Camera calibration

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

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

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

where $f_i : \mathbb{R}^n \to \mathbb{R}$, $i = 1, \ldots, m$ 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

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

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

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

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

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

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

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

where $\nabla F(x)$ is the gradient, and $\nabla^2 F(x)$ is the Hessian.

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

$$ \mathbf{H} = \nabla^2 F(x) = \begin{bmatrix} \frac{\partial^2 F}{\partial x_i \partial x_j}(x) \end{bmatrix}. $$

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
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

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

Descent direction

$$F(x + \alpha h) = F(x) + \alpha h^T F'(x) + O(\alpha^2)$$

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 $h^T F'(x) < 0$.

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 $\alpha h$ with positive $\alpha$, then the relative gain in function value satisfies

$$\lim_{\alpha \to 0} \frac{F(x) - F(x + \alpha h)}{\alpha \|h\|} = \frac{1}{\|h\|} h^T F'(x) = -\|F'(x)\| \cos \theta,$$

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

$$h_{sd} = -F'(x).$$

(2.8)

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

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

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

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

If \( F''(x) \) is positive definite

\[
h := h_n
\]

else

\[
h := h_{ld} \quad x := x + \alpha h
\]

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 $e^T e$ is minimal. Here, $e = x - \hat{x}$, 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 $\partial f(p) / \partial p$

it is required to find the $\delta_p$ that minimizes the quantity $||x - f(p + \delta_p)|| \approx ||x - f(p) - J\delta_p|| = ||e - J\delta_p||$

$$J^T J \delta_p = J^T e$$

$$N\delta_p = J^T e$$

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

$damping term$

Algorithm:

$k := 0; \nu := 2; p := p_0$

$A := J^T J; \epsilon_p := x - f(p); g := J^T \epsilon_p$

stop := $||g||_\infty \leq \epsilon_1$; $\mu := \tau \times \max(1, \ldots, n(A_0))$

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

$k := k + 1$

repeat

Solve $(A + \mu I)\delta_p = g$

if $||\delta_p|| \leq \epsilon_2 ||p||$

stop := true;

else

$P_{\text{new}} := p + \delta_p$

$\rho := (||p||^2 - ||x - f(p_{\text{new}})||^2) / (||p||^2 (\mu + g))$

if $\rho > 0$

$p := P_{\text{new}}$

$A := J^T J; \epsilon_p := x - f(p); g := J^T \epsilon_p$

stop := $||g||_\infty \leq \epsilon_1$;

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

else

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

endif

endif

until ($p > 0$) or (stop)

endwhile
Camera projection models

Pinhole camera model

Pinhole camera

Camera center (optical center)

principal axis

principal point

image plane

origin

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

\[ y = \frac{fY}{Z} \]

\[
\begin{pmatrix}
X \\
Y \\
Z \\
1
\end{pmatrix} =
\begin{pmatrix}
f & 0 & 0 & 0 \\
0 & f & 0 & 0 \\
0 & 0 & 1 & 0
\end{pmatrix}
\begin{pmatrix}
x \\
y \\
1
\end{pmatrix}
\]
**Pinhole camera model**

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

**Principal point offset**

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

**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

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

**Camera rotation and translation**

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

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

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

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

- 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”: \((x, y, z) \rightarrow (x, y)\)

  \[
  \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)
  
  - Also called “weak perspective”

- Affine projection
  - Also called “paraperspective”
Fun with perspective

Perspective cues

Perspective cues

Fun with perspective

Ames room
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
Linear regression

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

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

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

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/
  - 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

Step 4: camera calibration
**Step 5: refinement**

**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.

- $n$ 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

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

Euclidean distance
Bundle adjustment

Algorithm:

\[ k := 0; \nu := 2; p := p_0; \]
\[ A := JJ; \epsilon_p := x - f(p); g := J^T \epsilon_p; \]
\[ \text{stop} := (\|g\| \leq \epsilon_1); \mu := \tau \max_{i \in 1, \ldots, n}(A_i); \]
while (not stop) and (\(k < k_{\text{max}}\))
\[ k := k + 1; \]
repeat
\[ \text{Solve } (A + \mu I) \delta p = g; \]
if \(\|\delta p\| \leq \epsilon_2\|p\|\)
\[ \text{stop} := \text{true}; \]
else
\[ p_{\text{new}} := p + \delta p; \]
\[ \rho := \|\delta p\|^2 - \|x - f(p_{\text{new}})\|^2) / \delta p^T (\mu \delta p + g); \]
if \(\rho > 0\)
\[ p := p_{\text{new}}; \]
\[ A := JJ; \epsilon_p := x - f(p); g := J^T \epsilon_p; \]
\[ \text{stop} := (\|g\| \leq \epsilon_1); \]
\[ \mu := \mu \max(1, 1 - (2\rho - 1)^3); \nu := 2; \]
else
\[ \mu := \mu \nu; \nu := 2 \nu; \]
endif
endif
until \((\rho > 0)\) or (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
Bundle adjustment

\[
\begin{pmatrix}
U_1 & 0 & 0 & W_{11} & W_{12} & W_{13} & W_{14} \\
0 & U_2 & 0 & W_{21} & W_{22} & W_{23} & W_{24} \\
0 & 0 & U_3 & W_{31} & W_{32} & W_{33} & W_{34} \\
W_{11}^T & W_{12}^T & W_{13}^T & v_1 & 0 & 0 & 0 \\
W_{21}^T & W_{22}^T & W_{23}^T & 0 & v_2 & 0 & 0 \\
W_{31}^T & W_{32}^T & W_{33}^T & 0 & 0 & v_3 & 0 \\
W_{41}^T & W_{42}^T & W_{43}^T & 0 & 0 & 0 & v_4
\end{pmatrix}
\begin{pmatrix}
\delta_a \\
\delta_{b1} \\
\delta_{b2} \\
\delta_{b3} \\
\delta_{b4}
\end{pmatrix}
= 
\begin{pmatrix}
\epsilon_{b1} \\
\epsilon_{b2} \\
\epsilon_{b3} \\
\epsilon_{b4}
\end{pmatrix}
\]

\[
U^* = \begin{pmatrix} U_1^T & 0 & 0 \\ 0 & U_2^T & 0 \\ 0 & 0 & U_3^T \end{pmatrix},
V^* = \begin{pmatrix} v_1^T & 0 & 0 & 0 \\ 0 & v_2^T & 0 & 0 \\ 0 & 0 & v_3^T \\ 0 & 0 & 0 & v_4^T \end{pmatrix},
W = \begin{pmatrix} W_{11} & W_{12} & W_{13} & W_{14} \\ W_{21} & W_{22} & W_{23} & W_{24} \\ W_{31} & W_{32} & W_{33} & W_{34} \\ W_{41} & W_{42} & W_{43} & W_{44} \end{pmatrix}
\]

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

Bundle adjustment

Multiplied by

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

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

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

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

Recognising panoramas

- Parameterise each camera by rotation and focal length

\[
R_i = e^{[\theta_i]_x},\quad [\theta_i]_x = \begin{bmatrix} 0 & -\theta_{i3} & \theta_{i2} \\ \theta_{i3} & 0 & -\theta_{i1} \\ -\theta_{i2} & \theta_{i1} & 0 \end{bmatrix}
\]

\[
K_i = \begin{bmatrix} f_i & 0 & 0 \\ 0 & f_i & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]

- This gives pairwise homographies

\[
\bar{u}_i = H_{ij} \bar{u}_j,\quad H_{ij} = K_i R_i R_j^T K_j^{-1}
\]
Error function

- Sum of squared projection errors
  \[ e = \sum_{i=1}^{n} \sum_{j \in I(i)} \sum_{k \in F(i,j)} f(\epsilon_{ij}^k)^2 \]
  - \( n \) = number of images
  - \( I(i) \) = set of image matches to image \( i \)
  - \( F(i, j) \) = set of feature matches between images \( i, j \)
  - \( \epsilon_{ij}^k \) = residual of \( k \)th feature match between images \( i, j \)

- Robust error function
  \[ f(x) = \begin{cases} 
  |x|, & \text{if } |x| < x_{max} \\
  x_{max}, & \text{if } |x| \geq x_{max} 
  \end{cases} \]

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](http://www.ics.forth.gr/~lourakis/sba).
- You can use this library for your project #2.

MatchMove

Reference