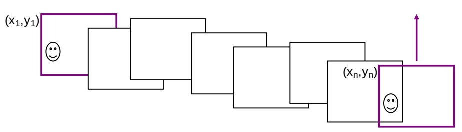
Global Alignment and Structure from Motion

Computer Vision
CSE455, Winter 2008
Noah Snavely

Readings

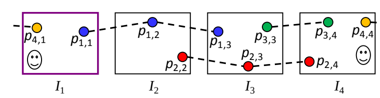
- Snavely, Seitz, Szeliski, **Photo Tourism: Exploring Photo Collections in 3D**. *SIGGRAPH 2006*.
  http://phototour.cs.washington.edu/Photo_Tourism.pdf
- Supplementary reading:
  Szeliski and Kang. **Recovering 3D shape and motion from image streams using non-linear least squares**. *J. Visual Communication and Image Representation, 1993*.
  http://hpl.hp.com/techreports/Compaq-DEC/CRL-93-3.pdf

Problem: Drift



- add another copy of first image at the end
- this gives a constraint: $y_n = y_1$
- there are a bunch of ways to solve this problem
  - add displacement of $(y_1 - y_n)/(n - 1)$ to each image after the first
  - compute a global warp: $y' = y + ax$
  - run a big optimization problem, incorporating this constraint
    - best solution, but more complicated
    - known as "bundle adjustment"

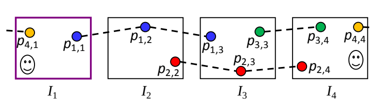
Global optimization



- Minimize a global energy function:
  - What are the variables?
    - The translation $t_j = (x_j, y_j)$ for each image
  - What is the objective function?
    - We have a set of matched features $p_{i,j} = (u_{i,j}, v_{i,j})$
    - For each point match $(p_{i,j}, p_{i,j+1})$:

      $p_{i,j+1} - p_{i,j} = t_{j+1} - t_j$

Global optimization



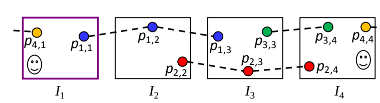
$$
\begin{aligned}
p_{1,2} - p_{1,1} &= t_2 - t_1 \\
p_{1,3} - p_{1,2} &= t_3 - t_2 \\
p_{2,3} - p_{2,2} &= t_3 - t_2 \\
&\dots \\
v_{4,1} - v_{4,4} &= y_1 - y_4
\end{aligned}
$$

$\Rightarrow$ minimize

w_{ij} = 1 if feature i is visible in image j
0 otherwise

$$\sum_{i=1}^{m}\sum_{j=1}^{n-1} w_{ij} \cdot \left\| (p_{i,j+1} - p_{i,j}) - (t_{j+1} - t_j) \right\|^2 + \sum_{i=1}^{m} w_{ij} \cdot \left\| (v_{i,n} - v_{i,1}) - (y_n - y_1) \right\|^2$$

Global optimization



$$
\begin{bmatrix}
-1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\
 & & & \dots & & & & \\
0 & 1 & 0 & 0 & 0 & 0 & 0 & -1
\end{bmatrix}
\begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \\ x_4 \\ y_4 \end{bmatrix}
=
\begin{bmatrix} u_{1,2} - u_{1,1} \\ v_{1,2} - v_{1,1} \\ \vdots \\ v_{4,1} - v_{4,4} \end{bmatrix}
$$

A (2m x 2n) **x** (2n x 1) **b** (2m x 1)

Global optimization

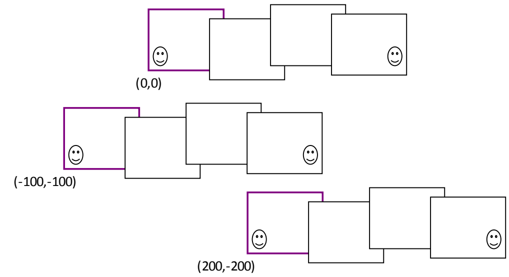
$$\begin{bmatrix} -1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 0 & 1 & 0 & 0 \\ & & & \cdots & & & & \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & -1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ x_2 \\ y_2 \\ x_3 \\ y_3 \\ x_4 \\ y_4 \end{bmatrix} = \begin{bmatrix} u_{1,2}-u_{1,1} \\ v_{1,2}-v_{1,1} \\ \vdots \\ v_{4,1}-v_{4,4} \end{bmatrix}$$

$$\underset{2m \times 2n}{\mathbf{A}} \qquad \underset{2n \times 1}{\mathbf{x}} \qquad \underset{2m \times 1}{\mathbf{b}}$$

Defines a least squares problem: minimize $\|\mathbf{Ax} - \mathbf{b}\|$

- Solution: $\hat{\mathbf{x}} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$
- Problem: there is no unique solution for $\hat{\mathbf{x}}$! ($\det(\mathbf{A}^T\mathbf{A}) = 0$)
- We can add a global offset to a solution $\hat{\mathbf{x}}$ and get the same error

Ambiguity in global location

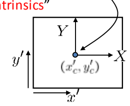


- Each of these solutions has the same error
- Called the *gauge ambiguity*
- Solution: fix the position of one image (e.g., make the origin of the 1st image (0,0))

Solving for camera parameters

Recap: a camera is described by several parameters

- Translation $\mathbf{t}$ of the optical center from the origin of world coords
- Rotation $\mathbf{R}$ of the image plane
- focal length f, principle point (x'_c, y'_c), pixel size (s_x, s_y)
- blue parameters are called "extrinsics," red are "intrinsics"

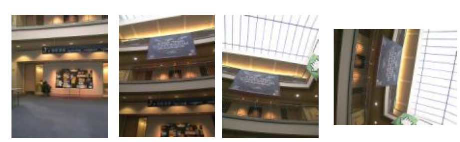


Projection equation

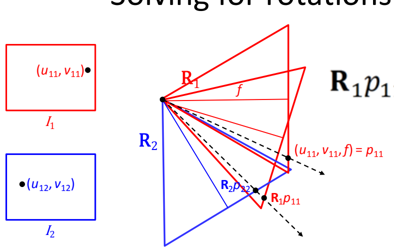
$$p = \begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \underbrace{\begin{bmatrix} -fs_x & 0 & x'_c \\ 0 & -fs_y & y'_c \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \left(\mathbf{R}_{3x3}\mathbf{x} + \mathbf{t}_{3x1}\right)$$

Solving for camera rotation

- Instead of spherically warping the images and solving for translation, we can directly solve for the rotation $\mathbf{R}_j$ of each camera
- Can handle tilt / twist

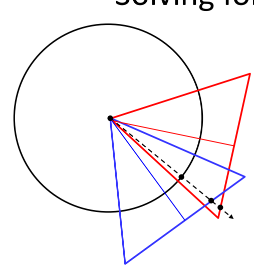


Solving for rotations



$$\mathbf{R}_1 p_{11} \cong \mathbf{R}_2 p_{12}$$

Solving for rotations



$$\mathbf{R}_1 p_{11} \cong \mathbf{R}_2 p_{12}$$

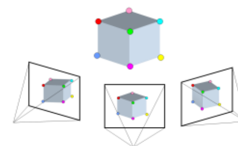
$$\mathbf{R}_1 \hat{p}_{11} = \mathbf{R}_2 \hat{p}_{12}$$

$$\text{minimize} \sum_{i=1}^{m} \sum_{j=1}^{n} w_{ij} \cdot \left\| \mathbf{R}_{j+1} \hat{p}_{i,j+1} - \mathbf{R}_j \hat{p}_{i,j} \right\|^2$$

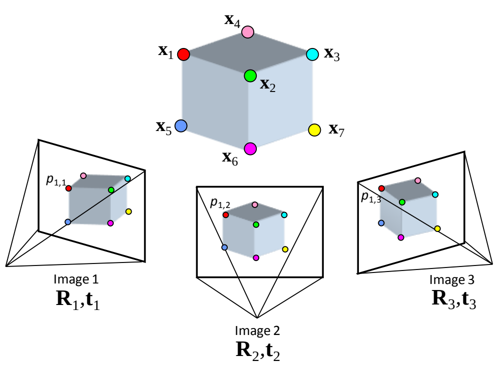
3D rotations

- How many degrees of freedom are there?
- How do we represent?
  - Rotation matrix (too many degrees of freedom)
  - Euler angles (e.g. yaw, pitch, and roll)
  - Quaternions (4-vector on unit sphere)
- Usually involves non-linear optimization

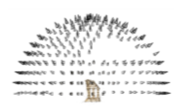
Solving for rotations and translations



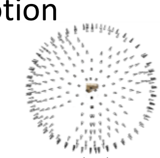
- *Structure from motion (SfM)*
- Unlike with panoramas, we often need to solve for *structure* (3D point positions) as well as *motion* (camera parameters)



Structure from motion

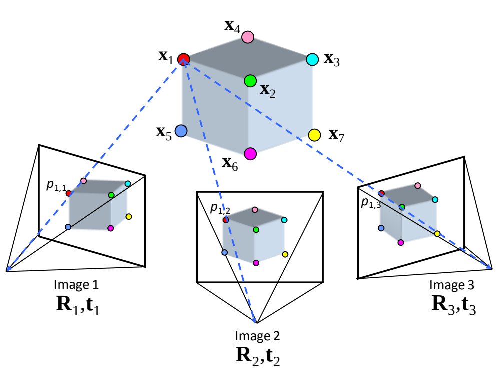


Reconstruction (side)



(top)

- Input: images with points in correspondence
  $p_{i,j} = (u_{i,j}, v_{i,j})$
- Output
  - structure: 3D location $\mathbf{x}_i$ for each point p_i
  - motion: camera parameters $\mathbf{R}_j$, $\mathbf{t}_j$
- Objective function: minimize *reprojection error*



SfM objective function

- Given point **x** and rotation and translation **R**, **t**

$$\begin{bmatrix}x'\\y'\\z'\end{bmatrix} = \mathbf{R}\mathbf{x}+\mathbf{t} \qquad u' = \frac{fx'}{z'} \quad v' = \frac{fy'}{z'} \qquad \begin{bmatrix}u'\\v'\end{bmatrix} = \mathbf{P}(\mathbf{x},\mathbf{R},\mathbf{t})$$

- Minimize sum of squared reprojection errors:

$$g(\mathbf{X},\mathbf{R},\mathbf{T}) = \sum_{i=1}^{m}\sum_{j=1}^{n} w_{ij} \cdot \Bigg\| \underbrace{\mathbf{P}(\mathbf{x}_i,\mathbf{R}_j,\mathbf{t}_j)}_{\substack{\textit{predicted}\\ \text{image location}}} - \underbrace{\begin{bmatrix}u_{i,j}\\v_{i,j}\end{bmatrix}}_{\substack{\textit{observed}\\ \text{image location}}} \Bigg\|^2$$

Solving structure from motion

- Minimizing *g* is difficult:
  - *g* is non-linear due to rotations, perspective division
  - lots of parameters: 3 for each 3D point, 6 for each camera
  - difficult to initialize
  - gauge ambiguity: error is invariant to a similarity transform (translation, rotation, uniform scale)
- Many techniques use non-linear least-squares (NLLS) optimization (*bundle adjustment*)
  - Levenberg-Marquardt is one common algorithm for NLLS
  - Lourakis, **The Design and Implementation of a Generic Sparse Bundle Adjustment Software Package Based on the Levenberg-Marquardt Algorithm**, http://www.ics.forth.gr/~lourakis/sba/
  - http://en.wikipedia.org/wiki/Levenberg-Marquardt_algorithm

Photo Tourism

- Structure from motion on Internet photo collections

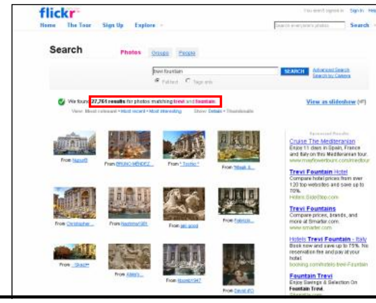


Photo Tourism

Photo Tourism overview

Input photographs

Scene reconstruction

Relative camera positions and orientations

Point cloud

Sparse correspondence

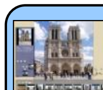


Photo Explorer

Scene reconstruction

Feature detection

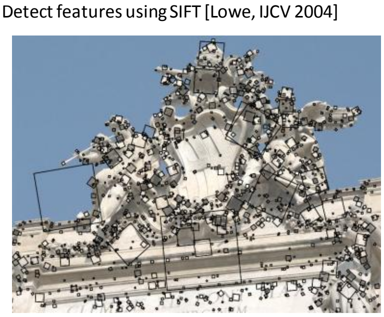
Pairwise feature matching

Correspondence estimation

Incremental structure from motion

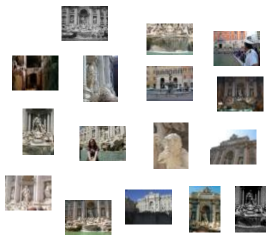
Feature detection

Detect features using SIFT [Lowe, IJCV 2004]



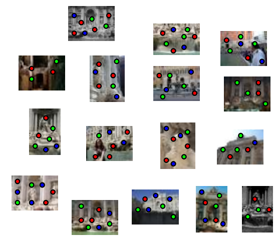
Feature detection

Detect features using SIFT [Lowe, IJCV 2004]



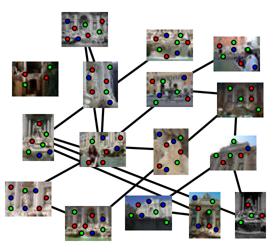
Feature detection

Detect features using SIFT [Lowe, IJCV 2004]



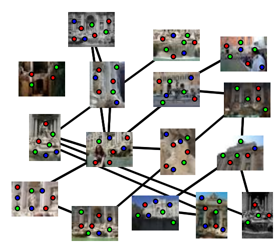
Feature matching

Match features between each pair of images

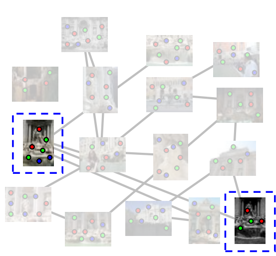


Feature matching

Refine matching using RANSAC [Fischler & Bolles 1987] to estimate fundamental matrices between pairs

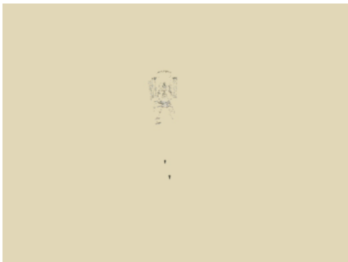


Incremental structure from motion



Incremental structure from motion

Incremental structure from motion



Problem size

- Trevi Fountain collection
  466 input photos
  \+ > 100,000 3D points
  = very large optimization problem

Photo Tourism overview

Photo Explorer

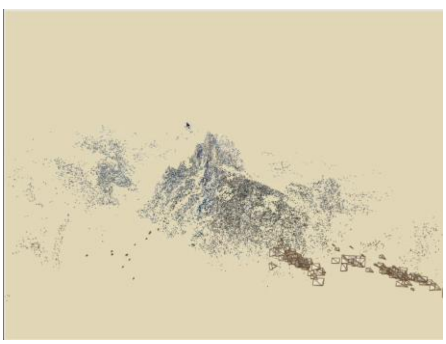
Overhead map

Prague Old Town Square

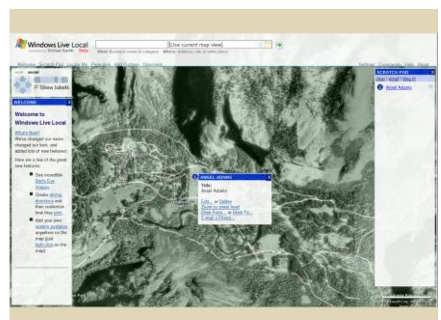
Annotations

Annotations

Yosemite



Yosemite



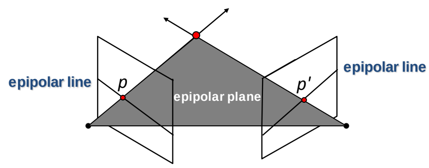
Two-view structure from motion

- Simpler case: can consider *motion* independent of structure

$$p = \begin{bmatrix} su \\ sv \\ s \end{bmatrix} = \underbrace{\begin{bmatrix} -fs_x & 0 & x'_c \\ 0 & -fs_y & y'_c \\ 0 & 0 & 1 \end{bmatrix}}_{\mathbf{K}} \left(\mathbf{R}_{3x3}\mathbf{x} + \mathbf{t}_{3x1}\right)$$

- Let's first consider the case where **K** is known
  - Each image point ($u_{i,j}$, $v_{i,j}$, 1) can be multiplied by $\mathbf{K}^{-1}$ to form a 3D ray
  - We call this the *calibrated* case

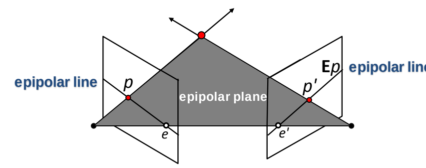
Notes on two-view geometry



- How can we express the epipolar constraint?
- Answer: there is a 3x3 matrix **E** such that

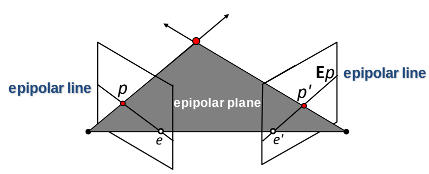
  $p'^T \mathbf{E} p = 0$
- **E** is called the essential matrix

Properties of the essential matrix



- $p'^T \mathbf{E} p = 0$
- $\mathbf{E}p$ is the epipolar line associated with p
- e and e' are called *epipoles*: $\mathbf{E}e = \mathbf{0}$ and $\mathbf{E}^T e' = \mathbf{0}$
- **E** can be solved for with 5 point matches
  - see Nister, **An efficient solution to the five-point relative pose problem**. *PAMI* 2004.

Properties of the essential matrix



$$\mathbf{E} = \mathbf{R}\begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix}$$

The Fundamental matrix

- If **K** is not known, then we use a related matrix called the *Fundamental matrix*, **F**
  - Called the *uncalibrated* case
- $\mathbf{F} = \mathbf{K}^{-T}\,\mathbf{E}\,\mathbf{K}^{-1}$
- **F** can be solved for linearly with eight points, or non-linearly with six or seven points

More information

- Paper: "Photo Tourism: Exploring photo collections in 3D," http://phototour.cs.washington.edu/Photo_Tourism.pdf
- http://phototour.cs.washington.edu
- http://labs.live.com/photosynth