Announcement

- Distinguished Lecture Series
- Prof. Alexei A. Efros from UC Berkeley
- 5/11 (Thurs), 3 pm in Kemper 1127

Last time

- RANSAC for robust fitting
  - Lines, translation
- Image mosaics
  - Fitting a 2D transformation
    - Homography
Today

Mosaics recap:
How to warp one image to the other, given $H$?

How to detect which features to match?

Motivation for feature-based alignment:
Image mosaics

Projective Transformations

$\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} a & b & c \\ d & e & f \\ g & h & i \end{bmatrix} \begin{bmatrix} x \\ y \\ w \end{bmatrix}$

Projective transformations:
- Affine transformations, and
- Projective warps

Parallel lines do not necessarily remain parallel

How to stitch together a panorama (a.k.a. mosaic)?

• Basic Procedure
  – Take a sequence of images from the same position
    • Rotate the camera about its optical center
  – Compute transformation between second image and first
  – Transform the second image to overlap with the first
  – Blend the two together to create a mosaic
  – (If there are more images, repeat)

Source: Steve Seitz

Mosaics

Obtain a wider angle view by combining multiple images.

How to stitch together a panorama (a.k.a. mosaic)?

• Basic Procedure
  – Take a sequence of images from the same position
    • Rotate the camera about its optical center
  – Compute transformation between second image and first
  – Transform the second image to overlap with the first
  – Blend the two together to create a mosaic
  – (If there are more images, repeat)

• …but wait, why should this work at all?
  – What about the 3D geometry of the scene?
  – Why aren't we using it?

Source: Steve Seitz
**Image reprojection**

**Basic question**
- How to relate two images from the same camera center?
  - how to map a pixel from PP1 to PP2

**Answer**
- Cast a ray through each pixel in PP1
- Draw the pixel where that ray intersects PP2

**Observation:**
Rather than thinking of this as a 3D reprojection, think of it as a 2D image warp from one image to another.

---

**Image reprojection: Homography**

A projective transform is a mapping between any two PPs with the same center of projection
- rectangle should map to arbitrary quadrilateral
- parallel lines aren’t preserved
- but must preserve straight lines

called **Homography**

\[
\begin{bmatrix}
\begin{array}{c}
1 \\
y' \\
wx' \\
w' \\
w
\end{array}
\end{bmatrix}
= H
\begin{bmatrix}
\begin{array}{c}
1 \\
y \\
wx \\
w' \\
w
\end{array}
\end{bmatrix}
\]

To compute the homography given pairs of corresponding points in the images, we need to set up an equation where the parameters of \( H \) are the unknowns...
Solving for homographies

\[ p' = Hp \]

\[
\begin{bmatrix}
wx' \\
wv' \\
w
\end{bmatrix} =
\begin{bmatrix}
h_{00} & h_{01} & h_{02} \\
h_{10} & h_{11} & h_{12} \\
h_{20} & h_{21} & h_{22}
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
1
\end{bmatrix}
\]

Defined up to a scale factor. Constrain Frobenius norm of H to be 1.

Problem to be solved:

\[ \min \|4h - h'\|^2 \quad \text{s.t.} \quad \|H\| = 1 \]

where vector of unknowns \( h = [h_{00}, h_{01}, h_{02}, h_{10}, h_{11}, h_{12}, h_{20}, h_{21}, h_{22}]^T \)

Adapted from Devi Parikh

There are 9 variables \( h_{00}, \ldots, h_{22} \). Are there 9 degrees of freedom?

No. We can multiply all \( h_{ij} \) by nonzero scalar \( k \) without changing the equations:

\[ w' = k w, \quad h' = k h \]

Enforcing 8 DOF

One approach: set \( h_{22} = 1 \)

\[
\begin{align*}
\hat{x}' &= \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + 1} \\
\hat{y}' &= \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + 1}
\end{align*}
\]

Second approach: impose unit vector constraint

\[
\begin{align*}
\hat{x}' &= \frac{h_{00}x + h_{01}y + h_{02}}{h_{20}x + h_{21}y + 1} \\
\hat{y}' &= \frac{h_{10}x + h_{11}y + h_{12}}{h_{20}x + h_{21}y + 1}
\end{align*}
\]

Subject to:

\[ h_{00} + h_{01} + h_{02} + h_{10} + h_{11} + h_{12} + h_{20} + h_{21} + h_{22} = 1 \]
Solving for homographies

\[
\begin{bmatrix}
\begin{bmatrix}
1 & 0 & 0 & 0 \\
0 & 1 & 0 & 0 \\
0 & 0 & 1 & 0 \\
0 & 0 & 0 & 1 \\
\end{bmatrix}
& \begin{bmatrix}
-h_1 & h_0 \\
h_1 & -h_0 \\
-h_2 & h_0 \\
h_2 & -h_0 \\
\end{bmatrix}
& \begin{bmatrix}
h_3 \\
h_6 \\
h_9 \\
h_{12} \\
\end{bmatrix}
\end{bmatrix}
\begin{bmatrix}
w_1 \\
w_2 \\
w_3 \\
w_4 \\
\end{bmatrix}
= \begin{bmatrix}
w_5 \\
w_6 \\
w_7 \\
w_8 \\
\end{bmatrix}
\]

\[A \cdot h = 0\]

Defines a least squares problem:

minimize \(\|Ah - 0\|^2\)

- Since \(h\) is only defined up to scale, solve for unit vector \(\hat{h}\)
- Solution: \(\hat{h}\) = eigenvector of \(A^T A\) with smallest eigenvalue
- Works with 4 or more points

Projective: # correspondences?

How many correspondences needed for projective?

Source: Alyosha Efros
Homography

\[(x, y) \rightarrow (x', y') = (wx/w, wy/w)\]

To apply a given homography $H$

- Compute $p' = Hp$ (regular matrix multiply)
- Convert $p'$ from homogeneous to image coordinates

\[
P' = H P
\]

Image warping

Given a coordinate transform and a source image $f(x, y)$, how do we compute a transformed image $g(x', y') = f(T(x, y))$?

Forward warping

Send each pixel $f(x, y)$ to its corresponding location $(x', y') = T(x, y)$ in the second image.

Q: what if pixel lands “between” two pixels?
**Forward warping**

Send each pixel \( f(x,y) \) to its corresponding location \((x',y') = T(x,y)\) in the second image.

**Q:** what if pixel lands “between” two pixels?

**A:** distribute color among neighboring pixels \((x',y')\)

— Known as “splatting”

**Inverse warping**

Get each pixel \( g(x',y') \) from its corresponding location \((x,y) = T^{-1}(x',y')\) in the first image.

**Q:** what if pixel comes from “between” two pixels?

**A:** Interpolate color value from neighbors

— Nearest neighbor, bilinear...
Bilinear interpolation
Sampling at \( f(x,y) \):

\[
\begin{array}{c}
(i,j) & (i,j+1) & (i+1,j) & (i+1,j+1) \\
\hline
\alpha & \beta & \gamma & \delta \\
\end{array}
\]

\[
f(x, y) = (1 - \alpha)(1 - \beta) f[i,j] \\
+ \alpha(1 - \beta) f[i+1,j] \\
+ \beta(1 - \alpha) f[i+1,j+1] \\
+ (1 - \alpha)\beta f[i,j+1]
\]

---

Image warping with homographies

---

Image rectification
Analysing patterns and shapes

What is the shape of the b/w floor pattern?

The floor (enlarged)

Automatic rectified floor

Slide from Antonio Criminisi

From Martin Kemp *The Science of Art* (manual reconstruction)

Analysing patterns and shapes

Automatic rectification

*St. Lucy Altarpiece, D. Veneziano*

Slide from Criminisi

What is the (complicated) shape of the floor pattern?

Automatically rectified floor

Slide from Criminisi
Analysing patterns and shapes

Automatic rectification

From Martin Kemp, *The Science of Art* (manual reconstruction)

Slide from Criminisi

---

Changing camera center

Does it still work?

Source: Alyosha Efros

---

Recall: same camera center

Can generate synthetic camera view as long as it has the same center of projection.

Source: Alyosha Efros
...Or: Planar scene (or far away)

PP3 is a projection plane of both centers of projection, so we are OK!
This is how big aerial photographs are made

RANSAC for estimating homography

RANSAC loop:
1. Select four feature pairs (at random)
2. Compute homography $H$ (exact)
3. Compute inliers where $SSD(p_i', H p_i) < \varepsilon$
4. Keep largest set of inliers
5. Re-compute least-squares $H$ estimate on all of the inliers
Robust feature-based alignment

- Extract features
- Compute putative matches

Source: L. Lazebnik
Robust feature-based alignment

- Extract features
- Compute putative matches
- Loop:
  - Hypothesize transformation $T$ (small group of putative matches that are related by $T$)
  - Verify transformation (search for other matches consistent with $T$)

Source: L. Lazebnik
Summary: alignment & warping

- Write 2d transformations as matrix-vector multiplication (including translation when we use homogeneous coordinates).
- Fitting transformations: solve for unknown parameters given corresponding points from two views (affine, projective (homography)).
- Perform image warping (forward, inverse).
- Mosaics: uses homography and image warping to merge views taken from same center of projection.

Boundary extension


Creating and Exploring a Large Photorealistic Virtual Space

http://www.youtube.com/watch?v=EdboU1oPq
Creating and Exploring a Large Photorealistic Virtual Space

Current view, and desired view in green

Synthesized view from new camera

Induced camera motion

Today

Mosaics recap:
How to warp one image to the other, given H?

How to detect which features to match?

Detecting local invariant features

- Detection of interest points
  - Harris corner detection
  - (Scale invariant blob detection: LoG)
- (Next time: description of local patches)
Local features: main components

1) Detection: Identify the interest points

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

3) Matching: Determine correspondence between descriptors in two views

Local features: desired properties

- **Repeatability**
  - The same feature can be found in several images despite geometric and photometric transformations

- **Saliency**
  - Each feature has a distinctive description

- **Compactness and efficiency**
  - Many fewer features than image pixels

- **Locality**
  - A feature occupies a relatively small area of the image; robust to clutter and occlusion

Applications

- Local features are used for:
  - Image alignment
  - 3D reconstruction
  - Motion tracking
  - Robot navigation
  - Indexing and database retrieval
  - Object recognition
A hard feature matching problem

NASA Mars Rover images

Answer below (look for tiny colored squares…)

NASA Mars Rover images with SIFT feature matches
Figure by Noah Snavely

Goal: interest operator repeatability

- We want to detect (at least some of) the same points in both images.

No chance to find true matches!

- Yet we have to be able to run the detection procedure independently per image.
Goal: descriptor distinctiveness

• We want to be able to reliably determine which point goes with which.

• Must provide some invariance to geometric and photometric differences between the two views.

Local features: main components

1) Detection: Identify the interest points

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

3) Matching: Determine correspondence between descriptors in two views

• What points would you choose?
**Corners** as distinctive interest points

We should easily recognize the point by looking through a small window. Shifting a window in *any direction* should give a large change in intensity.

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

Slide credit: Alyosha Efros, Darya Frolova, Denis Simakov

---

**Corners** as distinctive interest points

\[
M = \sum \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix}
\]

2 x 2 matrix of image derivatives (averaged in neighborhood of a point).

Notation:

\[
I_x \leftrightarrow \frac{\partial I}{\partial x}, \quad I_y \leftrightarrow \frac{\partial I}{\partial y}, \quad I_x I_y \leftrightarrow \frac{\partial I}{\partial x} \frac{\partial I}{\partial y}
\]

---

What does this matrix reveal?

First, consider an axis-aligned corner:
What does this matrix reveal?

First, consider an axis-aligned corner:

\[
M = \sum \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix} = \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix}
\]

This means dominant gradient directions align with x or y axis.

Look for locations where both \( \lambda \)'s are large.

If either \( \lambda \) is close to 0, then this is not corner-like.

What if we have a corner that is not aligned with the image axes?

What does this matrix reveal?

Since \( M \) is symmetric, we have

\[
M = X \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} X^T
\]

(Eigenvalue decomposition)

\[ Mx_i = \lambda_i x_i \]

The eigenvalues of \( M \) reveal the amount of intensity change in the two principal orthogonal gradient directions in the window.

Corner response function

"edge":
\( \lambda_1 \gg \lambda_2 \)
\( \lambda_2 \gg \lambda_1 \)

"corner":
\( \lambda_1 \) and \( \lambda_2 \) are large,
\( \lambda_1 \approx \lambda_2 \);

"flat" region
\( \lambda_1 \) and \( \lambda_2 \) are small:

\[
f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}
\]
Harris corner detector

1) Compute $M$ matrix for each image window to get their cornerness scores.
2) Find points whose surrounding window gave large corner response ($f > \text{threshold}$)
3) Take the points of local maxima, i.e., perform non-maximum suppression

Example of Harris application

Compute corner response at every pixel.
Example of Harris application

Properties of the Harris corner detector
Rotation invariant? Yes

Properties of the Harris corner detector
Rotation invariant? Yes
Translation invariant? Yes
Properties of the Harris corner detector

Rotation invariant?  Yes
Translation invariant?  Yes
Scale invariant?  No

All points will be classified as edges

Corner!

Summary

• Image warping to create mosaic, given homography

• Interest point detection
  – Harris corner detector
  – Next time:
    • Laplacian of Gaussian, automatic scale selection