Lecture 20: Fitting geometric models
Announcements

- PS8 - PS10 will be a bit easier (and PS10 will be short)
- Section this week: geometry tutorial
- My office hours next Monday (11/16) are cancelled
- Can chat in Friday OH instead, or by appointment
Today

• Finding correspondences
• Fitting a homography
• RANSAC
Panorama stitching (PS9)
Panorama stitching

Warp using homography
Panorama stitching

To estimate the homography, we need correspondences!
Finding correspondences with local features

1) **Detection**: Identify the interest points, the candidate points to match.

2) **Description**: Extract vector feature descriptor surrounding each interest point.

3) **Matching**: Determine correspondence between descriptors in two views.

\[ \mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}] \]

\[ \mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}] \]

Source: K. Grauman
What are good regions to match?

- How does the window change when you shift it?
- Shifting the window in any direction causes a big change

• "flat" region: no change in all directions
• "edge": no change along the edge direction
• "corner": significant change in all directions

Source: S. Seitz, D. Frolova, D. Simakov, N. Snavely
Finding good key points to match

Compute difference-of-Gaussians filter (approx. to Laplacian).

Find local optima in space and scale using Laplacian pyramid.
Feature descriptors

We know how to detect good points
Next question: **How to match them?**

Come up with a *descriptor* (feature vector) for each point, find similar descriptors between the two images
Simple idea: normalized image patch

We want invariance to rotation, lighting, and tiny spatial shifts.

Take 40x40 window around feature
- Find dominant orientation
- Rotate to horizontal
- Downsample to 8x8
- Intensity normalize the window by subtracting the mean, dividing by the standard deviation in the window
Scale Invariant Feature Transform (SIFT)

Basic idea: looks like a hand-crafted CNN
- Take 16x16 square window around detected feature
- Compute edge orientation for each pixel
- Create histogram of edge orientations

Source: N. Snavely, D. Lowe
Scale Invariant Feature Transform

Create the descriptor:

- Rotation invariance: rotate by “dominant” orientation
- Spatial invariance: spatial pool to 2x2
- Compute an orientation histogram for each cell
- (4 x 4) cells x 8 orientations = 128 dimensional descriptor

Source: N. Snavely, D. Lowe
SIFT invariances

Source: N. Snavely
Today

• Finding correspondences
• Computing local features
  • Matching
• Fitting a homography
• RANSAC
How can we tell if two features match?

Source: N. Snavely
Finding matches

How do we know if two features match?
- Simple approach: are they the nearest neighbor in $L_2$ distance, $\|f_1 - f_2\|$?
Finding matches

How do we know if two features match?

- Simple approach: are they the nearest neighbor in $L_2$ distance, $||f_1 - f_2||$?
- Can give good scores to ambiguous (incorrect) matches
Finding matches

Throw away matches that fail tests:

- **Ratio test**: this *by far* the best match?
  - Ratio distance = \[ \frac{||f_1 - f_2||}{||f_1 - f_2'||} \]
  - \(f_2\) is best SSD match to \(f_1\) in \(I_2\)
  - \(f_2'\) is 2\(^{nd}\) best SSD match to \(f_1\) in \(I_2\)
- **Forward-backward consistency**: \(f_1\) should also be nearest neighbor of \(f_2\)

Source: N. Snavely
Feature matching example

51 feature matches after ratio test

Source: N. Snavely
Feature matching example

58 feature matches after ratio test

Source: N. Snavely
Today

- Finding correspondences
- Computing local features
- Matching
- Fitting a homography
- RANSAC
From matches to a homography

\[(x_1, y_1) \Rightarrow (x_1', y_1')\]

\[
\begin{pmatrix}
  x_1' \\
  y_1' \\
  w_1
\end{pmatrix} =
\begin{pmatrix}
  a & b & c \\
  d & e & f \\
  g & h & i
\end{pmatrix} \cdot
\begin{pmatrix}
  x_1 \\
  y_1 \\
  1
\end{pmatrix}
\]

Source: Torralba, Isola, Freeman
From matches to a homography

Point in 1st image

Matched point in 2nd

minimize \( J(H) = \sum_{i} ||f_H(p_i) - p'_i||^2 \)

where \( f_H(p_i) = Hp_i/(H^T_3 p_i) \) applies homography

(remember: homogenous coordinates)
Option #1: Direct linear transform

\[
\begin{pmatrix}
    x_1' \\
    y_1' \\
    w_1
\end{pmatrix} = \begin{pmatrix}
    a & b & c \\
    d & e & f \\
    g & h & i
\end{pmatrix} \cdot \begin{pmatrix}
    x_1 \\
    y_1 \\
    1
\end{pmatrix}
\]

Going to heterogeneous coordinates:

\[
x_1' = \frac{ax_1 + by_1 + c}{gx_1 + hy_1 + i}
\]

\[
y_1' = \frac{dx_1 + ey_1 + f}{gx_1 + hy_1 + i}
\]

Re-arranging the terms:

\[
gx_1x_1' + hy_1x_1' + ix_1' = ax_1 + by_1 + c
\]

\[
gx_1y_1' + hy_1y_1' + ix_1' = dx_1 + ey_1 + f
\]

Source: Torralba, Freeman, Isola
Option #1: Direct linear transform

\[ gx_1 x'_1 + hy_1 x'_1 + ix'_1 = ax_1 + by_1 + c \]
\[ gx_1 y'_1 + hy_1 y'_1 + ix'_1 = dx_1 + ey_1 + f \]

Re-arranging the terms:
\[ gx_1 x'_1 + hy_1 x'_1 + ix'_1 - ax_1 - by_1 - c = 0 \]
\[ gx_1 y'_1 + hy_1 y'_1 + iy'_1 - dx_1 - ey_1 - f = 0 \]

In matrix form. Can solve using Singular Value Decomposition (SVD).

\[
\begin{bmatrix}
-x_1 & -y_1 & -1 & 0 & 0 & 0 & x_1 x'_1 & y_1 x'_1 & x'_1 \\
0 & 0 & 0 & -x_1 & -y_1 & -1 & x_1 y'_1 & y_1 y'_1 & y'_1
\end{bmatrix}
\begin{bmatrix}
\begin{bmatrix}
a \\ b \\ c \\ d \\ e \\ f \\ g \\ h \\ i
\end{bmatrix}
= \\
\begin{bmatrix}
0 \\
0
\end{bmatrix}
\]

Fast to solve (but not using “right” loss function). Uses an algebraic trick. Often used in practice for initial solutions!

Source: Torralba, Freeman, Isola
Option #2: Optimization

\[ J(H) \]

\[ \minimize J(H) = \sum_{i} \| f_{H}(p_i) - p_i' \|^2 \]
Optimization

\[ J(H) = \sum_i \| f_H(p_i) - p_i' \|^2 \]

- Can use gradient descent, just like when learning neural nets

- The problem is \textbf{smaller scale} than deep learning but has \textbf{more local optima}:
  - Use 2nd derivatives to improve optimization
  - Can use finite differences or autodiff
  - Can use special-purpose \textbf{nonlinear least squares} methods.

- Exploits structure in the problem for a sum-of-squares loss.
Outliers

Source: N. Snavely
One idea: robust loss functions

\[
\text{minimize} \quad J(H) = \sum_{i=1}^{N} \sum_{j=1}^{2} \rho(f_H(p_i)_j - p'_{ij})
\]

where \( \rho(x) \) is a robust loss.

Special case: \( \rho(x) = x^2 \) is L2 loss (same as before)
Robust loss functions

L1 loss: $\rho(x) = |x|$
Robust loss functions

Truncated quadratic: \( \rho(x) = \min(x^2, \tau) \)
Robust loss functions

Huber loss:

\[ \rho(x) = \begin{cases} 
\frac{1}{2}x^2 & \text{if } |x| \leq \tau, \\
\tau(|x| - \frac{1}{2}\tau), & \text{else}
\end{cases} \]
Robust loss functions

Source: [Barron 2019, “A General and Adaptive Robust Loss Function”]
Handling outliers

• Can be hard to fit robust loss
  • Can be low, or get stuck in bad local minima
• Let’s consider the problem of linear regression

Problem: Fit a line to these data points

Least squares fit

Source: N. Snavely
Counting inliers

Source: N. Snavely
Counting inliers

Inliers: 3

Source: N. Snavely
Counting inliers

Inliers: 20

Source: N. Snavely
RANSAC

- Idea:
  - All the inliers will agree with each other on the solution; the (hopefully small) number of outliers will (hopefully) disagree with each other
  - RANSAC only has guarantees if there are < 50% outliers

  - “All good matches are alike; every bad match is bad in its own way.”

  - Tolstoy via Alyosha Efros

Source: N. Snavely
RANSAC: random sample consensus

RANSAC loop (for N iterations):

• Select four feature pairs (at random)
• Compute homography $H$
• Count **inliers** where $\|p'_i - H p_i\| < \varepsilon$

Afterwards:

• Choose $H$ with largest set of inliers
• Recompute $H$ using only those inliers (often using high-quality nonlinear least squares)

Source: Torralba, Freeman, Isola
Simple example: fit a line

- Rather than homography $H$ (8 numbers), fit $y=ax+b$ (2 numbers $a$, $b$) to 2D pairs.

Source: Torralba, Freeman, Isola
Simple example: fit a line

- Pick 2 points
- Fit line
- Count inliers
Simple example: fit a line

- Pick 2 points
- Fit line
- Count inliers

Source: Torralba, Freeman, Isola
Simple example: fit a line

- Pick 2 points
- Fit line
- Count inliers

Source: Torralba, Freeman, Isola
Simple example: fit a line

- Pick 2 points
- Fit line
- Count inliers

Source: Torralba, Freeman, Isola
Simple example: fit a line

- Use biggest set of inliers
- Do least-square fit
Example: fitting a translation

Source: N. Snavely
Random Sample Consensus

Select one match at random, count inliers

Source: N. Snavely
Select one match at random, count inliers

Source: N. Snavely
RAndom SAmple Consensus

Select one match at random, count inliers

Source: N. Snavely
RAndom SAmple Consensus

Select another match at random, count inliers

Source: N. Snavely
RAndom SAmple Consensus

Select another match at random, count inliers

Source: N. Snavely
RAndon SAmple Consensus

Choose the translation with the highest number of inliers
Then compute average translation, using only inliers

Source: N. Snavely
Warping with a homography (PS9)

1. Compute features using SIFT
2. Match features
3. Compute homography using RANSAC

Source: N. Snavely
Next class: more 3D