Lecture 10: Feature Matching and Transforms
Announcements

• HW 1
  – Some clarifications will be posted (check FAQ on piazza)
• HW 1 will be due a bit later (prob Fri)

• In class review: Fri Feb 27

• Prelim: Mar 5 2015
Feature descriptors

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

**Answer:** Come up with a *descriptor* for each point, find similar descriptors between the two images.
Scale Invariant Feature Transform

Basic idea:

• Take 16x16 square window around detected feature
• Compute gradient orientation for each pixel
• Throw out weak edges (threshold gradient magnitude)
• Create histogram of surviving edge orientations

Adapted from slide by David Lowe
SIFT vector formation

- Computed on rotated and scaled version of window according to computed orientation & scale
  - resample the window
- Based on gradients weighted by a Gaussian of variance 1.5 times the window (for smooth falloff)
SIFT vector formation

- 4x4 array of gradient orientation histogram weighted by magnitude
- 8 orientations x 4x4 array = 128 dimensions
- Motivation: some sensitivity to spatial layout, but not too much

Image gradients → Keypoint descriptor showing only 2x2 here but is 4x4
Ensure smoothness

- Gaussian weight
- Trilinear interpolation
  - a given gradient contributes to 8 bins:
  - 4 in space times 2 in orientation
Reduce effect of illumination

- 128-dim vector normalized to 1
- Threshold gradient magnitudes to avoid excessive influence of high gradients
  - after normalization, clamp gradients >0.2
  - renormalize
Properties of SIFT

Extraordinarily robust matching technique

– Can handle changes in viewpoint
  • Up to about 60 degree out of plane rotation
– Can handle significant changes in illumination
  • Sometimes even day vs. night (below)
– Fast and efficient—can run in real time
– Lots of code available:
Summary

• Keypoint detection: repeatable and distinctive
  – Corners, blobs, stable regions
  – Harris, DoG

• Descriptors: robust and selective
  – spatial histograms of orientation
  – SIFT and variants are typically good for stitching and recognition
  – But, need not stick to one
Which features match?
Feature matching

Given a feature in $I_1$, how to find the best match in $I_2$?

1. Define distance function that compares two descriptors
2. Test all the features in $I_2$, find the one with min distance
Feature distance

How to define the difference between two features $f_1, f_2$?

- Simple approach: $L_2$ distance, $||f_1 - f_2||$
- can give good scores to ambiguous (incorrect) matches
Feature distance

How to define the difference between two features $f_1, f_2$?

- Better approach: 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 $I_2$
  - gives bad scores for ambiguous matches
Evaluating the results

How can we measure the performance of a feature matcher?
True/false positives

How can we measure the performance of a feature matcher?

The distance threshold affects performance

- **True positives** = # of detected matches that are correct
  - Suppose we want to maximize these—how to choose threshold?
- **False positives** = # of detected matches that are incorrect
  - Suppose we want to minimize these—how to choose threshold?
Precision and recall are two important metrics used in evaluating the performance of information retrieval systems. Precision is calculated as the number of relevant items that are selected divided by the total number of items selected. Recall is calculated as the number of relevant items that are selected divided by the total number of relevant items in the dataset.

\[
\text{Precision} = \frac{\text{Number of correct positives selected}}{\text{Number of positives selected}}
\]

\[
\text{Recall} = \frac{\text{Number of correct positives selected}}{\text{Number of relevant positives}}
\]
Confusion Matrix

https://uberpython.wordpress.com/2012/01/01/precision-recall-sensitivity-and-specificity/
Classified as

Positive

Negative

True Positive

False Negative

False Positive

True Negative

Really is

Positive

Negative

— Precision in red, recall in yellow
Precision vs. Recall

Examples

1000 animals, 100 dogs
Algorithm finds 50 (of which 40 are dogs, 10 are cats)
  Precision =
  Recall =

Algorithm finds 10 (of which 10 are dogs)
  Precision =
  Recall =

Algorithm returns 1000 (of which 100 are dogs)
  Precision =
  Recall =
Sensitivity in yellow, specificity in red
How can we measure the performance of a feature matcher?

Evaluating the results

- **true positive rate**
  - \( \frac{\text{# true positives}}{\text{# matching features (positives)}} \)
  - "recall"

- **false positive rate**
  - \( \frac{\text{# false positives}}{\text{# unmatched features (negatives)}} \)

Graph: A scatter plot with axes labeled "false positive rate" on the x-axis and "true positive rate" on the y-axis.
Evaluating the results

How can we measure the performance of a feature matcher?

“recall” = \frac{\text{# true positives}}{\text{# matching features (positives)}}

roc curve (“receiver operator characteristic”)

AUC (area under curve) = \int_0^1 \text{roc(t)} \, dt
More on feature detection/description

Affine Covariant Regions

Publications

Region detectors


Region descriptors


Performance evaluation

Lots of applications

Features are used for:

– Image alignment (e.g., mosaics)
– 3D reconstruction
– Motion tracking
– Object recognition
– Indexing and database retrieval
– Robot navigation
– ... other
3D Reconstruction

Internet Photos (“Colosseum”) -> Reconstructed 3D cameras and points
Object recognition (David Lowe)
Sony Aibo

**SIFT usage:**

- Recognize charging station
- Communicate with visual cards
- Teach object recognition
Available at a web site near you...

• For most local feature detectors, executables are available online:
  – http://www.robots.ox.ac.uk/~vgg/research/affine
  – http://www.cs.ubc.ca/~lowe/keypoints/
  – http://www.vision.ee.ethz.ch/~surf
Questions?
Image alignment
What is the geometric relationship between these two images?

Answer: Similarity transformation (translation, rotation, uniform scale)
What is the geometric relationship between these two images?
What is the geometric relationship between these two images?

Very important for creating mosaics!
Image Warping

• image filtering: change range of image
  • $g(x) = h(f(x))$

\[ f(x) \xrightarrow{h} g(x) \]

• image warping: change domain of image
  • $g(x) = f(h(x))$

\[ f(x) \xrightarrow{h} g(x) \]
Image Warping

- image filtering: change range of image
  - \( g(x) = h(f(x)) \)

- image warping: change domain of image
  - \( g(x) = f(h(x)) \)
Parametric (global) warping

• Examples of parametric warps:

  translation
  rotation
  aspect
Parametric (global) warping

- Transformation $T$ is a coordinate-changing machine:
  \[ p' = T(p) \]

- What does it mean that $T$ is global?
  - Is the same for any point $p$
  - Can be described by just a few numbers (parameters)

- Let’s consider linear xforms (can be represented by a 2D matrix):

\[
p' = Tp \quad \begin{bmatrix} x' \\ y' \end{bmatrix} = T \begin{bmatrix} x \\ y \end{bmatrix}
\]
Common linear transformations

- Uniform scaling by $s$:

$$S = \begin{bmatrix} s & 0 \\ 0 & s \end{bmatrix}$$

What is the inverse?
Common linear transformations

• Rotation by angle $\theta$ (about the origin)

\[
\mathbf{R} = \begin{bmatrix}
\cos \theta & -\sin \theta \\
\sin \theta & \cos \theta
\end{bmatrix}
\]

What is the inverse?

For rotations:

\[
\mathbf{R}^{-1} = \mathbf{R}^T
\]
2x2 Matrices

- What types of transformations can be represented with a 2x2 matrix?

2D mirror about Y axis?

\[
\begin{align*}
    x' &= -x \\
    y' &= y
\end{align*}
\]

\[ T = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \]

2D mirror across line \( y = x \)?

\[
\begin{align*}
    x' &= y \\
    y' &= x
\end{align*}
\]

\[ T = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \]
• What types of transformations can be represented with a 2x2 matrix?

2x2 Matrices

2D Translation?

\[
x' = x + t_x
\]
\[
y' = y + t_y
\]

NO!

Translation is not a linear operation on 2D coordinates
All 2D Linear Transformations

• Linear transformations are combinations of ...
  – Scale,
  – Rotation,
  – Shear, and
  – Mirror

\[
\begin{bmatrix}
x'
\end{bmatrix} = \begin{bmatrix}
a & b \\
\end{bmatrix} \begin{bmatrix}
x
\end{bmatrix} \\
\begin{bmatrix}
y'
\end{bmatrix} = \begin{bmatrix}
c & d \\
\end{bmatrix} \begin{bmatrix}
y
\end{bmatrix}
\]

• Properties of linear transformations:
  – Origin maps to origin
  – Lines map to lines
  – Parallel lines remain parallel
  – Ratios are preserved
  – Closed under composition

\[
\begin{bmatrix}
x'
\end{bmatrix} = \begin{bmatrix}
a & b \\
\end{bmatrix} \begin{bmatrix}
e & f \\
\end{bmatrix} \begin{bmatrix}
i & j \\
\end{bmatrix} \begin{bmatrix}
x
\end{bmatrix} \\
\begin{bmatrix}
y'
\end{bmatrix} = \begin{bmatrix}
c & d \\
\end{bmatrix} \begin{bmatrix}
g & h \\
\end{bmatrix} \begin{bmatrix}
k & l \\
\end{bmatrix} \begin{bmatrix}
y
\end{bmatrix}
\]
Homogeneous coordinates

Trick: add one more coordinate:

\[(x, y) \Rightarrow \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}\]

homogeneous image coordinates

Converting *from* homogeneous coordinates

\[
\begin{bmatrix} x \\ y \\ w \end{bmatrix} \Rightarrow \left(\frac{x}{w}, \frac{y}{w}\right)
\]
Translation

- Solution: homogeneous coordinates to the rescue

\[
T = \begin{bmatrix}
1 & 0 & t_x \\
0 & 1 & t_y \\
0 & 0 & 1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 0 & t_x \\
0 & 1 & t_y \\
0 & 0 & 1 \\
\end{bmatrix}
\begin{bmatrix}
x \\
y \\
1 \\
\end{bmatrix}
= \begin{bmatrix}
x + t_x \\
y + t_y \\
1 \\
\end{bmatrix}
\]
Affine transformations

\[ T = \begin{bmatrix}
1 & 0 & t_x \\
0 & 1 & t_y \\
0 & 0 & 1
\end{bmatrix} \]

any transformation with last row \([ 0 \ 0 \ 1 ]\) we call an affine transformation

\[
\begin{bmatrix}
a & b & c \\
d & e & f \\
0 & 0 & 1
\end{bmatrix}
\]
Basic affine transformations

Translate

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} =
\begin{bmatrix}
  1 & 0 & t_x \\
  0 & 1 & t_y \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

2D in-plane rotation

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} =
\begin{bmatrix}
  \cos \theta & -\sin \theta & 0 \\
  \sin \theta & \cos \theta & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

Scale

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} =
\begin{bmatrix}
  s_x & 0 & 0 \\
  0 & s_y & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]

Shear

\[
\begin{bmatrix}
  x' \\
  y' \\
  1
\end{bmatrix} =
\begin{bmatrix}
  1 & sh_x & 0 \\
  sh_y & 1 & 0 \\
  0 & 0 & 1
\end{bmatrix}
\begin{bmatrix}
  x \\
  y \\
  1
\end{bmatrix}
\]
Affine Transformations

• Affine transformations are combinations of ...
  – Linear transformations, and
  – Translations

\[
\begin{bmatrix}
x' \\
y' \\
w
\end{bmatrix} = \begin{bmatrix}
a & b & c \\
d & e & f \\
0 & 0 & 1
\end{bmatrix} \begin{bmatrix}
x \\
y \\
w
\end{bmatrix}
\]

• Properties of affine transformations:
  – Origin does not necessarily map to origin
  – Lines map to lines
  – Parallel lines remain parallel
  – Ratios are preserved
  – Closed under composition
Is this an affine transformation?