Multi-view 3D Reconstruction for Dummies

Jianxiong Xiao
What will you learn in the following one hour?
SFMedu Program with Code

Download from:
http://mit.edu/jxiao/Public/software/SFMedu/
After this Lecture

You should be able to

– Write your own structure from motion pipeline
– Implement a multiple view stereo system
– Know well about pinhole camera model
– Establish foundation to learn more multiple view geometry theories
– Know how to estimate the parameters for a model using linear system of equations
– Know how to solve non-linear least square problem in practice
How

Reading: the MVG bible
How

Reading: the MVG bible
(need ~2 years)
How

Reading: the MVG bible
(need ~2 years)

Coding
How

Reading: the MVG bible
(need ~2 years)

Coding

Crying: Not Working At All
How

Reading: the MVG bible (need ~2 years)

Coding

Crying: Not Working At All
How

Reading: the MVG bible
(need ~2 years)

Coding

Crying: Not Working At All

PhD?

Yes

PhD?
How

Reading: the MVG bible
(need ~2 years)

Coding

5-6 years

PhD?

Yes

Crying: Not Working At All
# How: a complete new way of learning

<table>
<thead>
<tr>
<th>Standard Way</th>
<th>Our Way</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reading:</strong> the MVG bible (need ~2 years)</td>
<td>5-6 years</td>
</tr>
<tr>
<td><strong>Coding</strong></td>
<td>Yes</td>
</tr>
<tr>
<td><strong>Crying:</strong> Not Working At All</td>
<td>PhD?</td>
</tr>
</tbody>
</table>

Our Way: PhD? Yes
# How: a complete new way of learning

<table>
<thead>
<tr>
<th>Standard Way</th>
<th>Our Way</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Reading:</strong> the MVG bible (need ~2 years)</td>
<td>Run a program</td>
</tr>
<tr>
<td>Coding</td>
<td>See pictures</td>
</tr>
<tr>
<td>Crying: Not Working At All</td>
<td>Listen stories</td>
</tr>
<tr>
<td>PhD</td>
<td>Play with math</td>
</tr>
<tr>
<td><strong>5-6 years</strong></td>
<td><strong>1 hour</strong></td>
</tr>
<tr>
<td>Yes</td>
<td>Dance or sing</td>
</tr>
</tbody>
</table>

$x = PX$
The Basics: My Life Story
My Life as a 3D Point

X

X = [X, Y, Z]
I am small

\[
X = \begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix}
\]
I am sexy

People loves to take picture of me.

\[ X = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix} \]
When they take a picture:

\[
\begin{align*}
\begin{bmatrix}
X \\
Y \\
Z
\end{bmatrix} =
\begin{bmatrix}
x \\
y
\end{bmatrix}
\end{align*}
\]
When they take a picture:

\[
\begin{bmatrix}
  x \\
  y \\
  f
\end{bmatrix}
= \begin{bmatrix}
  X \\
  Y \\
  Z
\end{bmatrix}
\]

\[
= \begin{bmatrix}
  1 & 0 & 0 & 0 \\
  0 & 1 & 0 & 0 \\
  0 & 0 & 1 & 0 \\
  0 & 0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  X \\
  Y \\
  Z \\
  1
\end{bmatrix}
\]

Up to a scale.
Why the image plane is in front?

Image Plane

Camera Center

The World

Image Plane

Camera Center

Image Plane

The World
When they take a picture:
When they take a picture:

Matlab Image Row

Matlab Image Column

Picture
When they take a picture:

Matlab Image Row $r$

Matlab Image Column $C$

$y$

$x$

Picture
When they take a picture:

We let $f$ to take care of this as well. Unit of $f$ is pixel/world unit.

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

$$\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}$$
I live in a real world

Camera Coordinate System

\[
\begin{pmatrix}
X_{\text{cam}} \\
Y_{\text{cam}} \\
Z_{\text{cam}} \\
1
\end{pmatrix} =
\begin{bmatrix}
R & t \\
0^T & 1
\end{bmatrix}
\begin{pmatrix}
X \\
Y \\
Z \\
1
\end{pmatrix}
\]

World Coordinate System

Model-view Transformation \( R, t \)
World Coor. $\rightarrow$ Camera Coor.

$$
\begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix} =
\begin{bmatrix}
    f & 0 & 0 \\
    0 & f & 0 \\
    0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
    1 & 0 & 0 & 0 \\
    0 & 1 & 0 & 0 \\
    0 & 0 & 1 & 0
\end{bmatrix}
\begin{bmatrix}
    R \\
    t \\
    0^T \\
    1
\end{bmatrix}
\begin{bmatrix}
    X \\
    Y \\
    Z \\
    1
\end{bmatrix}
$$

$$
K =
\begin{bmatrix}
    f & 0 & 0 \\
    0 & f & 0 \\
    0 & 0 & 1
\end{bmatrix}
\quad
x =
\begin{bmatrix}
    x \\
    y \\
    1
\end{bmatrix}
\quad
x = K[R|t]X
$$

Camera Parameter
Camera Projection Matrix

$P = K[R|t]$
I am sexy

- When people take a picture of me:

\[ x = K[R|t]X \]
I am very sexy

• When people take two pictures with same camera setting:

\[ x_1 = K[R_1|t_1]X \]

\[ x_2 = K[R_2|t_2]X \]
I am very very sexy

• When people take three pictures with same camera setting:

\[ x_1 = K[R_1|t_1]X \]
\[ x_2 = K[R_2|t_2]X \]
\[ x_3 = K[R_3|t_3]X \]
I am very very sexy

Animation adapted from Noah Snavely
I join “America's Next Top Model”
I join “America's Next Top Model”

Animation adapted from Noah Snavely
I join “America's Next Top Model”

<table>
<thead>
<tr>
<th>Point 1</th>
<th>Point 2</th>
<th>Point 3</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Image 1</strong></td>
<td>$x_1^1 = K[R_1</td>
<td>t_1]X^1$</td>
</tr>
<tr>
<td><strong>Image 2</strong></td>
<td>$x_2^1 = K[R_2</td>
<td>t_2]X^1$</td>
</tr>
<tr>
<td><strong>Image 3</strong></td>
<td>$x_3^1 = K[R_3</td>
<td>t_3]X^1$</td>
</tr>
</tbody>
</table>

Same Camera Same Setting = Same $K$
Triangulation

Animation adapted from Noah Snavely
Triangulation

Image 1
\( R_1, t_1 \)

Image 2
\( R_2, t_2 \)

Image 3
\( R_3, t_3 \)

Animation adapted from Noah Snavely
Other Intrinsic Camera Parameters

• Principle point offset
  – especially when images are cropped (Internet)

• Skew

\[
K = \begin{bmatrix} f & 0 & 0 \\ 0 & f & 0 \\ 0 & 0 & 1 \end{bmatrix}
\]

\[
K = \begin{bmatrix} f_x & s & p_x \\ 0 & f_y & p_y \\ 0 & 0 & 1 \end{bmatrix}
\]

• Radial distortion (due to optics of the lens)

\[
r^2 = \|x\|^2 = x^2 + y^2
\]

\[
x' = (1 + k_1r^2 + k_2r^4)x
\]
Other Camera Models

- Fisheye
- Mirror
- Panorama
- Tilt-Shift Lens
- Biological Eyes

http://www.popgadget.net/2006/07/fisheye_camera.php
http://www.0-360.com/

http://sun360.csail.mit.edu
Canon TS-E 24mm f/3.5L II
What is a camera?

Definition: A camera is a two-parameter linear family of lines—that is, a degenerate regulus (rank-3 family), or a non-degenerate linear congruence (rank-4 family).
White-board Exercise

I brought a normal compact digital camera. I took a picture at resolution 640x480 pixels. What is the possible value of the focal length $f$?
Steps

Images $\rightarrow$ Points: Structure from Motion
Points $\rightarrow$ More points: Multiple View Stereo
Points $\rightarrow$ Meshes: Model Fitting

+ Meshes $\rightarrow$ Models: Texture Mapping

= Images $\rightarrow$ Models: Image-based Modeling
Steps

Images $\rightarrow$ Points: Structure from Motion

Points $\rightarrow$ More points: Multiple View Stereo

Points $\rightarrow$ Meshes: Model Fitting

Meshes $\rightarrow$ Models: Texture Mapping

Images $\rightarrow$ Models: Image-based Modeling
Steps

Images $\rightarrow$ Points: Structure from Motion

Points $\rightarrow$ More points: Multiple View Stereo

Points $\rightarrow$ Meshes: Model Fitting

Meshes $\rightarrow$ Models: Texture Mapping

$=$ Images $\rightarrow$ Models: Image-based Modeling
Steps

Images → Points: Structure from Motion

Points → More points: Multiple View Stereo

Points → Meshes: Model Fitting

Meshes → Models: Texture Mapping

Images → Models: Image-based Modeling
Steps

Images $\rightarrow$ Points:  Structure from Motion

Points $\rightarrow$ More points:  Multiple View Stereo

**Points $\rightarrow$ Meshes:**  Model Fitting

Meshes $\rightarrow$ Models:  Texture Mapping

$\Rightarrow$  Images $\rightarrow$ Models:  Image-based Modeling
Steps

Images $\rightarrow$ Points: Structure from Motion

Points $\rightarrow$ More points: Multiple View Stereo

Points $\rightarrow$ Meshes: Model Fitting

+ Meshes $\rightarrow$ Models: Texture Mapping

= Images $\rightarrow$ Models: Image-based Modeling
Steps

Images $\rightarrow$ Points: Structure from Motion
Points $\rightarrow$ More points: Multiple View Stereo
Points $\rightarrow$ Meshes: Model Fitting
Meshes $\rightarrow$ Models: Texture Mapping

$+$

Meshes $\rightarrow$ Models: Image-based Modeling
Steps

Images $\rightarrow$ Points: Structure from Motion

Points $\rightarrow$ More points: Multiple View Stereo

Points $\rightarrow$ Meshes: Model Fitting

Meshes $\rightarrow$ Models: Texture Mapping

$\Rightarrow$ Images $\rightarrow$ Models: Image-based Modeling
Steps

Images → Points: Structure from Motion
Points → More points: Multiple View Stereo
Points → Meshes: Model Fitting
Meshes → Models: Texture Mapping

= Images → Models: Image-based Modeling
Steps

Images $\rightarrow$ Points: Structure from Motion

Points $\rightarrow$ More points: Multiple View Stereo

Points $\rightarrow$ Meshes: Model Fitting

Meshes $\rightarrow$ Models: Texture Mapping

$\Rightarrow$ Images $\rightarrow$ Models: Image-based Modeling
Structure From Motion

- Structure = 3D Point Cloud of the Scene
- Motion = Camera Location and Orientation
- SFM = Get the Point Cloud from Moving Cameras
- Structure and Motion: Joint Problems to Solve
Pipeline

Structure from Motion (SFM)

Multi-view Stereo (MVS)
Pipeline

Structure from Motion (SFM)

Multi-view Stereo (MVS)
Two-view Reconstruction
Two-view Reconstruction
Two-view Reconstruction

keypoints → match → fundamental matrix → essential matrix → $[R|t]$ → triangulation
Keypoints Detection

- keypoints
- match → fundamental matrix → essential matrix → $[R|t]$ → triangulation
Descriptor for each point

- keypoints
- match
- fundamental matrix
- essential matrix
- $[R|t]$
- triangulation

SIFT descriptor
Same for the other images

keypoints → match → fundamental matrix → essential matrix → \([R|t]\) → triangulation
Point Match for correspondences

keypoints → match → fundamental matrix → essential matrix → $[R|t]$ → triangulation
Point Match for correspondences

- keypoints
- SIFT descriptor
- fundamental matrix
- essential matrix
- \([R|t]\)
- triangulation

(Point Match)

Match keypoints

SIFT descriptor
Fundamental Matrix

\[ x_1 \leftrightarrow x_2 \]

\[ x_1^T F x_2 = 0 \]
Gangnam Style
Gangnam Style
Fundamental Matrix Song
Fundamental Matrix Song

After this song, there will be a quiz. Listen carefully!
Exercise

What is the difference between Fundamental Matrix and Homography? (Both of them are to explain 2D point to 2D point correspondences.)
Estimating Fundamental Matrix

- Given a correspondence
  \[ x_1 \leftrightarrow x_2 \]
- The basic incidence relation is
  \[ x_1^T F x_2 = 0 \]

\[ \begin{bmatrix} f_{11} \\ f_{12} \\ f_{13} \\ f_{21} \\ f_{22} \\ f_{23} \\ f_{31} \\ f_{32} \\ f_{33} \end{bmatrix} = 0 \]

Need 8 points
Estimating Fundamental Matrix

\[ x^T_1 F x_2 = 0 \] for 8 point correspondences:

\[ x_1^1 \leftrightarrow x_2^1, x_1^2 \leftrightarrow x_2^2, x_1^3 \leftrightarrow x_2^3, x_1^4 \leftrightarrow x_2^4, x_1^5 \leftrightarrow x_2^5, x_1^6 \leftrightarrow x_2^6, x_1^7 \leftrightarrow x_2^7, x_1^8 \leftrightarrow x_2^8 \]

\[
\begin{bmatrix}
x_1^1 x_2^1 & x_1^1 y_2^1 & x_1^1 & y_1^1 x_2^1 & y_1^1 y_2^1 & y_1^1 & x_1^1 & y_2^1 & 1 \\
x_1^2 x_2^2 & x_1^2 y_2^2 & x_1^2 & y_1^2 x_2^2 & y_1^2 y_2^2 & y_1^2 & x_1^2 & y_2^2 & 1 \\
x_1^3 x_2^3 & x_1^3 y_2^3 & x_1^3 & y_1^3 x_2^3 & y_1^3 y_2^3 & y_1^3 & x_1^3 & y_2^3 & 1 \\
x_1^4 x_2^4 & x_1^4 y_2^4 & x_1^4 & y_1^4 x_2^4 & y_1^4 y_2^4 & y_1^4 & x_1^4 & y_2^4 & 1 \\
x_1^5 x_2^5 & x_1^5 y_2^5 & x_1^5 & y_1^5 x_2^5 & y_1^5 y_2^5 & y_1^5 & x_1^5 & y_2^5 & 1 \\
x_1^6 x_2^6 & x_1^6 y_2^6 & x_1^6 & y_1^6 x_2^6 & y_1^6 y_2^6 & y_1^6 & x_1^6 & y_2^6 & 1 \\
x_1^7 x_2^7 & x_1^7 y_2^7 & x_1^7 & y_1^7 x_2^7 & y_1^7 y_2^7 & y_1^7 & x_1^7 & y_2^7 & 1 \\
x_1^8 x_2^8 & x_1^8 y_2^8 & x_1^8 & y_1^8 x_2^8 & y_1^8 y_2^8 & y_1^8 & x_1^8 & y_2^8 & 1 \\
\end{bmatrix}
\begin{bmatrix}
f_{11} \\
f_{12} \\
f_{13} \\
f_{21} \\
f_{22} \\
f_{23} \\
f_{31} \\
f_{32} \\
f_{33} \\
\end{bmatrix}
= 0

Ax = b
Af = 0

Direct Linear Transformation (DLT)
Algebraic Error vs. Geometric Error

• Algebraic Error

\[ \min \| A f \| \]

• Geometric Error (better)  Unit: pixel

\[ \min \sum_{j} \left\| x_{1}^{j} - F x_{2}^{j} \right\|^{2} \]

Solved by (non-linear) least square solver (e.g. Ceres)
RANSAC to Estimate Fundamental Matrix

• For many times
  – Pick 8 points
  – Compute a solution for $F$ using these 8 points
  – Count number of inliers that with $x_1^T F x_2$ close to 0
• Pick the one with the largest number of inliers
Minimal problems in Computer Vision

Overview

Minimal problems in computer vision arise when computing geometrical models from image data. They often lead to solving systems of algebraic equations.

This page provides links to publications, software, data, and evaluation of minimal problems.

NEW:
ACCV 2010
Bujnak M., Kukelova Z., Pajdla T., New efficient solution to the absolute pose problem for camera with unknown focal length and radial distortion, ACCV 2010, Queenstown, NZ, November 8-12, 2010. [pdf]
CODE:
4-point absolute pose problem with unknown focal length and radial distortion (P4Pfr)

Kukelova Z., Bujnak M., Pajdla T., Closed-form solutions to the minimal absolute pose problems with known vertical direction, ACCV 2010, Queenstown, NZ, November 8-12, 2010. [pdf]
CODE:
2-point absolute pose problem with known vertical direction (up2p)
3-point absolute pose problem with known vertical direction and unknown focal length and radial distortion (up3pfr)

SOURCE CODES TO SEVERAL MINIMAL PROBLEMS
4-point absolute pose problem with unknown focal length (P4PF) (new fast Matlab version)
8-point "uncalibrated" relative pose problem with radial distortion
4-point absolute pose problem with unknown focal length and radial distortion (P4Pfr)
2-point absolute pose problem with known vertical direction (up2p)
3-point absolute pose problem with known vertical direction and unknown focal length and radial distortion (up3pfr)

Minimal problems:
3-point absolute pose problem (P3P)
4-point absolute pose problem with unknown focal length (P4PF) [NEW] [FAST MATLAB SOURCE CODE]

4-point absolute pose problem with unknown focal length and radial distortion (P4Pfr) [NEW]
2-point absolute pose problem with known vertical direction (up2p) [NEW]
3-point absolute pose problem with known vertical direction and unknown focal length and radial distortion (up3pfr) [NEW]

http://cmp.felk.cvut.cz/minimal/
Fundamental Matrix $\rightarrow$ Essential Matrix

\[ x_1^T F x_2 = 0 \]

\[ E = K_1^T F K_2 \]

Animation adapted from Noah Snavely
Essential Matrix $\rightarrow \begin{bmatrix} \mathbf{R} & \mathbf{t} \end{bmatrix}$

\[
x_1^T \mathbf{Fx}_2 = 0
\]

\[
\mathbf{E} = \mathbf{K}_1^T \mathbf{FK}_2
\]

Animation adapted from Noah Snavely
Essential Matrix $\rightarrow [R|t]$

**Result 9.19.** For a given essential matrix

$$E = U \text{diag}(1,1,0) V^T,$$

and the first camera matrix $P_1 = \begin{bmatrix} I & 0 \end{bmatrix}$, there are four possible choices for the second camera matrix $P_2$:

$$P_2 = \begin{bmatrix} UWV^T & +u_3 \end{bmatrix}$$
$$P_2 = \begin{bmatrix} UWV^T & -u_3 \end{bmatrix}$$
$$P_2 = \begin{bmatrix} UW^TV^T & +u_3 \end{bmatrix}$$
$$P_2 = \begin{bmatrix} UW^TV^T & -u_3 \end{bmatrix}$$

$$W = \begin{bmatrix} 0 & -1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$
Fig. 9.12. The four possible solutions for calibrated reconstruction from E. Between the left and right sides there is a baseline reversal. Between the top and bottom rows camera B rotates 180° about the baseline. Note, only in (a) is the reconstructed point in front of both cameras.

Page 260 of the bible (Multiple View Geometry, 2nd Ed)
Triangulation

Animation adapted from Noah Snavely
In front of the camera?

• Camera Extrinsic $[\mathbf{R} | \mathbf{t}]$

$$
\begin{bmatrix}
X_{\text{cam}} \\
Y_{\text{cam}} \\
Z_{\text{cam}}
\end{bmatrix} = \mathbf{R} \begin{bmatrix}
X_{\text{world}} \\
Y_{\text{world}} \\
Z_{\text{world}}
\end{bmatrix} + \mathbf{t}
$$

$$
\begin{bmatrix}
X_{\text{world}} \\
Y_{\text{world}} \\
Z_{\text{world}}
\end{bmatrix} = \mathbf{R}^{-1} \left( \begin{bmatrix}
X_{\text{cam}} \\
Y_{\text{cam}} \\
Z_{\text{cam}}
\end{bmatrix} - \mathbf{t} \right) = \mathbf{R}^T \begin{bmatrix}
X_{\text{cam}} \\
Y_{\text{cam}} \\
Z_{\text{cam}}
\end{bmatrix} - \mathbf{R}^T \mathbf{t}
$$

• Camera Center

$$
\begin{bmatrix}
X_{\text{cam}} \\
Y_{\text{cam}} \\
Z_{\text{cam}}
\end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}
$$

$$
\mathbf{C} = \begin{bmatrix}
X_{\text{world}} \\
Y_{\text{world}} \\
Z_{\text{world}}
\end{bmatrix} = \mathbf{R}^T \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} - \mathbf{R}^T \mathbf{t} = -\mathbf{R}^T \mathbf{t}
$$

• View Direction

$$
\begin{bmatrix}
0 \\ 0 \\ 1
\end{bmatrix} - \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix}
$$

$$
\left( \mathbf{R}^T \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} - \mathbf{R}^T \mathbf{t} \right) - \mathbf{C} = \left( \mathbf{R}(3,:)^T - \mathbf{R}^T \mathbf{t} \right) - \left( -\mathbf{R}^T \mathbf{t} \right) = \mathbf{R}(3,:)^T
$$
In front of the camera?

- A point $X$
- Direction from camera center to point $X - C$
- Angle Between Two Vectors

$$A \cdot B = \|A\|\|B\| \cos \theta$$

- Angle Between $X - C$ and View Direction
- Just need to test

$$(X - C) \cdot R(3,:)^T > 0?$$
Pick the Solution

With maximal number of points in front of both cameras.

Fig. 9.12. The four possible solutions for calibrated reconstruction from E. Between the left and right sides there is a baseline reversal. Between the top and bottom rows camera B rotates 180° about the baseline. Note, only in (a) is the reconstructed point in front of both cameras.
Two-view Reconstruction

- keypoints
- match
- fundamental matrix
- essential matrix
- $[R|t]$
- triangulation
Pipeline

Structure from Motion (SFM)

Multi-view Stereo (MVS)
Pipeline

Taught

Next
Merge Two Point Cloud
Merge Two Point Cloud

There can be only one $\begin{bmatrix} R_2 & t_2 \end{bmatrix}$
Merge Two Point Cloud

• From the 1\textsuperscript{st} and 2\textsuperscript{nd} images, we have
  \[ [R_1 | t_1] \text{ and } [R_2 | t_2] \]

• From the 2\textsuperscript{nd} and 3\textsuperscript{rd} images, we have
  \[ [R_2 | t_2] \text{ and } [R_3 | t_3] \]

• Exercise: How to transform the coordinate system of the second point cloud to align with the first point cloud so that there is only one
  \[ [R_2 | t_2] \]?
Merge Two Point Cloud
Oops

See From a Different Angle
Bundle Adjustment
"America's Next Top Model"

Animation adapted from Noah Snavely
“America's Next Top Model”

<table>
<thead>
<tr>
<th></th>
<th>Point 1</th>
<th>Point 2</th>
<th>Point 3</th>
</tr>
</thead>
<tbody>
<tr>
<td>Image 1</td>
<td>(x_1^1 = K[R_1</td>
<td>t_1]X^1)</td>
<td>(x_1^2 = K[R_1</td>
</tr>
<tr>
<td>Image 2</td>
<td>(x_2^1 = K[R_2</td>
<td>t_2]X^1)</td>
<td>(x_2^2 = K[R_2</td>
</tr>
<tr>
<td>Image 3</td>
<td>(x_3^1 = K[R_3</td>
<td>t_3]X^1)</td>
<td></td>
</tr>
</tbody>
</table>
Rethinking the SFM problem

• Input: Observed 2D image position

\[
\tilde{x}_1^1 \quad \tilde{x}_1^2 \\
\tilde{x}_2^1 \quad \tilde{x}_2^2 \quad \tilde{x}_2^3 \\
\tilde{x}_3^1 \quad \tilde{x}_3^3
\]

• Output:

Unknown Camera Parameters (with some guess)

\[
[R_1 | t_1], [R_2 | t_2], [R_3 | t_3]
\]

Unknown Point 3D coordinate (with some guess)

\[
X^1, X^2, X^3, \ldots
\]
Bundle Adjustment

A valid solution $[\mathbf{R}_1 | t_1], [\mathbf{R}_2 | t_2], [\mathbf{R}_3 | t_3]$ and $\mathbf{X}^1, \mathbf{X}^2, \mathbf{X}^3, \ldots$ must let

Re-projection

\[
\begin{aligned}
\mathbf{x}_1^1 &= K [\mathbf{R}_1 | t_1] \mathbf{X}^1 \\
\mathbf{x}_2^1 &= K [\mathbf{R}_2 | t_2] \mathbf{X}^1 \\
\mathbf{x}_3^1 &= K [\mathbf{R}_3 | t_3] \mathbf{X}^1 \\
\mathbf{x}_1^2 &= K [\mathbf{R}_1 | t_1] \mathbf{X}^2 \\
\mathbf{x}_2^2 &= K [\mathbf{R}_2 | t_2] \mathbf{X}^2 \\
\mathbf{x}_3^2 &= K [\mathbf{R}_3 | t_3] \mathbf{X}^3 \\
\mathbf{x}_1^3 &= K [\mathbf{R}_1 | t_1] \mathbf{X}^3 \\
\mathbf{x}_2^3 &= K [\mathbf{R}_2 | t_2] \mathbf{X}^3 \\
\mathbf{x}_3^3 &= K [\mathbf{R}_3 | t_3] \mathbf{X}^3
\end{aligned}
\]

\[
\begin{aligned}
\mathbf{X}_{1}^{\hat{}} &= \hat{\mathbf{x}}_1^1 \\
\mathbf{X}_{2}^{\hat{}} &= \hat{\mathbf{x}}_1^2 \\
\mathbf{X}_{3}^{\hat{}} &= \hat{\mathbf{x}}_1^3
\end{aligned}
\]

Observation
Bundle Adjustment

A valid solution \([R_1|t_1],[R_2|t_2],[R_3|t_3]\) and \(X^1,X^2,X^3,...\) must let the Re-projection close to the Observation, i.e. to minimize the reprojection error

\[
\min \sum \sum (\hat{x}_i^j - K [R_i|t_i] X^j)^2
\]
Bundle Adjustment

A valid solution \([R_1|t_1],[R_2|t_2],[R_3|t_3]\) and \(X^1,X^2,X^3,\ldots\) must let the Re-projection close to the Observation, i.e. to minimize the reprojection error

\[
\min \sum \sum (\tilde{x}_i^j - K [R_i|t_i] X^j)^2
\]

Question: What is the unit of this objective function?
Bundle Adjustment

A valid solution \([ R_1 \mid t_1 ], [ R_2 \mid t_2 ], [ R_3 \mid t_3 ]\) and \( X_1, X_2, X_3, \ldots \) must let the Re-projection close to the Observation, i.e. to minimize the reprojection error

\[
\min \sum_i \sum_j \left( \tilde{x}_i^j - K [ R_i \mid t_i ] X^j \right)^2
\]
Linking

SIFT Matching

SIFT Matching

observation matrix
Solving This Optimization Problem

• Theory:
  The Levenberg–Marquardt algorithm
  http://en.wikipedia.org/wiki/Levenberg-Marquardt_algorithm

• Practice:
  The Ceres-Solver from Google
  http://code.google.com/p/ceres-solver/
Ceres-solver: A Nonlinear Least Squares Minimizer

Toy problem to solve $\min((10 - x)^2)$

class SimpleCostFunction
  : public ceres::SizedCostFunction<1 /* number of residuals */,
    1 /* size of first parameter */>
{
public:
  virtual ~SimpleCostFunction() {}  
  virtual bool Evaluate(double const* const* parameters,
                        double* residuals,
                        double** jacobians) const {
    const double x = parameters[0][0];
    residuals[0] = 10 - x;  // $f(x) = 10 - x$
  // Compute the Jacobian if asked for.
  if (jacobians != NULL && jacobians[0] != NULL) {
    jacobians[0][0] = -1;
  }
  return true;
}
Ceres-solver: A Nonlinear Least Squares Minimizer

Toy problem to solve \( \min(10 - x)^2 \)

```c
int main(int argc, char** argv) {
    double x = 5.0;
    ceres::Problem problem;

    // The problem object takes ownership of the newly allocated
    // SimpleCostFunction and uses it to optimize the value of x.
    problem.AddResidualBlock(new SimpleCostFunction, NULL, &x);

    // Run the solver!
    Solver::Options options;
    options.max_num_iterations = 10;
    options.linear_solver_type = ceres::DENSE_QR;
    options.minimizer_progress_to_stdout = true;
    Solver::Summary summary;
    Solve(options, &problem, &summary);
    std::cout << summary.BriefReport() << "\n";
    std::cout << "x : 5.0 -> " << x << "\n";
    return 0;
}
```
Ceres-solver: A Nonlinear Least Squares Minimizer

Toy problem to solve \( \min(10 - x)^2 \)

Ceres Solver Report: Iterations: 2, Initial cost: 1.250000e+01,  
Final cost: 1.388518e-16, Termination: PARAMETER_TOLERANCE.  
x : 5, -> 10
Non-linear Least Square Cuboid Reconstruction

Observation:
Pixel location of the 7 (or 6) corners

Parameters to be estimated:
Cuboid: Height $h \times$ Width $w \times 1$
Camera: Intrinsic $K$ and Extrinsic $R, t$

$$\min \sum_{i} \left\| x_i - K(RX_i + t) \right\|^2$$

$$X = \begin{bmatrix} h & h & h & h & -h & -h & -h & -h \\ w & w & -w & -w & w & w & -w & -w \\ 1 & -1 & 1 & -1 & 1 & -1 & 1 & -1 \end{bmatrix}$$
Parameterizing Rotation Matrix

• 2D Rotation

\[
\begin{bmatrix}
    x' \\
    y'
\end{bmatrix} =
\begin{bmatrix}
    \cos(\theta) & -\sin(\theta) \\
    \sin(\theta) & \cos(\theta)
\end{bmatrix}
\begin{bmatrix}
    x \\
    y
\end{bmatrix}
\]

\[R^T = R^{-1}, \det R = 1\]
3D Rotation

Yaw, pitch and roll are $\alpha, \beta, \gamma$

$$\mathbf{R}_x(\theta) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$$

$$\mathbf{R}_y(\theta) = \begin{bmatrix} \cos(\theta) & 0 & \sin(\theta) \\ 0 & 1 & 0 \\ -\sin(\theta) & 0 & \cos(\theta) \end{bmatrix}$$

$$\mathbf{R}_z(\theta) = \begin{bmatrix} \cos(\theta) & -\sin(\theta) & 0 \\ \sin(\theta) & \cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Euler angles are $\alpha, \beta, \gamma$

$$\mathbf{R}_x(\gamma)\mathbf{R}_y(\beta)\mathbf{R}_z(\alpha)$$

$$\mathbf{R}_z(\gamma)\mathbf{R}_x(\beta)\mathbf{R}_y(\alpha)$$

http://en.wikipedia.org/wiki/Rotation_matrix
3D Rotation

Axis-angle representation

Quaternions

\[ q = \left( v \sin \left( \frac{\theta}{2} \right), \cos \left( \frac{\theta}{2} \right) \right)^T \]

Avoid Gimbal Lock!

Triplet Representation \( v\theta \) (3 dof) \( \leftrightarrow \) Not over-parameterized

Rodrigues' rotation formula

\[ k_{rot} = k \cos \theta + (v \times k) \sin \theta + v (v \cdot k) (1 - \cos \theta) \]

http://en.wikipedia.org/wiki/Axis-angle_representation
http://en.wikipedia.org/wiki/Quaternions_and_spatial_rotation
Matlab Symbolic Operation

How to derive complicated equations (after passing math classes)?

```
syms h, w, K, alpha, beta, gamma, tx, ty, tz
X = [h; w; f];
Y = [...] * X;
Z = K* Y;
x = Z(1:2,:)./Z([3,3],:);
code(x,'file','x.cpp'); % ←matlab will generate C code for x
```

http://www.mathworks.com/help/symbolic/sym.html
http://mit.edu/jxiao/Public/software/fitCuboid/fitCuboid/derive4cuboid.m
Ceres for Bundle Adjustment

A valid solution $[\mathbf{R}_1 | \mathbf{t}_1], [\mathbf{R}_2 | \mathbf{t}_2], [\mathbf{R}_3 | \mathbf{t}_3]$ and $\mathbf{X}^1, \mathbf{X}^2, \mathbf{X}^3, \ldots$ must let the Re-projection close to the Observation, i.e. to minimize the reprojection error

$$\min \sum_i \sum_j \left( \tilde{x}_i^j - K [\mathbf{R}_i | \mathbf{t}_i] \mathbf{X}_j \right)^2$$
```c++
struct AlignmentError2D {
    AlignmentError2D(double* observed_in): observed(observed_in) {}
}

template <typename T>
bool operator()(const T* const camera_extrinsic,
                const T* const point,
                T* residuals) const {

    T p[3];
    // camera_extrinsic[0,1,2] are the angle-axis rotation.
    ceres::AngleAxisRotatePoint(camera_extrinsic, point, p);
    // camera_extrinsic[3,4,5] are the translation.
    p[0] += camera_extrinsic[3];
    p[1] += camera_extrinsic[4];
    p[2] += camera_extrinsic[5];

    // let p[2] == 0
    if (T(0.0) <= p[2]){
        if (p[2] < EPS){
            p[2] = EPS;
        }
    }else{
        if (p[2] > -EPS){
        }
    }

    // project it
    p[0] = T(fx) * p[0] / p[2] + T(px);
    // reprojection error
    residuals[0] = p[0] - T(observed[0]);
    return true;
}

double* observed;
};
```

\[ \mathbf{RX} + \mathbf{t} = \begin{bmatrix} \mathbf{R} | \mathbf{t} \end{bmatrix} \mathbf{X} \]

To avoid divide by 0

\[ \mathbf{x} = \mathbf{K} \begin{bmatrix} \mathbf{R} | \mathbf{t} \end{bmatrix} \mathbf{X} \]

\[ \mathbf{X} - \tilde{\mathbf{X}} \]
Initialization Matters

• Input: Observed 2D image position

• Output:

Unknown Camera Parameters (with some guess)

\[
\begin{bmatrix}
\mathbf{R}_1 | \mathbf{t}_1 \\
\mathbf{R}_2 | \mathbf{t}_2 \\
\mathbf{R}_3 | \mathbf{t}_3 
\end{bmatrix}
\]

Unknown Point 3D coordinate (with some guess)

\[\mathbf{X}^1, \mathbf{X}^2, \mathbf{X}^3, \cdots\]
Multiple View Stereo

State-of-the-art:
PMVS: http://grail.cs.washington.edu/software/pmvs/
Accurate, Dense, and Robust Multi-View Stereopsis, Y Furukawa and J Ponce, 2007.

Benchmark:
http://vision.middlebury.edu/mview/
A Comparison and Evaluation of Multi-View Stereo Reconstruction Algorithms.
SM Seitz, B Curless, J Diebel, D Scharstein, R Szeliski. 2006.

Baseline:
Key idea: Matching Propagation


In another context:

Simplest Matching Propagation

We are going to learn a very simple algorithm that is implemented in the SFMedu program.
Descriptor: ZNCC (Zero-mean Normalized Cross-Correlation)

- Invariant to linear radiometric changes
- More conservative than others such as sum of absolute or square differences in uniform regions
- More tolerant in textured areas where noise might be important

\[
ZNCC(x_1, x_2) = \frac{\sum_i (I(x_1 + i) - \bar{I}(x_1))(I(x_2 + i) - \bar{I}(x_2))}{\sqrt{\sum_i (I(x_1 + i) - \bar{I}(x_1))^2 \sum_i (I(x_2 + i) - \bar{I}(x_2))^2}}
\]
Descriptor: ZNCC (Zero-mean Normalized Cross-Correlation)

- Invariant to linear radiometric changes
- More conservative than others such as sum of absolute or square differences in uniform regions
- More tolerant in textured areas where noise might be important

\[
ZNCC(x_1, x_2) = \frac{\sum_i (I(x_1 + i) - \bar{I}(x_1))(I(x_2 + i) - \bar{I}(x_2))}{\sqrt{\sum_i (I(x_1 + i) - \bar{I}(x_1))^2 \sum_i (I(x_2 + i) - \bar{I}(x_2))^2}}
\]
Seed for propagation
Matching Propagation (propagate.m)

- Maintain a priority queue Q
- Initialize: Put all seeds into Q with their ZNCC values as scores
- For each iteration:
  - Pop the match with best ZNCC score from Q
  - Add new potential matches in their immediate spatial neighborhood into Q
- Safety: handle uniqueness, and propagate only on matchable area
Matchable Area

the area with maximal gradience > threshold
Result (denseMath/run.m)
Another Example
Another Example
Final Result
Colorize the Point Cloud
Wait: How to get the focal length?

• Auto-calibration
  http://mit.edu/jxiao/Public/software/autocalibrate/autocalibration_lin.m

• Grid Search to look for the solution with minimal reprojection error
  for f=\text{min}_f:\text{max}_f
    do everything, then obtain reprojection error after bundle adjustment

• Optimize for this value in bundle adjustment

• Camera Calibration (with checkerboard)
  http://www.vision.caltech.edu/bouguetj/calib_doc/

• EXIF of JPEG file recorded from digital camera
  Read the code of Bundler to understand how to convert EXIF into focal length value
  http://phototour.cs.washington.edu/bundler/
Steps

Images $\rightarrow$ Points: Structure from Motion

Points $\rightarrow$ More points: Multiple View Stereo

Points $\rightarrow$ Meshes: Model Fitting

Meshes $\rightarrow$ Models: Texture Mapping

$+$

Images $\rightarrow$ Models: Image-based Modeling
<table>
<thead>
<tr>
<th>Steps</th>
</tr>
</thead>
<tbody>
<tr>
<td>Images $\rightarrow$ Points:</td>
</tr>
<tr>
<td>Structure from Motion</td>
</tr>
<tr>
<td>Points $\rightarrow$ More points:</td>
</tr>
<tr>
<td>Multiple View Stereo</td>
</tr>
<tr>
<td>Points $\rightarrow$ Meshes:</td>
</tr>
<tr>
<td>Model Fitting</td>
</tr>
<tr>
<td>Meshes $\rightarrow$ Models:</td>
</tr>
<tr>
<td>Texture Mapping</td>
</tr>
<tr>
<td>=</td>
</tr>
<tr>
<td>Images $\rightarrow$ Models:</td>
</tr>
<tr>
<td>Image-based Modeling</td>
</tr>
</tbody>
</table>
Real World Applications

- Streetview Reconstruction and Recognition
  
  [http://people.csail.mit.edu/jxiao/StreetSeg/](http://people.csail.mit.edu/jxiao/StreetSeg/)


- Microsoft Photosynth  [http://photosynth.net/](http://photosynth.net/)

- 2d3, boujor (Matchmovers) and movies
  


Simultaneous Localization And Mapping
Steps

Images → Points: Structure from Motion

Points → More points: Multiple View Stereo

Points → Meshes: Model Fitting

Meshes → Models: Texture Mapping

Images → Models: Image-based Modeling
Point Cloud $\rightarrow$ 3D Mesh Model

- **Surface Reconstruction**

- **Model Fitting**
  - InverseCSG
InverseCSG Algorithm

InverseCSG Algorithm
Noisy Points

top-down view of input points
InverseCSG Algorithm

Constructive Solid Geometry (CSG)
InverseCSG Algorithm

Constructive Solid Geometry (CSG)

Cuboids as primitives

Xiao et al. NIPS 2012
Xiao et al. Siggraph Asia 2012a
Bottom-up & Top-down

model

3D cuboid

2D rectangle

2D line

point
System Overview

3D point cloud → 2D CSG (floorplan) → 3D CSG model → Wall model → Final textured model
System Overview
Cut into Slices

3D point cloud → 2D CSG (floorplan) → 3D CSG model → Wall model → Final textured model

side view
Cut into Slices

3D point cloud → 2D CSG (floorplan) → 3D CSG model → Wall model → Final textured model

point count
Cut into Slices
2D CSG Reconstruction

1. Generate primitives

2. Choose a subset
2D CSG Reconstruction

1. Generate primitives

point $\rightarrow$ line
2D CSG Reconstruction

1. Generate primitives

  point $\rightarrow$ line
  line $\rightarrow$ rectangle

From 4 line segments
2D CSG Reconstruction

1. Generate primitives

- point $\rightarrow$ line
- line $\rightarrow$ rectangle

From 3 line segments
2D CSG Reconstruction

1. Generate primitives
2D CSG Reconstruction

1. Generate primitives

2. Choose a subset
2D CSG Reconstruction

1. Generate primitives

2. Choose a subset

Repeat in each slice
2D CSG Reconstruction

Explain the data
- Free space
- Laser points

Simple
- Regularization
Objective Function

\[
E_1(T) = \frac{\text{Sum of free-space scores inside } T}{\text{Total sum in the domain without negative scores}}
\]

\[
E_2(T) = \frac{\# \text{ of points on the surface of } T}{\text{total # of points}}
\]

\[
E_3(T) = \frac{\text{perimeter of } T \text{ near laser points (within 0.2 meters)}}{\text{total perimeter of } T}
\]

\[
E(T) = w_1E_1(T) + w_2E_2(T) + w_3E_3(T)
\]
2D CSG Reconstruction

1. Generate primitives

2. Choose a subset
3D CSG Reconstruction

1. Generate primitives (cuboids)

2. Choose a subset (out of primitive candidates)
3D CSG Reconstruction

1. Generate primitives (cuboids)
3D CSG Reconstruction

1. Generate primitives (cuboids)

Rectangle primitive

2D CSG

5D point cloud → 2D CSG (floorplan) → 3D CSG model → Wall model → Final textured model
3D CSG Reconstruction

1. Generate primitives (cuboids)

Diagram showing the process of 3D CSG reconstruction from a 3D point cloud to a final textured model.
3D CSG Reconstruction

1. Generate primitives (cuboids)

Rectangle primitive
3D CSG Reconstruction

1. Generate primitives (cuboids)

2. Choose a subset
Step-by-step visualization of 3D CSG model reconstruction
Result

The Frick Collection

The State Tretyakov Gallery

Uffizi Gallery
Steps

Images → Points: Structure from Motion

Points → More points: Multiple View Stereo

Points → Meshes: Model Fitting

Meshes → Models: Texture Mapping

Images → Models: Image-based Modeling
Texture Mapping

• Texture Stitching
• View-based Texture Mapping
• View Interpolation and Warping
• Interactive Visualization
Image Warping: Forward Warping

Image Warping: Backward Warping

Texture Stitching

The process of assembling projected images to form a composite rendering

Image from Paul Debevec’s Thesis
Texture Stitching

Texture Stitching

View-Dependent Texture Mapping

Modeling and Rendering Architecture from Photographs,
View Interpolation

Match Propagation from Image-based Modeling and Rendering
M Lhuillier and L Quan, 2012
View Interpolation
View Interpolation

Match Propagation from Image-based Modeling and Rendering, M Lhuillier and L Quan, 2012
View Interpolation

Match Propagation from Image-based Modeling and Rendering, M Lhuillier and L Quan, 2012
Interactive Visualization

Gallery: 492 images - 2 Mpixels

Reconstructing Building Interiors from Images, Y Furukawa, B Curless, SM Seitz, R Szeliski, 2009
Download the Viewer from: http://grail.cs.washington.edu/projects/interior/
Bird’s-eye View Indoor Maps

Aerial+Ground Visualization

<table>
<thead>
<tr>
<th></th>
<th>Ground</th>
<th>Aerial</th>
<th>Ground + Aerial</th>
</tr>
</thead>
<tbody>
<tr>
<td>Outdoor</td>
<td>Google Streetview</td>
<td>Google/Bing/NASA ...</td>
<td>Google MapsGL</td>
</tr>
<tr>
<td>Indoor</td>
<td>Furukawa et al.</td>
<td>Xiao et al.</td>
<td>Xiao et al.</td>
</tr>
</tbody>
</table>
Finish our Journey!

Images $\rightarrow$ Points: Structure from Motion

Points $\rightarrow$ More points: Multiple View Stereo

Points $\rightarrow$ Meshes: Model Fitting

Meshes $\rightarrow$ Models: Texture Mapping

$\Rightarrow$ Images $\rightarrow$ Models: Image-based Modeling
History of 3D Reconstruction
(an 1-min over-simplified view)
50 years of 3D reconstruction

1960s  Problem Definition
1970s  Image Formulation
1980s  Geometry
1990s  Reconstruction
2000s  Visualization
2010s  Visualization

50 years of 3D reconstruction

1960s  Problem Definition
1970s  Image Formulation
1980s  Geometry
1990s  Reconstruction
2000s  Visualization
2010s

Shape from Shading, Horn, MIT AI Memos 232, 1970.
50 years of 3D reconstruction

- **1960s**: Problem Definition
- **1970s**: Image Formulation
- **1980s**: Geometry
- **1990s**: Reconstruction
- **2000s**: Visualization
- **2010s**: Visualization

**Essential Matrix**

\[ X_\mu Q_{\mu\nu} X_\nu = 0 \]  \hspace{1cm} (11)

Dividing equation (11) by \( X_3 X_3 \) we arrive at the desired relationship between the image coordinates:

\[ x_\mu Q_{\mu\nu} x_\nu = 0 \]  \hspace{1cm} (12)

50 years of 3D reconstruction

1960s  Problem Definition
1970s  Image Formulation
1980s  Geometry
1990s  Reconstruction
2000s  Visualization
2010s

Structure from Motion  Multi-view Stereo
Pollefeys et al  Furukawa & Ponce

3D Model Fitting
Xiao et al
50 years of 3D reconstruction

- 1960s: Problem Definition
- 1970s: Image Formulation
- 1980s: Geometry
- 1990s: Reconstruction
- 2000s: Visualization
- 2010s: Visualization

Snavely et al
Furukawa et al
Xiao et al
50 years of 3D reconstruction

1960s | Problem Definition
1970s | Image Formulation
1980s | Geometry
1990s | Reconstruction
2000s | Visualization
2010s

Goals of Computer Vision:
• Let machines see
• Let humans see better
Longer Summary of the History

Research Landmarks for 3D Reconstruction

Steve Seitz

"History of 3D Computer Vision"

NSF Frontiers in Computer Vision, 2011

http://www.youtube.com/watch?v=kylzMr917Rc

Homework: Play with SFMedu

- Run the SFMedu program
- Debug the code step by step in Matlab
- Link the code and the theory we cover in this lecture
- Try some other images
- Locate the unstable part of the algorithm
Final Project Ideas: Improve SFMedu

• 5-point algorithm
• Estimate focal length on the fly
• Beyond pairwise neighboring matching
• Tri-focal tensor
• Make it stable for different images input
• Make it work for a video input
• Make it work faster
• Check all the theories and fix all bugs 😊
Q & A