Camera calibration

Digital Visual Effects, Spring 2005
Yung-Yu Chuang
2005/4/6

with slides by Richard Szeliski, Steve Seitz, and Marc Pollefeys
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 $x^*$, a local minimizer for

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

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

\[ y(t) = M(x, t) = x_0 + x_1 t \]

\[ f_i(x) = y_i - M(x, t_i) \]

\[ M(x, t) = x_0 + x_1 t + x_2 t^3 \] is linear, too.
Nonlinear least square

\[ M(x, t) = x_3 e^{x_1 t} + x_4 e^{x_2 t} \]

parameters \( x = [x_1, x_2, x_3, x_4]^T \)

residuals \( f_i(x) = y_i - M(x, t_i) \)
\[ = y_i - x_3 e^{x_1 t_i} - x_4 e^{x_2 t_i} \]
Function minimization

Least square is related to function minimization.

Global Minimizer
Given $F : \mathbb{R}^n \to \mathbb{R}$. Find

$$x^+ = \arg\min_x \{F(x)\}.$$  

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

Local Minimizer
Given $F : \mathbb{R}^n \to \mathbb{R}$. Find $x^*$ so that

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

We assume that the cost function $F$ is differentiable and so smooth that the following Taylor expansion is valid, \(^2\)

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

where $g$ is the gradient,

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

and $H$ is the Hessian,

$$H \equiv F''(x) = \begin{bmatrix}
\frac{\partial^2 F}{\partial x_i \partial x_j}(x)
\end{bmatrix}.$$
Quadratic functions

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

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

isocontour

gradient
Quadratic functions
Descent methods

1. Find a descent direction $h_d$
2. Find a step length giving a good decrease in the $F$-value.

Algorithm Descent method

begin
  $k := 0; \ x := x_0; \ found := \text{false}$ \hspace{1cm} \{Starting point\}
  while (not found) and ($k < k_{max}$)
    $h_d := \text{search\_direction}(x)$ \hspace{1cm} \{From $x$ and downhill\}
    if (no such $h$ exists)
      $found := \text{true}$ \hspace{1cm} \{x is stationary\}
    else
      $\alpha := \text{step\_length}(x, h_d)$ \hspace{1cm} \{from $x$ in direction $h_d$\}
      $x := x + \alpha h_d; \ k := k+1$ \hspace{1cm} \{next iterate\}
  end
Descent direction

\[ F(x + \alpha h) = F(x) + \alpha h^\top F'(x) + O(\alpha^2) \]

\[ \simeq F(x) + \alpha h^\top F'(x) \quad \text{for } \alpha \text{ sufficiently small.} \]

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

**Definition Descent direction.**

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

If no such \( h \) exists, then \( F'(x) = 0 \), showing that in this case \( x \) is stationary.
Steepest descent method

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

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

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

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

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

![Diagram showing the steepest descent method](image-url)
Newton’s method

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

\[
F'(x+h) = F'(x) + F''(x)h + O(\|h\|^2)
\]

\[
\approx F'(x) + F''(x)h \quad \text{for } \|h\| \text{ sufficiently small}
\]

we derive Newton’s method: Find \( h_n \) as the solutions to

\[
H h_n = -F'(x) \quad \text{with} \quad H = F''(x),
\]  

(2.9a)

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

\[
0 < h_n^T H h_n = -h_n^T F'(x),
\]  

(2.10)

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

\[
\text{if } F''(x) \text{ is positive definite} \\
h := h_n \\
\text{else} \\
h := h_{sd} \\
x := x + \alpha h
\]

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

\[ \varphi(\alpha) = F(x + \alpha h) , \quad x \text{ and } h \text{ fixed, } \alpha \geq 0 . \]
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 $x$, try to find the best parameter vector $p$ so that the squared distance $\varepsilon \varepsilon^T$ is minimal. Here, $\varepsilon = x - \hat{x}$, with $\hat{x} = f(p)$. 
Levenberg-Marquardt method

For a small $||\delta_p||$, $f(p + \delta_p) \approx f(p) + J\delta_p$

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

it is required to find the $\delta_p$ that minimizes the quantity

$||x - f(p + \delta_p)|| \approx ||x - f(p) - J\delta_p|| = ||\epsilon - J\delta_p||$

$J^TJ\delta_p = J^T\epsilon$

$N\delta_p = J^T\epsilon$

$N_{ii} = \mu + [J^T J]_{ii}$

damping term
Levenberg-Marquardt method

If a covariance matrix $\Sigma_x$ for the measured vector $x$ is available, it can be incorporated into the LM algorithm by minimizing the squared $\Sigma_x^{-1}$-norm $\epsilon^T \Sigma_x^{-1} \epsilon$ 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

$$J^T \Sigma_x^{-1} J \delta_p = J^T \Sigma_x^{-1} \epsilon.$$  \hspace{1cm} (4)
Algorithm:
\[ k := 0; \nu := 2; p := p_0; \]
\[ A := J^T J; \epsilon_p := x - f(p); g := J^T \epsilon_p; \]
\[ \text{stop} := (||g||_\infty \leq \epsilon_1); \mu := \tau \cdot \max_{i=1,\ldots,m}(A_{ii}); \]
while (not stop) and (\[ k < k_{\text{max}} \])
\[ k := k + 1; \]
repeat
\[ \text{Solve (} A + \mu I) \delta_p = g; \]
\[ \text{if (} ||\delta_p|| \leq \epsilon_2 ||p|| \text{)} \]
\[ \text{stop} := \text{true}; \]
else
\[ p_{\text{new}} := p + \delta_p; \]
\[ \rho := (||\epsilon_p||^2 - ||x - f(p_{\text{new}})||^2)/(\delta_p^T (\mu \delta_p + g)); \]
\[ \text{if } \rho > 0 \]
\[ p := p_{\text{new}}; \]
\[ A := J^T J; \epsilon_p := x - f(p); g := J^T \epsilon_p; \]
\[ \text{stop} := (||g||_\infty \leq \epsilon_1); \]
\[ \mu := \mu \cdot \max(\frac{1}{3}, 1 - (2\rho - 1)^3); \nu := 2; \]
else
\[ \mu := \mu \cdot \nu; \nu := 2 \cdot \nu; \]
endif
until (\rho > 0) or (stop)
endwhile
Camera projection models
Pinhole camera

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

Soli deliquium Anno Christi
1544. Die 24: Januarij
Louaniæ

Sic nos exacte Anno 1544. Louanii eclipsis Solis
observeramus, inuenimusque; deficere paulo plus et dexter-
Pinhole camera model

- origin
- camera centre
- principal point
- (optical center)
- image plane
- principal axis
Pinhole camera model

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

\[
x = \frac{fX}{Z}
\]

\[
y = \frac{fY}{Z}
\]
Pinhole camera model

\[
\begin{pmatrix}
  x \\
  y \\
  1 \\
\end{pmatrix}
\sim
\begin{pmatrix}
  fX \\
  fY \\
  Z
\end{pmatrix}
= \begin{bmatrix}
  f & 0 & 0 & 1 & 0 & 0 & 0 \\
  0 & f & 0 & 0 & 1 & 0 & 0 \\
  0 & 0 & 1 & 0 & 0 & 1 & 0 \\
\end{bmatrix}
\begin{pmatrix}
  X \\
  Y \\
  Z \\
  1
\end{pmatrix}
\]
Principal point offset

Intrinsic matrix

\[
x \sim K[I|0]X
\]

\[
\begin{pmatrix}
x \\
y \\
1
\end{pmatrix}
\sim
\begin{pmatrix}
fX \\
fY \\
fZ
\end{pmatrix}
=
\begin{bmatrix}
f & 0 & x_0 \\
0 & f & y_0 \\
0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1
\end{bmatrix}
\begin{pmatrix}
X \\
Y \\
Z \\
1
\end{pmatrix}
\]
Intrinsic matrix

Is this form of $K$ good enough?

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

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

$$K = \begin{bmatrix} fa & s & x_0 \\ 0 & f & y_0 \\ 0 & 0 & 1 \end{bmatrix}$$
Camera rotation and translation

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

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

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

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

- Perspective
- Weak perspective

Increasing focal length

Increasing distance from camera
Orthographic projection

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

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

- Also called "parallel projection": \((x, y, z) \mapsto (x, y)\)
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 optimization
3. multiple planar patterns
Chromaglyphs (HP research)
Linear regression

\[ x \sim K[R|t]X = MX \]

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

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

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

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

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

Solve for Projection Matrix \( M \) using least-square techniques
Normal equation

Given an overdetermined system

\[ Ax = b \]

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

\[ A^T Ax = A^T b \]
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

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

• Likelihood of \( M \) given \( \{(u_i, v_i)\} \)

\[ L = \prod_i p(u_i | \hat{u}_i) p(v_i | \hat{v}_i) \]
\[ = \prod_i e^{-\frac{(u_i - \hat{u}_i)^2}{\sigma^2}} e^{-\frac{(v_i - \hat{v}_i)^2}{\sigma^2}} \]
Optimal estimation

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

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

• How do we minimize $C$?
• Non-linear regression (least squares), because $\hat{u}_i$ and $v_i$ are non-linear functions of $\mathbf{M}$
• 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!
Step 1: data acquisition
Step 2: specify corner order
Step 3: corner extraction

The red crosses should be close to the image corners.
Step 3: corner extraction
Step 4: minimize projection error

Calibration results

Focal Length: \( f_c = [\ 657.46290, 657.94673 \] \ ± \ [\ 0.31819, 0.34046 \]

Principal point: \( c_c = [\ 303.13665, 242.56935 \] \ ± \ [\ 0.64682, 0.59218 \]

Skew: \( \alpha_c = [\ 0.00000 \] \ ± \ [\ 0.00000 \] \ => angle of pixel axes =

Distortion: \( k_c = [\ -0.25403, 0.12143, -0.00021, 0.00002, 0.00000 \]

Pixel error: \( \text{err} = [\ 0.11689, 0.11500 \]
Step 4: camera calibration
Step 4: camera calibration
Step 5: refinement

Reprojection error (in pixel) - To exit: right button
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

$$
\min_{a_j, b_i} \sum_{i=1}^{n} \sum_{j=1}^{m} d(Q(a_j, b_i), x_{ij})^2
$$

predicted projection
Euclidean distance
Bundle adjustment
Algorithm:
\[ k := 0; \ \nu := 2; \ \mathbf{p} := \mathbf{p}_0; \]
\[ \mathbf{A} := \mathbf{J}^T\mathbf{J}; \ \mathbf{e}_p := \mathbf{x} - f(\mathbf{p}); \ \mathbf{g} := \mathbf{J}^T\mathbf{e}_p; \]
\[ \text{stop} := (||\mathbf{g}||_\infty \leq \varepsilon_1); \ \mu := \tau \max_{i=1,\ldots,m}(A_{ii}); \]
while (not stop) and \((k < k_{\text{max}})\)
\[ k := k + 1; \]
repeat
\[ \text{Solve } (\mathbf{A} + \mu \mathbf{I})\delta_p = \mathbf{g}; \]
\[ \text{if } (||\delta_p|| \leq \varepsilon_2||\mathbf{p}||) \]
\[ \text{stop} := \text{true}; \]
else
\[ \mathbf{p}_{\text{new}} := \mathbf{p} + \delta_p; \]
\[ \rho := (||\mathbf{e}_p||^2 - ||\mathbf{x} - f(\mathbf{p}_{\text{new}})||^2)/(\delta_p^T(\mu \delta_p + \mathbf{g})); \]
if \(\rho > 0\)
\[ \mathbf{p} := \mathbf{p}_{\text{new}}; \]
\[ \mathbf{A} := \mathbf{J}^T\mathbf{J}; \ \mathbf{e}_p := \mathbf{x} - f(\mathbf{p}); \ \mathbf{g} := \mathbf{J}^T\mathbf{e}_p; \]
\[ \text{stop} := (||\mathbf{g}||_\infty \leq \varepsilon_1); \]
\[ \mu := \mu \max(\frac{1}{3}, 1 - (2\rho - 1)^3); \ \nu := 2; \]
else
\[ \mu := \mu \cdot \nu; \ \nu := 2 \cdot \nu; \]
endif
until \((\rho > 0) \text{ or } (\text{stop})\)
endwhile
Bundle adjustment

3 views and 4 points

\[
\frac{\partial X}{\partial P} = \begin{pmatrix}
A_{11} & 0 & 0 & B_{11} & 0 & 0 & 0 \\
0 & A_{12} & 0 & B_{12} & 0 & 0 & 0 \\
0 & 0 & A_{13} & B_{13} & 0 & 0 & 0 \\
A_{21} & 0 & 0 & 0 & B_{21} & 0 & 0 \\
0 & A_{22} & 0 & 0 & B_{22} & 0 & 0 \\
0 & 0 & A_{23} & 0 & B_{23} & 0 & 0 \\
A_{31} & 0 & 0 & 0 & 0 & B_{31} & 0 \\
0 & A_{32} & 0 & 0 & 0 & B_{32} & 0 \\
0 & 0 & A_{33} & 0 & 0 & B_{33} & 0 \\
A_{41} & 0 & 0 & 0 & 0 & 0 & B_{41} \\
0 & A_{42} & 0 & 0 & 0 & 0 & B_{42} \\
0 & 0 & A_{43} & 0 & 0 & 0 & B_{43}
\end{pmatrix}
\]
Typical Jacobian

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Bundle adjustment

\[
\begin{pmatrix}
U_1 & 0 & 0 & W_{11} & W_{21} & W_{31} & W_{41} \\
0 & U_2 & 0 & W_{12} & W_{22} & W_{32} & W_{42} \\
0 & 0 & U_3 & W_{13} & W_{23} & W_{33} & W_{43} \\
W_{11}^T & W_{12}^T & W_{13}^T & V_1 & 0 & 0 & 0 \\
W_{21}^T & W_{22}^T & W_{23}^T & 0 & V_2 & 0 & 0 \\
W_{31}^T & W_{32}^T & W_{33}^T & 0 & 0 & V_3 & 0 \\
W_{41}^T & W_{42}^T & W_{43}^T & 0 & 0 & 0 & V_4
\end{pmatrix}
\begin{pmatrix}
\delta_{a_1} \\
\delta_{a_2} \\
\delta_{a_3} \\
\delta_{b_1} \\
\delta_{b_2} \\
\delta_{b_3} \\
\delta_{b_4}
\end{pmatrix}
=
\begin{pmatrix}
\epsilon_{a_1} \\
\epsilon_{a_2} \\
\epsilon_{a_3} \\
\epsilon_{b_1} \\
\epsilon_{b_2} \\
\epsilon_{b_3} \\
\epsilon_{b_4}
\end{pmatrix}
\]

\[
U^* = \begin{pmatrix}
U_1^* & 0 & 0 \\
0 & U_2^* & 0 \\
0 & 0 & U_3^*
\end{pmatrix},
V^* = \begin{pmatrix}
V_1^* & 0 & 0 & 0 \\
0 & V_2^* & 0 & 0 \\
0 & 0 & V_3^* & 0 \\
0 & 0 & 0 & V_4^*
\end{pmatrix},
W = \begin{pmatrix}
W_{11} & W_{21} & W_{31} & W_{41} \\
W_{12} & W_{22} & W_{32} & W_{42} \\
W_{13} & W_{23} & W_{33} & W_{43}
\end{pmatrix}
\]

\[
\begin{pmatrix}
U^* & W \\
W^T & V^*
\end{pmatrix}
\begin{pmatrix}
\delta_a \\
\delta_b
\end{pmatrix}
=
\begin{pmatrix}
\epsilon_a \\
\epsilon_b
\end{pmatrix}
\]
Block structure of normal equation
Bundle adjustment

Multiplied by

\[
\begin{pmatrix}
I & -W V^{*-1} \\
0 & I
\end{pmatrix}
\]

\[
\begin{pmatrix}
U^* - W V^{*-1} W^T & 0 \\
W^T & V^*
\end{pmatrix}
\begin{pmatrix}
\delta_a \\
\delta_b
\end{pmatrix}
= \begin{pmatrix}
\epsilon_a - W V^{*-1} \epsilon_b \\
\epsilon_b
\end{pmatrix}
\]

\[
(U^* - W V^{*-1} W^T) \delta_a = \epsilon_a - W V^{*-1} \epsilon_b
\]

\[
V^* \delta_b = \epsilon_b - W^T \delta_a
\]
Recognising panoramas

- Parameterise each camera by rotation and focal length

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

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

- This gives pairwise homographies

\[ \tilde{u}_i = \mathbf{H}_{ij}\tilde{u}_j, \quad \mathbf{H}_{ij} = \mathbf{K}_i \mathbf{R}_i \mathbf{R}_j^T \mathbf{K}_j^{-1} \]
Error function

- Sum of squared projection errors

\[ e = \sum_{i=1}^{n} \sum_{j \in I(i)} \sum_{k \in F(i,j)} f(r_{ij}^k)^2 \]

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

- Robust error function

\[ f(x) = \begin{cases} |x|, & \text{if } |x| < x_{\text{max}} \\ x_{\text{max}}, & \text{if } |x| \geq x_{\text{max}} \end{cases} \]
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](http://www.ics.forth.gr/~lourakis/sba).
- You can use this library for your project #2.
Reference