \mathbf{b}) = 0$$

$$\Leftrightarrow x = \left(\mathbf{A}^T \mathbf{A} \right)^{-1} \mathbf{A}^T \mathbf{b}$$

Normal equation (Differential Interpretation)

$$E(\mathbf{x}) = \|\mathbf{Ax} - \mathbf{b}\|^2$$

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

$n \times m$, n equations, m variables

Normal equation

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

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



Carl Friedrich Gauss

Any issues with the method?

Nonlinear optimization

- A probabilistic view of least square
- Feature measurement equations

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

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

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

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

Optimal estimation

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

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

$$= \sum_{i=1}^N \frac{1}{\sigma_i^2} \left(u_i - \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1} \right)^2 + \frac{1}{\sigma_i^2} \left(v_i - \frac{m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1} \right)^2$$

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

Nonlinear least square methods

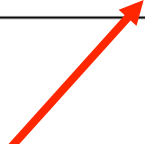
Least square fitting

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 $m \geq n$.

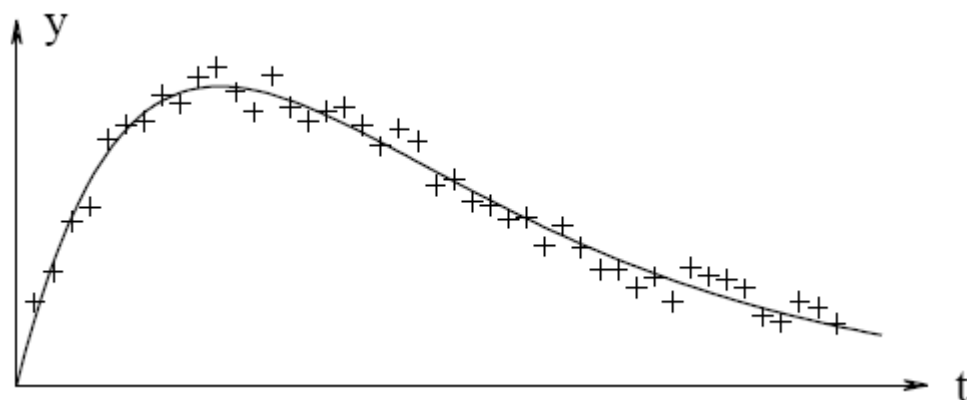


number of data points



number of parameters

Nonlinear least square fitting



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

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

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

$$= y_i - x_3 e^{x_1 t_i} - x_4 e^{x_2 t_i}$$

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*,

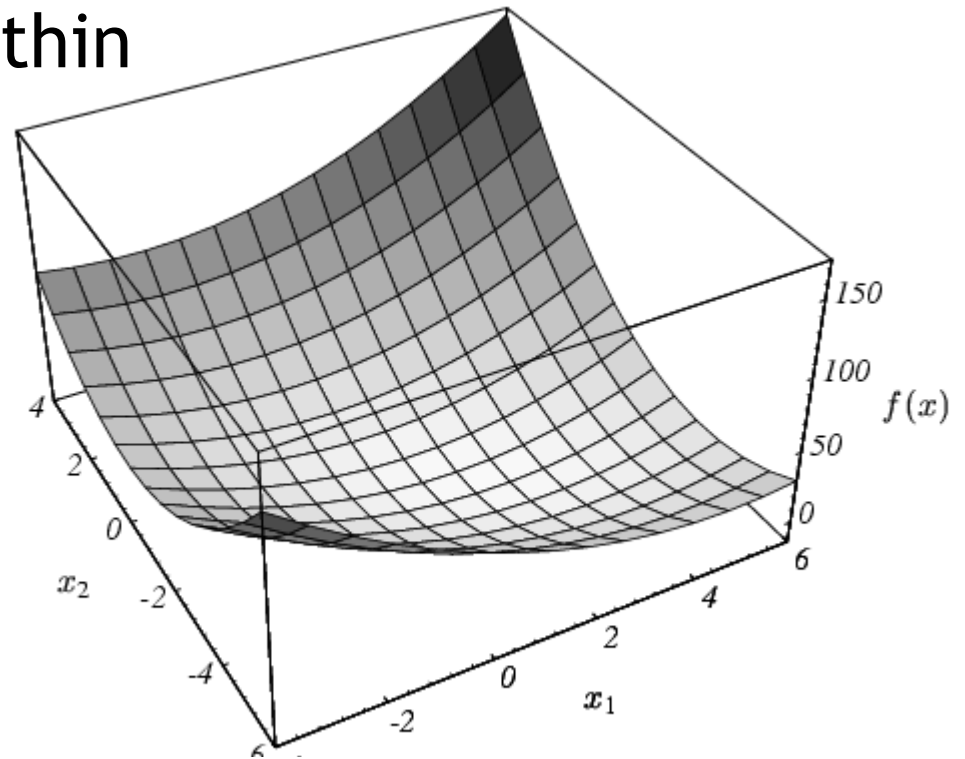
$$\mathbf{g} \equiv \mathbf{F}'(\mathbf{x}) = \begin{bmatrix} \frac{\partial F}{\partial x_1}(\mathbf{x}) \\ \vdots \\ \frac{\partial F}{\partial x_n}(\mathbf{x}) \end{bmatrix} ,$$

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



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

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

Function minimization

Theorem 1.5. Necessary condition for a local minimizer.

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

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

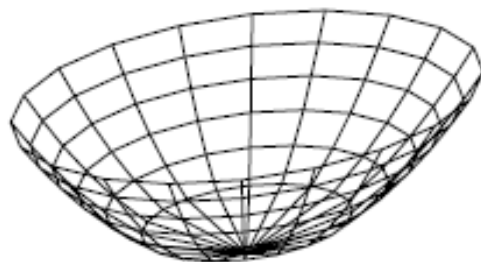
Definition 1.6. Stationary point. If

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

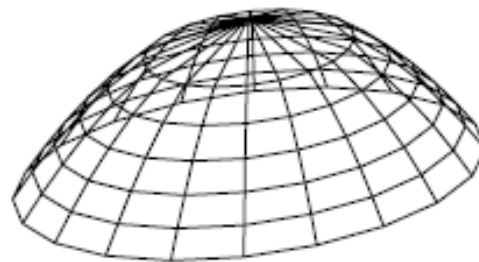
then $\mathbf{x}_s$ is said to be a *stationary point* for F .

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

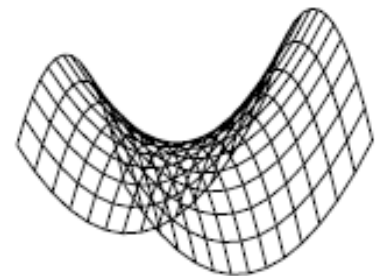
$\mathbf{H}_s$ is *positive definite*.



a) *minimum*



b) *maximum*



c) *saddle point*

Computing gradient and Hessian

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

Computing gradient and Hessian

$$\begin{aligned} F(\mathbf{x} + \mathbf{h}) &= \frac{1}{2} \sum_{i=1}^m \left(f_i(\mathbf{x} + \mathbf{h}) \right)^2 \\ &= \frac{1}{2} \sum_{i=1}^m \left(f_i(\mathbf{x}) + \frac{\partial f_i}{\partial \mathbf{x}}^{\text{T}} \mathbf{h} + \frac{1}{2} \mathbf{h}^{\text{T}} \frac{\partial^2 f_i}{\partial \mathbf{x}^2} \mathbf{h} \right)^2 \end{aligned}$$

Computing gradient and Hessian

$$\begin{aligned} F(\mathbf{x} + \mathbf{h}) &= \frac{1}{2} \sum_{i=1}^m (f_i(\mathbf{x} + \mathbf{h}))^2 \\ &= \frac{1}{2} \sum_{i=1}^m \left(f_i(\mathbf{x}) + \frac{\partial f_i}{\partial \mathbf{x}}^T \mathbf{h} + \frac{1}{2} \mathbf{h}^T \frac{\partial^2 f_i}{\partial \mathbf{x}^2} \mathbf{h} \right)^2 \\ &= \frac{1}{2} \sum_{i=1}^m \left((f_i(\mathbf{x}))^2 + 2 \left(f_i(\mathbf{x}) \frac{\partial f_i}{\partial \mathbf{x}} \right)^T \mathbf{h} + \mathbf{h}^T \left(f_i(\mathbf{x}) \frac{\partial^2 f_i}{\partial \mathbf{x}^2} + \frac{\partial f_i}{\partial \mathbf{x}} \frac{\partial f_i}{\partial \mathbf{x}}^T \right) \mathbf{h} \right) \end{aligned}$$

Computing gradient and Hessian

$$\begin{aligned} F(\mathbf{x} + \mathbf{h}) &= \frac{1}{2} \sum_{i=1}^m (f_i(\mathbf{x} + \mathbf{h}))^2 \\ &= \frac{1}{2} \sum_{i=1}^m \left(f_i(\mathbf{x}) + \frac{\partial f_i}{\partial \mathbf{x}}^T \mathbf{h} + \frac{1}{2} \mathbf{h}^T \frac{\partial^2 f_i}{\partial \mathbf{x}^2} \mathbf{h} \right)^2 \\ &= \frac{1}{2} \sum_{i=1}^m \left((f_i(\mathbf{x}))^2 + 2 \left(f_i(\mathbf{x}) \frac{\partial f_i}{\partial \mathbf{x}} \right)^T \mathbf{h} + \mathbf{h}^T \left(f_i(\mathbf{x}) \frac{\partial^2 f_i}{\partial \mathbf{x}^2} + \frac{\partial f_i}{\partial \mathbf{x}} \frac{\partial f_i}{\partial \mathbf{x}}^T \right) \mathbf{h} \right) \\ &= \frac{1}{2} \sum_{i=1}^m (f_i(\mathbf{x}))^2 + \left(\sum_{i=1}^m f_i(\mathbf{x}) \frac{\partial f_i}{\partial \mathbf{x}} \right)^T \mathbf{h} + \frac{1}{2} \mathbf{h}^T \left(\sum_{i=1}^m f_i(\mathbf{x}) \frac{\partial^2 f_i}{\partial \mathbf{x}^2} + \frac{\partial f_i}{\partial \mathbf{x}} \frac{\partial f_i}{\partial \mathbf{x}}^T \right) \mathbf{h} \end{aligned}$$

Computing gradient and Hessian

$$\begin{aligned} F(\mathbf{x} + \mathbf{h}) &= \frac{1}{2} \sum_{i=1}^m (f_i(\mathbf{x} + \mathbf{h}))^2 \\ &= \frac{1}{2} \sum_{i=1}^m \left(f_i(\mathbf{x}) + \frac{\partial f_i}{\partial \mathbf{x}}^T \mathbf{h} + \frac{1}{2} \mathbf{h}^T \frac{\partial^2 f_i}{\partial \mathbf{x}^2} \mathbf{h} \right)^2 \\ &= \frac{1}{2} \sum_{i=1}^m \left((f_i(\mathbf{x}))^2 + 2 \left(f_i(\mathbf{x}) \frac{\partial f_i}{\partial \mathbf{x}} \right)^T \mathbf{h} + \mathbf{h}^T \left(f_i(\mathbf{x}) \frac{\partial^2 f_i}{\partial \mathbf{x}^2} + \frac{\partial f_i}{\partial \mathbf{x}} \frac{\partial f_i}{\partial \mathbf{x}}^T \right) \mathbf{h} \right) \\ &= \frac{1}{2} \sum_{i=1}^m (f_i(\mathbf{x}))^2 + \left(\sum_{i=1}^m f_i(\mathbf{x}) \frac{\partial f_i}{\partial \mathbf{x}} \right)^T \mathbf{h} + \frac{1}{2} \mathbf{h}^T \left(\sum_{i=1}^m f_i(\mathbf{x}) \frac{\partial^2 f_i}{\partial \mathbf{x}^2} + \frac{\partial f_i}{\partial \mathbf{x}} \frac{\partial f_i}{\partial \mathbf{x}}^T \right) \mathbf{h} \\ &\approx \frac{1}{2} \sum_{i=1}^m (f_i(\mathbf{x}))^2 + \underbrace{\left(\sum_{i=1}^m f_i(\mathbf{x}) \frac{\partial f_i}{\partial \mathbf{x}} \right)^T}_{\text{Gradient}} \mathbf{h} + \frac{1}{2} \mathbf{h}^T \underbrace{\left(\sum_{i=1}^m \frac{\partial f_i}{\partial \mathbf{x}} \frac{\partial f_i}{\partial \mathbf{x}}^T \right)}_{\text{Hessian}} \mathbf{h} \end{aligned}$$

Searching for update $\mathbf{h}$

$$F(\mathbf{x} + \mathbf{h}) \approx \frac{1}{2} \sum_{i=1}^m (f_i(\mathbf{x}))^2 + \overset{\text{Gradient}}{\mathbf{g}^T \mathbf{h}} + \frac{1}{2} \overset{\text{Hessian}}{\mathbf{h}^T \mathbf{H} \mathbf{h}}$$

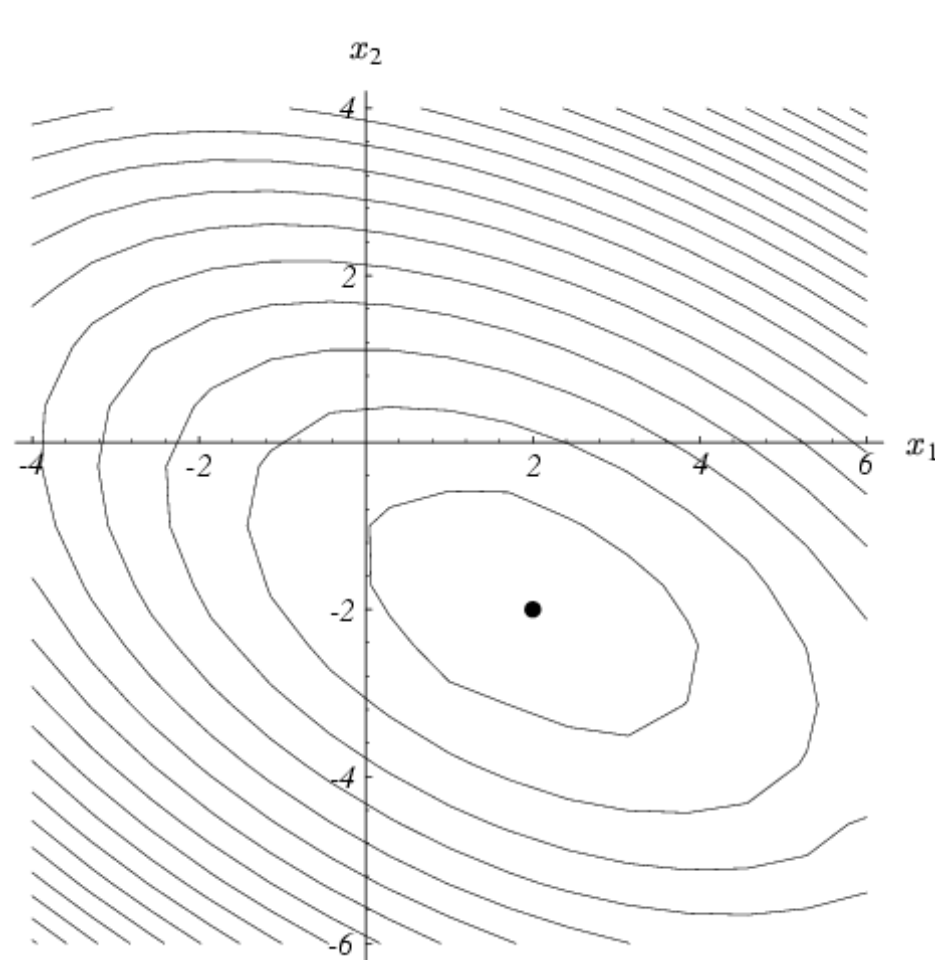
Idea 1: Steepest Descent

Let $\mathbf{h} = -\alpha \mathbf{g}$

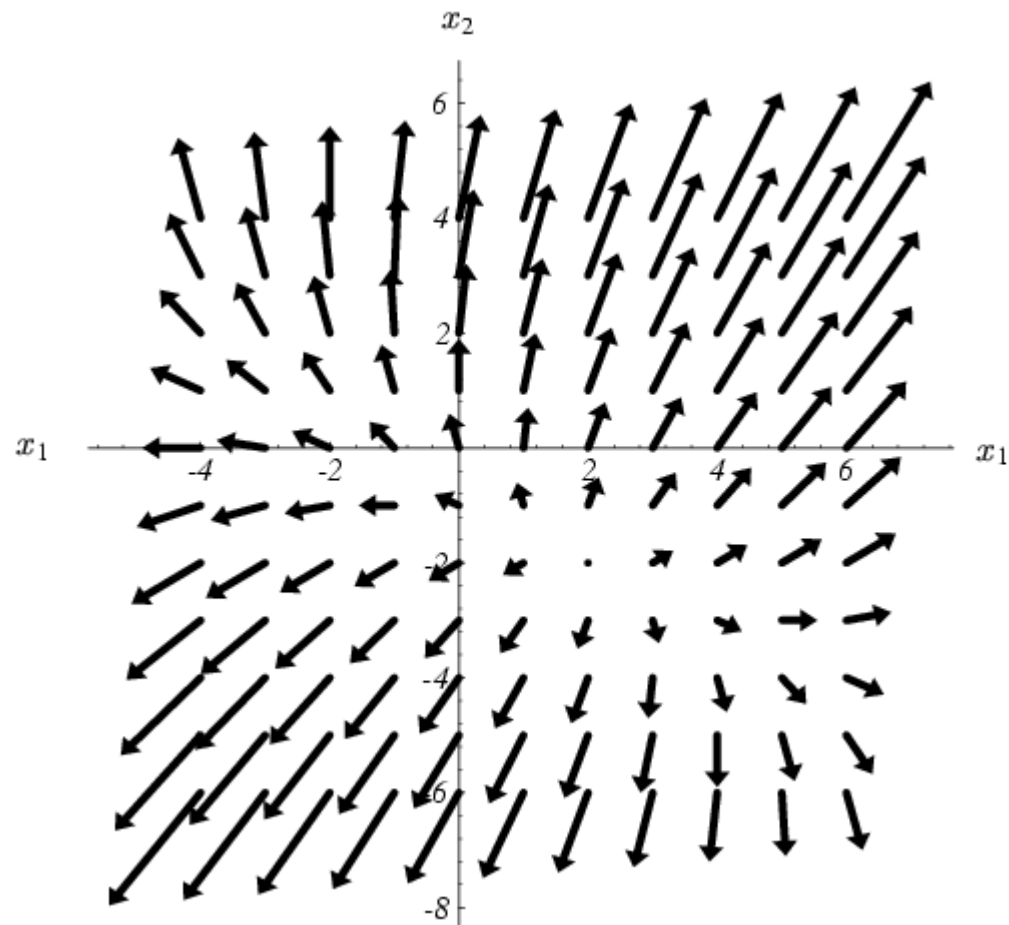
$$F(\mathbf{x} + \mathbf{h}) \approx \frac{1}{2} \sum_{i=1}^m (f_i(\mathbf{x}))^2 - \alpha \mathbf{g}^T \mathbf{g} + \frac{\alpha^2}{2} \mathbf{g}^T \mathbf{H} \mathbf{g}$$

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

Steepest descent method

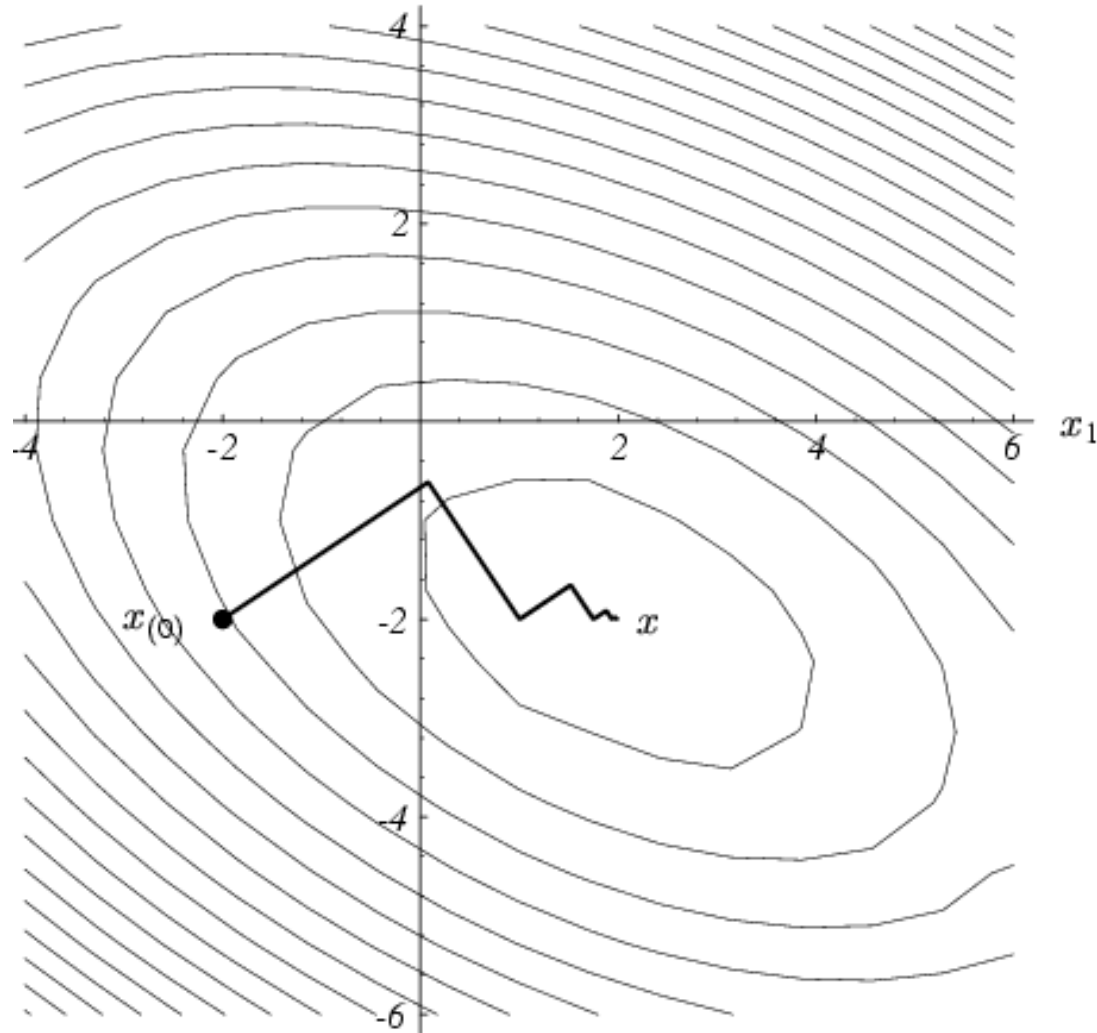


isocontour



gradient

Steepest descent method



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

Searching for update $\mathbf{h}$

$$F(\mathbf{x} + \mathbf{h}) \approx \frac{1}{2} \sum_{i=1}^m (f_i(\mathbf{x}))^2 + \overset{\text{Gradient}}{\mathbf{g}^T \mathbf{h}} + \frac{1}{2} \mathbf{h}^T \overset{\text{Hessian}}{\mathbf{H}} \mathbf{h}$$

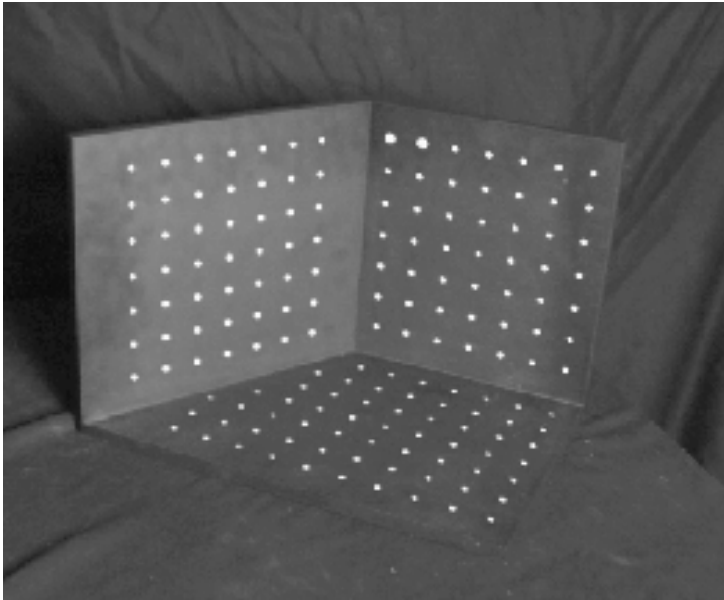
Idea 2: minimizing the quadric directly

$$\text{Let } \frac{\partial}{\partial \mathbf{h}} F(\mathbf{x} + \mathbf{h}) \approx \mathbf{g} + \mathbf{H}\mathbf{h} = 0$$

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

Converge faster but needs to solve the linear system

Recap: Calibration



- 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)

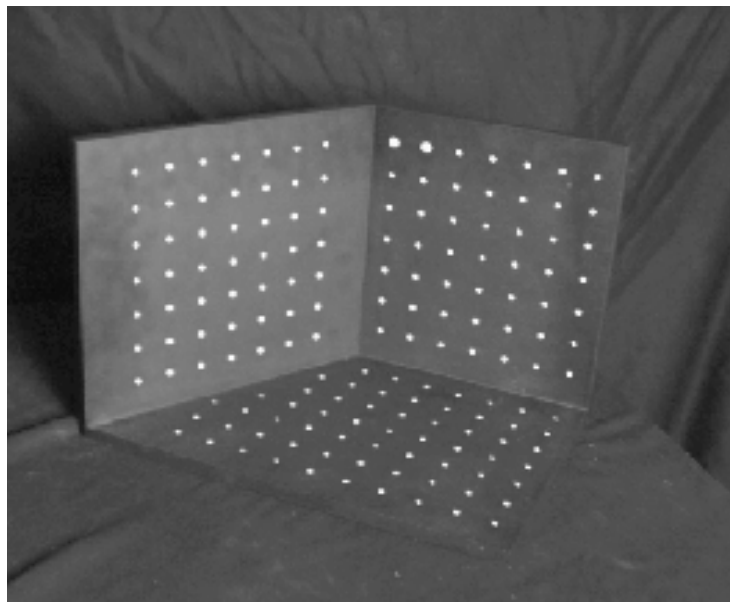
$$\mathbf{x} \sim \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{X} = \mathbf{M}\mathbf{X}$$

Camera Model:

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

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

Recap: Calibration



- 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)

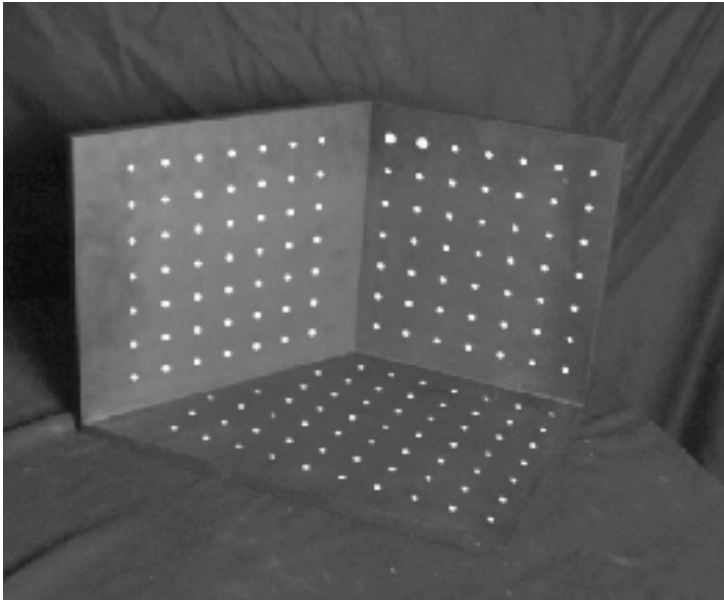
$$\mathbf{x} \sim \mathbf{K}[\mathbf{R}|\mathbf{t}]\mathbf{X} = \mathbf{MX}$$

Linear Approach:

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

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

Recap: Calibration



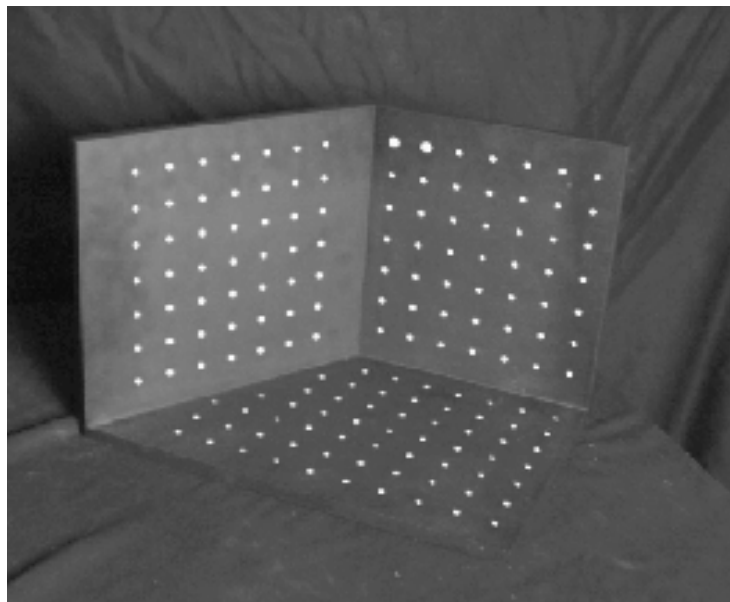
- Directly estimate 11 unknowns in the **M** matrix using known 3D points (X_i, Y_i, Z_i) and measured feature positions (u_i, v_i)

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

NonLinear Approach:

$$\sum_{i=1}^N \left(u_i - \frac{m_{00}X_i + m_{01}Y_i + m_{02}Z_i + m_{03}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1} \right)^2 + \left(v_i - \frac{m_{10}X_i + m_{11}Y_i + m_{12}Z_i + m_{13}}{m_{20}X_i + m_{21}Y_i + m_{22}Z_i + 1} \right)^2$$

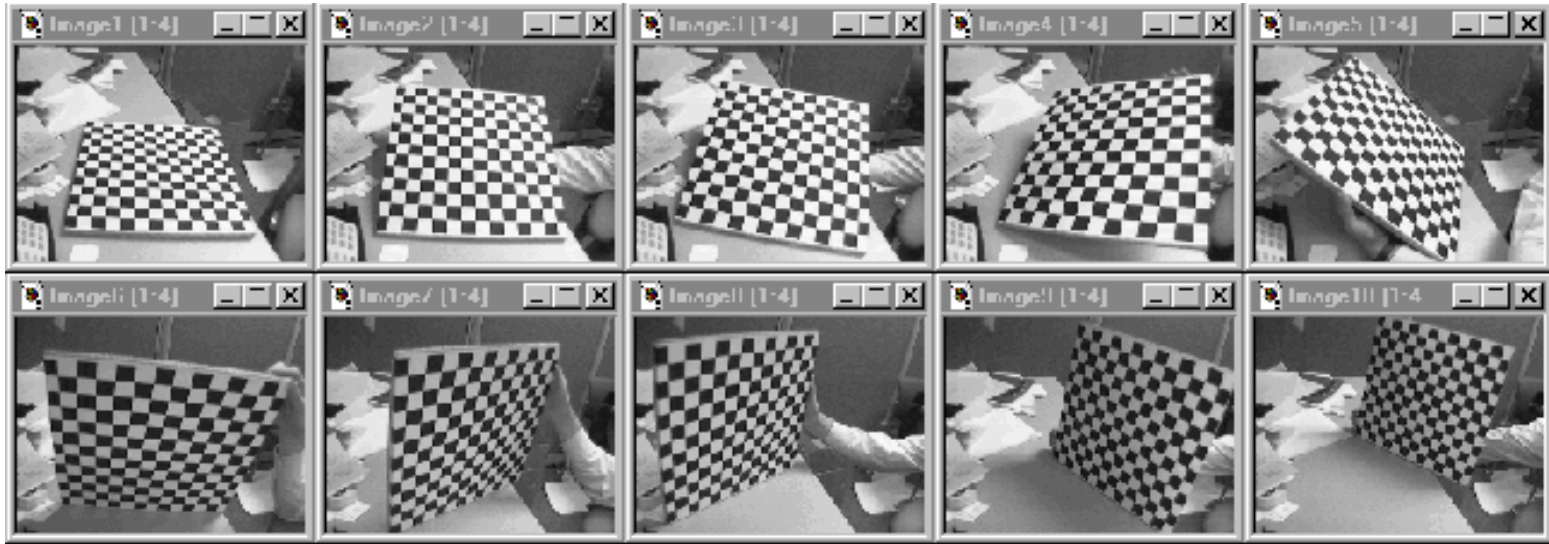
Practical Issue



is hard to make and the 3D feature positions are difficult to measure!

A popular calibration tool

Multi-plane calibration

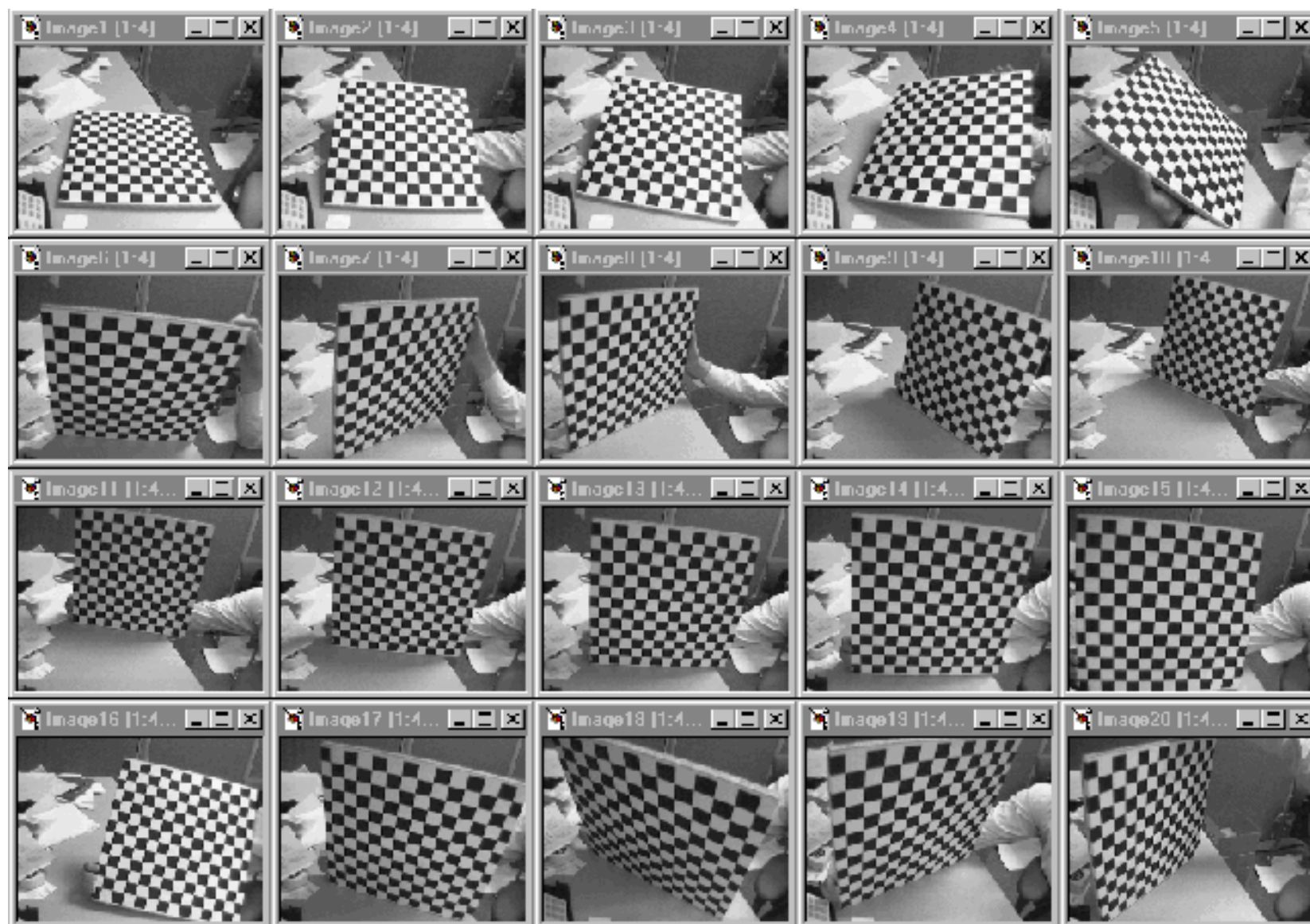


Images courtesy Jean-Yves Bouguet, Intel Corp.

Advantage

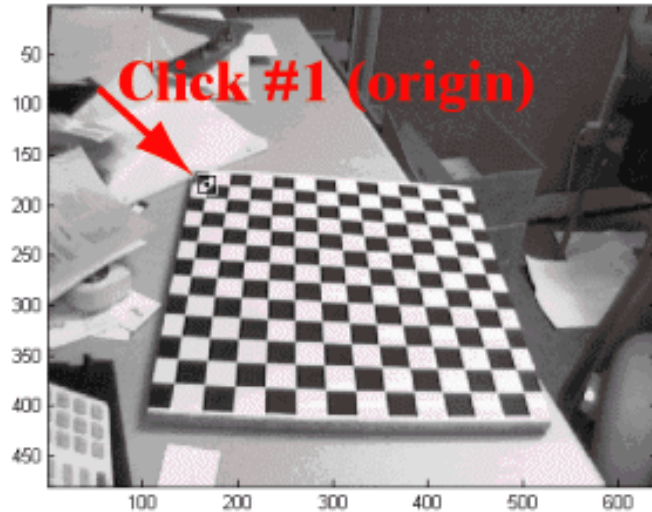
- Only requires a plane
- Don't have to know positions/orientations
- Good code available online!
 - Intel's OpenCV library: <http://www.intel.com/research/mrl/research/opencv/>
 - Matlab version by Jean-Yves Bouget: http://www.vision.caltech.edu/bouguetj/calib_doc/index.html
 - Zhengyou Zhang's web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

Step 1: data acquisition

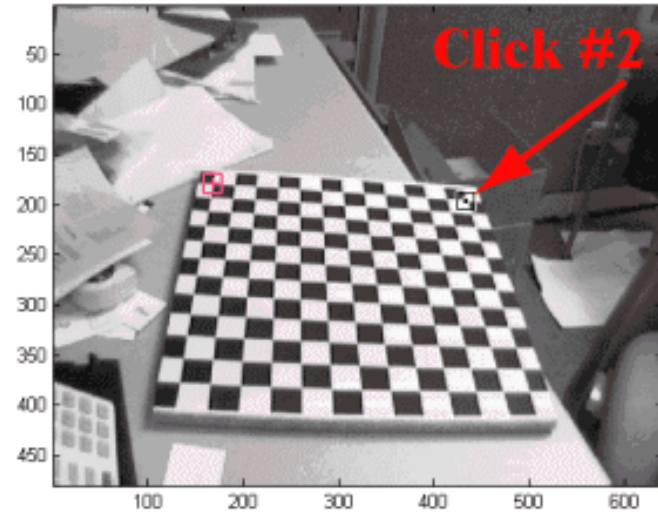


Step 2: specify corner order

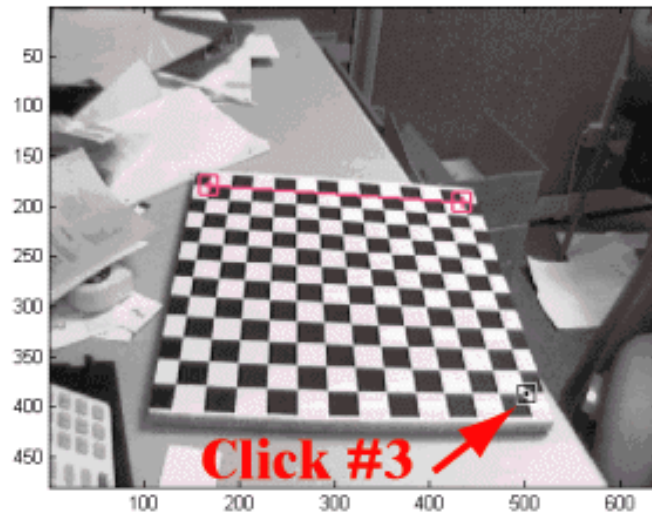
Click on the four extreme corners of the rectangular pattern (first corner = origin)... Image 1



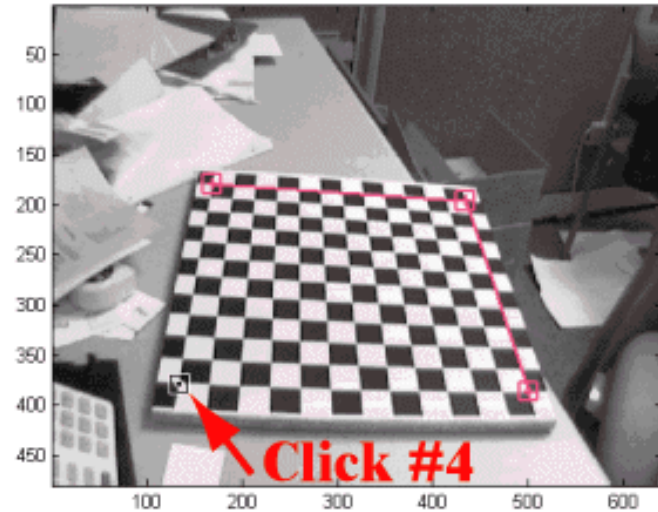
Click on the four extreme corners of the rectangular pattern (first corner = origin)... Image 1



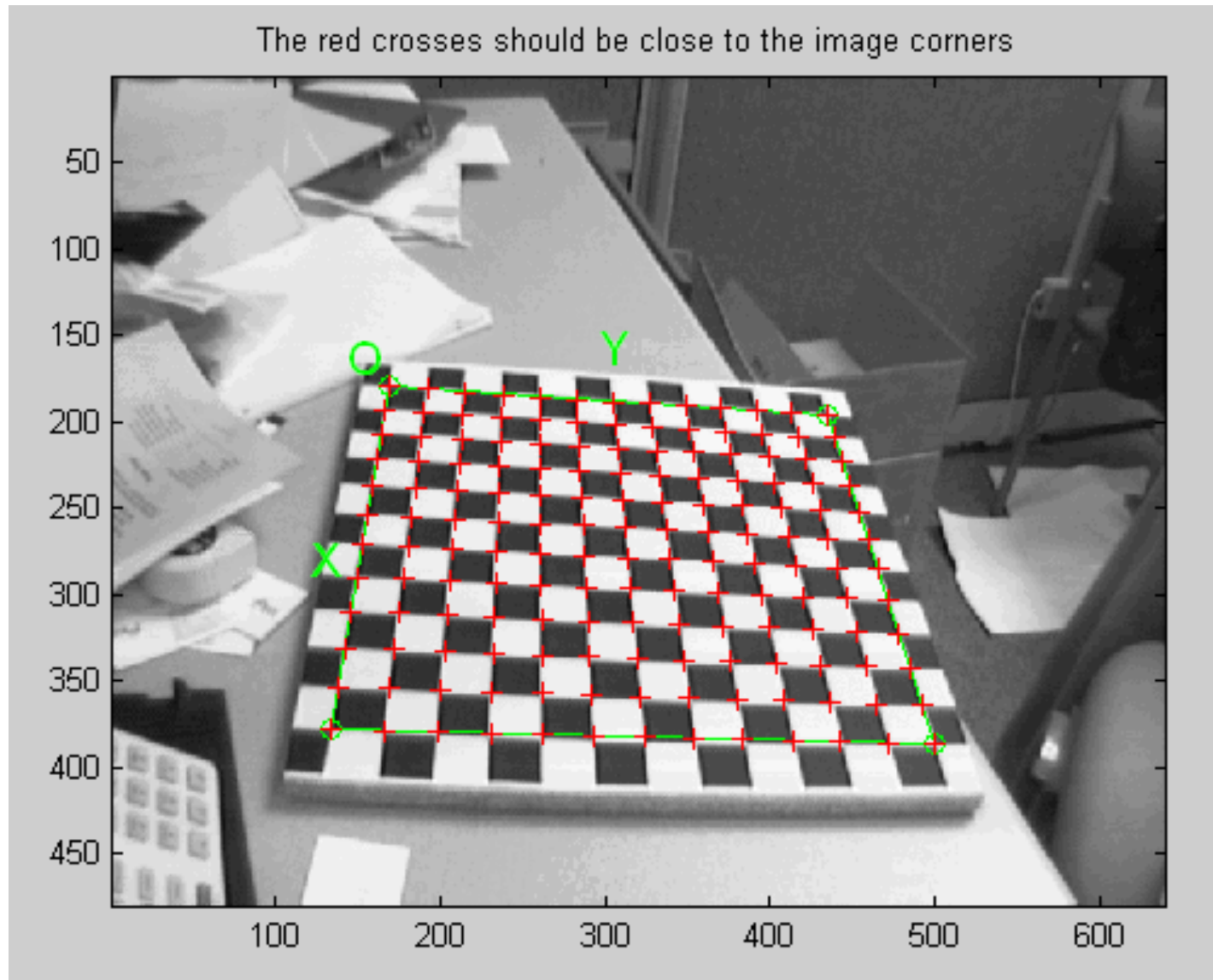
Click on the four extreme corners of the rectangular pattern (first corner = origin)... Image 1



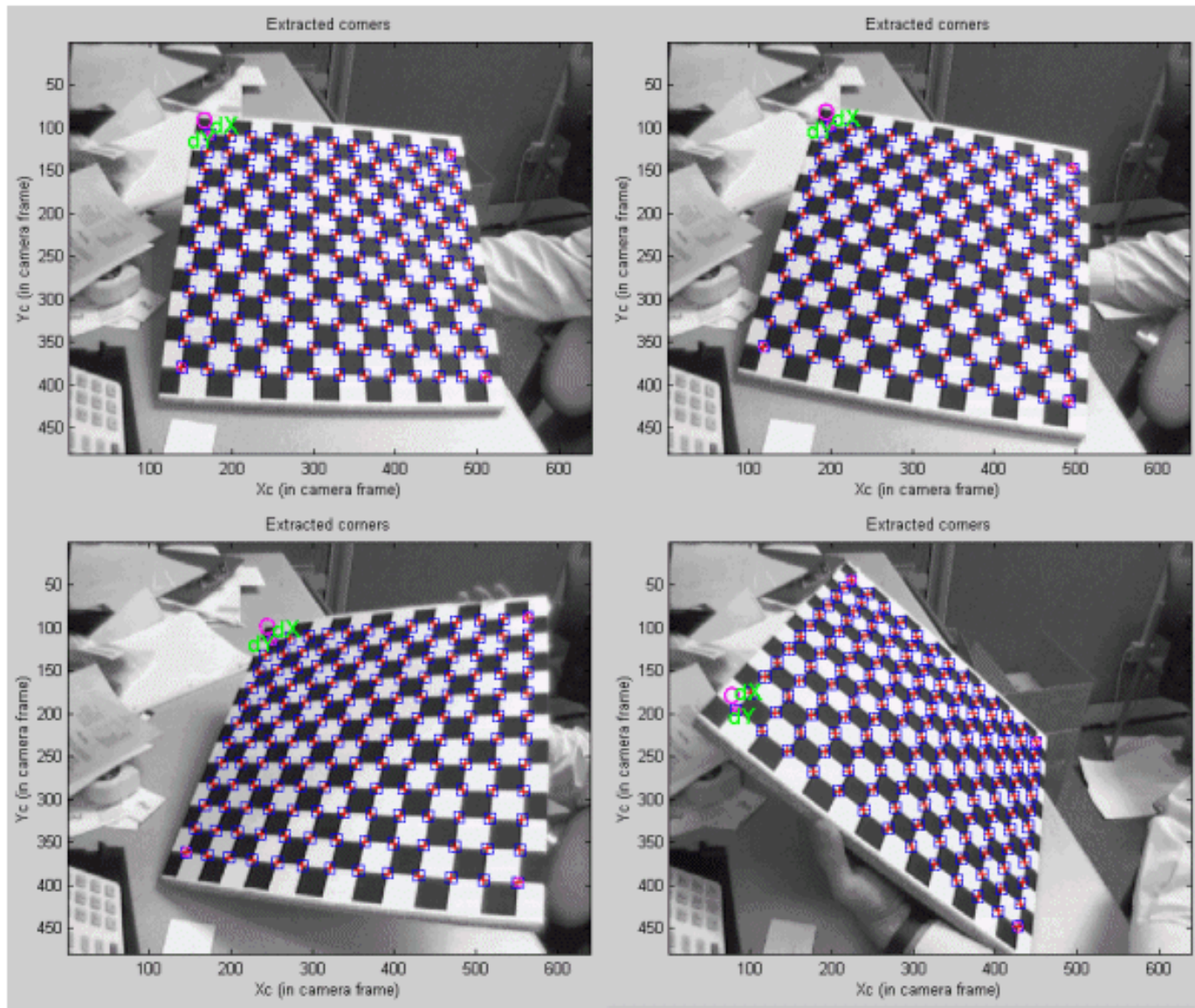
Click on the four extreme corners of the rectangular pattern (first corner = origin)... Image 1



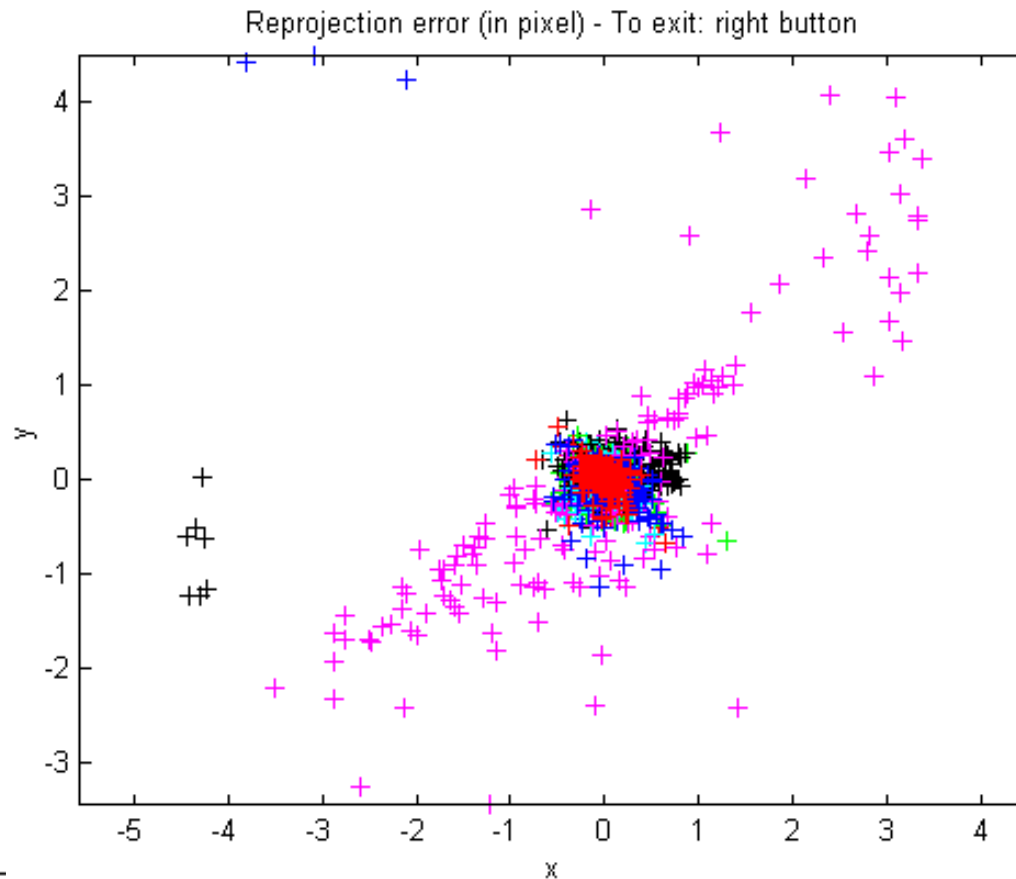
Step 3: corner extraction



Step 3: corner extraction

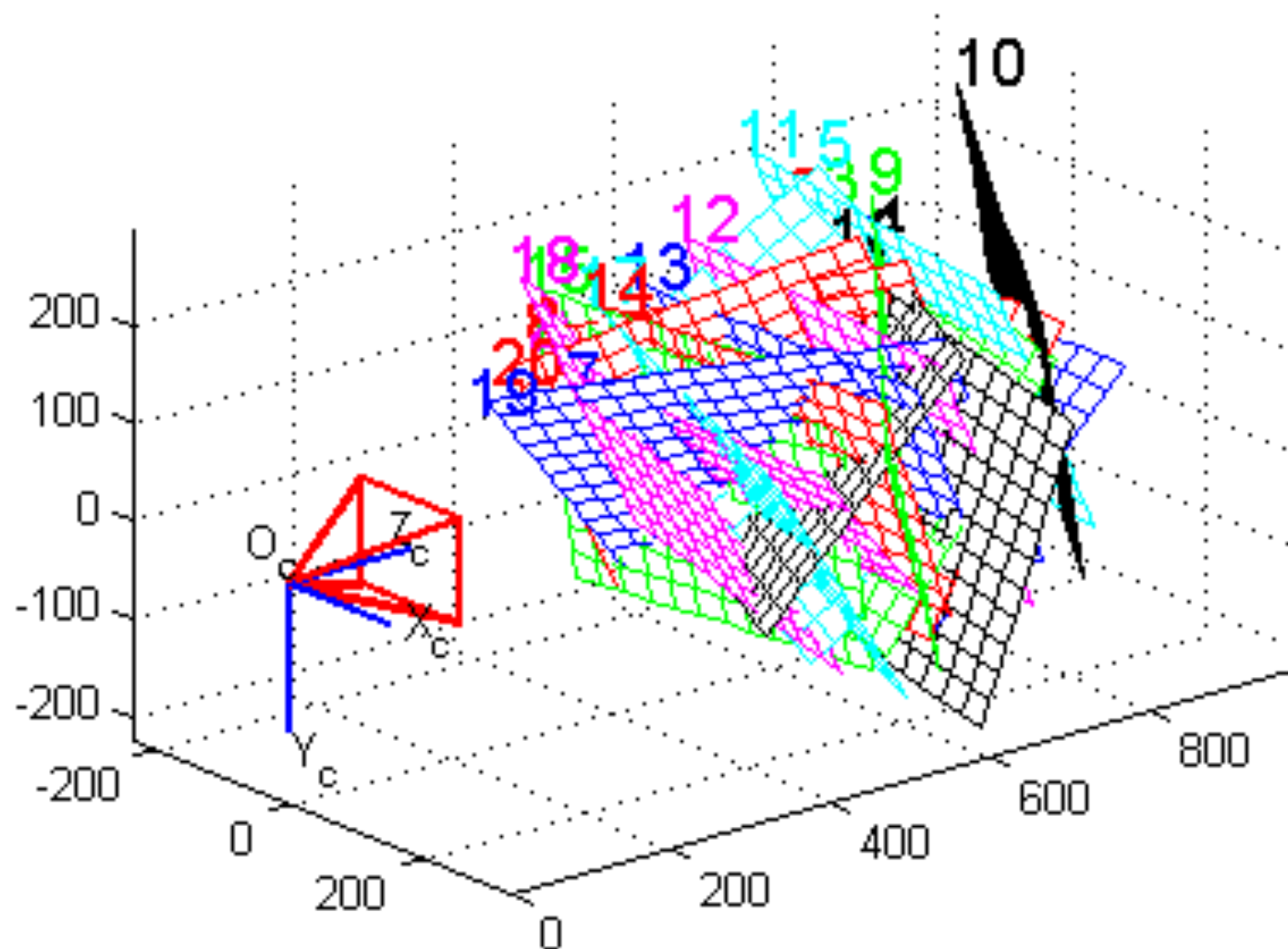


Step 4: minimize projection error

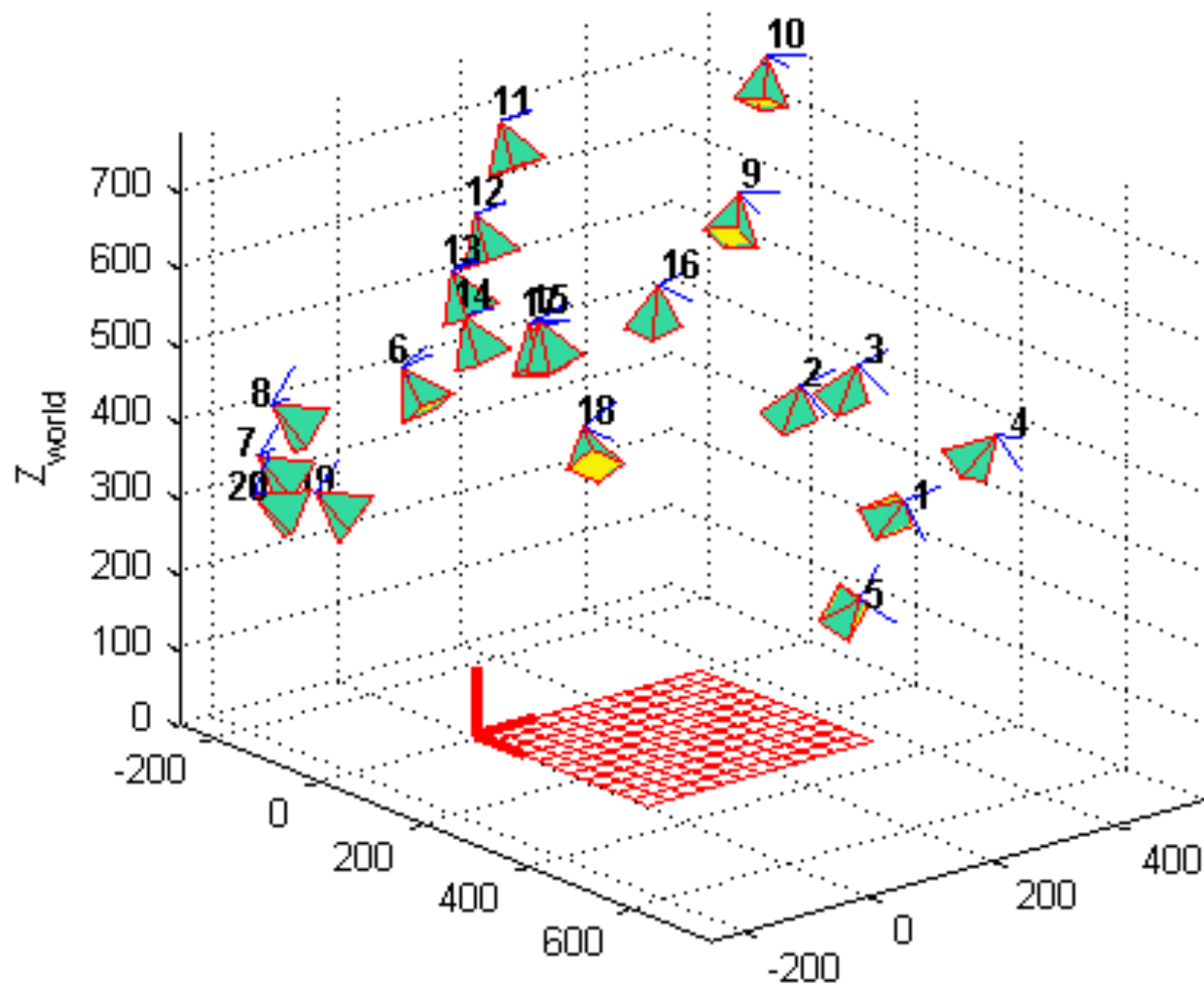


```
Focal Length:      fc = [ 657.46290  657.94673 ] ± [ 0.31819  0.34046 ]
Principal point:    cc = [ 303.13665  242.56935 ] ± [ 0.64682  0.59218 ]
Skew:              alpha_c = [ 0.00000 ] ± [ 0.00000 ] => angle of pixel axes =
Distortion:         kc = [ -0.25403  0.12143  -0.00021  0.00002  0.00000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

Step 4: camera calibration



Step 4: camera calibration



Step 5: refinement

