CS5670: Computer Vision
Feature descriptors and matching
Reading

• Szeliski: 4.1
Announcements

• Project 1 artifact due tonight at 11:59pm
• Project 2 to be released next week
  – Due Monday, March 2
  – To be done in groups of 2
Local features: main components

1) **Detection**: Identify the interest points

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

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

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

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

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

Lots of possibilities

- Simple option: match square windows around the point
- State of the art approach: SIFT
Invariance vs. discriminability

• Invariance:
  – Descriptor shouldn’t change even if image is transformed

• Discriminability:
  – Descriptor should be highly unique for each point
Image transformations revisited

- Geometric
  - Rotation
- Photometric
  - Intensity change
Invariant descriptors

• We looked at invariant / equivariant detectors

• Most feature descriptors are also designed to be invariant to:
  – Translation, 2D rotation, scale

• They can usually also handle
  – Limited 3D rotations (SIFT works up to about 60 degrees)
  – Limited affine transforms (some are fully affine invariant)
  – Limited illumination/contrast changes
How to achieve invariance

Need both of the following:

1. Make sure your detector is invariant
2. Design an invariant feature descriptor
   - Simplest descriptor: a single 0
     • What’s this invariant to?
   - Next simplest descriptor: a square, axis-aligned 5x5 window of pixels
     • What’s this invariant to?
   - Let’s look at some better approaches...
Rotation invariance for feature descriptors

• Find dominant orientation of the image patch
  – E.g., given by $x_{\text{max}}$, the eigenvector of $H$ corresponding to $\lambda_{\text{max}}$ (the larger eigenvalue)
  – Or simply the orientation of the (smoothed) gradient
  – Rotate the patch according to this angle
Multiscale Oriented PatcheS descriptor

Take 40x40 square window around detected feature
- Scale to 1/5 size (using prefiltering)
- Rotate to horizontal
- Sample 8x8 square window centered at feature
- Intensity normalize the window by subtracting the mean, dividing by the standard deviation in the window (why?)

Adapted from slide by Matthew Brown
Detections at multiple scales

Figure 1. Multi-scale Oriented Patches (MOPS) extracted at five pyramid levels from one of the Matter images. The boxes show the feature orientation and the region from which the descriptor vector is sampled.
**Scale Invariant Feature Transform**

Basic idea:

- Take 16x16 square window around detected feature
- Compute edge orientation (angle of the gradient - 90°) for each pixel
- Throw out weak edges (threshold gradient magnitude)
- Create histogram of surviving edge orientations

Adapted from slide by David Lowe
SIFT descriptor

Full version

- Divide the 16x16 window into a 4x4 grid of cells (2x2 case shown below)
- Compute an orientation histogram for each cell
- 16 cells * 8 orientations = 128 dimensional descriptor

Adapted from slide by David Lowe
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))
- Relatively fast—hard to make real-time, but can run in <1s for moderate image sizes
- Lots of code available
SIFT Example

868 SIFT features
Other descriptors

- **HOG**: Histogram of Gradients (HOG)
  - Dalal/Triggs
  - Sliding window, pedestrian detection

- **FREAK**: Fast Retina Keypoint
  - Perceptually motivated
  - Used in Visual SLAM

- **LIFT**: Learned Invariant Feature Transform
  - Learned via deep learning
    
    [Source](https://arxiv.org/abs/1603.09114)
Questions?
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 small distances for 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 2nd best SSD match to $f_1$ in $I_2$
- gives large values for ambiguous matches
Feature distance

• Does the SSD vs “ratio distance” change the best match to a given feature in image 1?
Feature matching example

58 matches (thresholded by ratio score)
Feature matching example

51 matches (thresholded by ratio score)

We’ll deal with outliers later
Evaluating the results

How can we measure the performance of a feature matcher?

feature distance
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?
Evaluating the results

How can we measure the performance of a feature matcher?

\[
\text{recall} = \frac{\text{# true positives}}{\text{# matching features (positives)}}
\]

\[
1 - \text{specificity} = \frac{\text{# false positives}}{\text{# unmatched features (negatives)}}
\]
Evaluating the results

How can we measure the performance of a feature matcher?

ROC curve ("Receiver Operator Characteristic")

Single number: Area Under the Curve (AUC)
E.g. AUC = 0.87
1 is the best

\[
\text{recall} = \frac{\text{# true positives}}{\text{# matching features (positives)}}
\]

\[
\text{specificity} = 1 - \frac{\text{# false positives}}{\text{# unmatched features (negatives)}}
\]
More on feature detection/description

http://www.robots.ox.ac.uk/~vgg/research/affine/
http://www.cs.ubc.ca/~lowe/keypoints/
http://www.vision.ee.ethz.ch/~surf/

**Publications**

**Region detectors**

**Region descriptors**
- **SIFT**: D. Lowe. Distinctive image features from scale invariant keypoints. In IJC V 60(2):91-110, 2004. [PDF](http://www.robots.ox.ac.uk/~vgg/research/affine/)

**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
Object recognition (David Lowe)
3D Reconstruction

Internet Photos ("Colosseum")

Reconstructed 3D cameras and points
Augmented Reality
Questions?