Recognizing object instances

Kristen Grauman
UT-Austin

Announcements

• Assignment 1 is out, due Fri Sept 22
• Presentation assignments - up this week

• Reminder – no laptops, phones, etc. in class please

Plan for today

• 1. Basics in feature extraction: filtering
• 2. Invariant local features
• 3. Recognizing object instances

Basics in feature extraction

Image Formation

Digital images
Digital images

- **Sample** the 2D space on a regular grid
- **Quantize** each sample (round to nearest integer)
- Image thus represented as a matrix of integer values.

Digital color images

- **Color images**, RGB color space

Main idea: image filtering

- Compute a function of the local neighborhood at each pixel in the image
  - Function specified by a "filter" or mask saying how to combine values from neighbors.
- Uses of filtering:
  - Enhance an image (denoise, resize, etc)
  - Extract information (texture, edges, etc)
  - Detect patterns (template matching)

Motivation: noise reduction

- Even multiple images of the **same static scene** will not be identical.
- Even multiple images of the same static scene will not be identical.
  - How could we reduce the noise, i.e., give an estimate of the true intensities?
  - **What if there's only one image?**
First attempt at a solution

- Let's replace each pixel with an average of all the values in its neighborhood
- Assumptions:
  - Expect pixels to be like their neighbors
  - Expect noise processes to be independent from pixel to pixel

Weighted Moving Average

Can add weights to our moving average

Weights \([1, 1, 1, 1, 1] / 5\)

Moving Average In 2D

\[ F[x, y] \quad G[x, y] \]

Weighted Moving Average

Non-uniform weights \([1, 4, 6, 4, 1] / 16\)
Moving Average In 2D

\[ F[x, y] \quad G[x, y] \]

Source: S. Seitz

Correlation filtering

Say the averaging window size is \( 2k+1 \times 2k+1 \):

\[
G[i, j] = \frac{1}{(2k + 1)^2} \sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i + u, j + v]
\]

Now generalize to allow different weights depending on neighboring pixel's relative position:

\[
G[i, j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u, v] F[i + u, j + v]
\]

This is called cross-correlation, denoted \( G = H \otimes F \).

Filtering an image: replace each pixel with a linear combination of its neighbors.

The filter "kernel" or "mask" \( H[u, v] \) is the prescription for the weights in the linear combination.
Averaging filter

- What values belong in the kernel $H$ for the moving average example?

$$G = H \otimes F$$

Smoothing by averaging

- What if the filter size was 5 x 5 instead of 3 x 3?

Gaussian filter

- What if we want nearest neighboring pixels to have the most influence on the output?

- Removes high-frequency components from the image ("low-pass filter").

Smoothing with a Gaussian

- What parameters matter here?

- Variance of Gaussian: determines extent of smoothing

- Parameter $\sigma$ is the "scale" / "width" / "spread" of the Gaussian kernel, and controls the amount of smoothing.

Gaussian filters

- $\sigma = 2$ with 30 x 30 kernel
- $\sigma = 5$ with 30 x 30 kernel

Smoothing with a Gaussian

```matlab
for sigma=1:3:10
    h = fspecial('gaussian', fsize, sigma);
    out = imfilter(im, h);
    imshow(out);
    pause;
end
```
Properties of smoothing filters

- **Smoothing**
  - Values positive
  - Sum to 1
  - Amount of smoothing proportional to mask size
  - Remove "high-frequency" components; "low-pass" filter

Predict the outputs using correlation filtering

\[
\begin{array}{cccc}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{array} \times \begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array} = ?
\]

\[
\begin{array}{cccc}
0 & 0 & 1 \\
0 & 0 & 1 \\
0 & 0 & 0 \\
\end{array} \times \begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array} = ?
\]

Practice with linear filters

Original

\[
\begin{array}{cccc}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{array}
\]

Original

Filtered (no change)

Source: D. Lowe

Original

Shifted left by 1 pixel with correlation

Source: D. Lowe
Practice with linear filters

Original

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

Source: D. Lowe

Blur (with a box filter)

Original

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

Source: D. Lowe

Practice with linear filters

Original

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

Source: D. Lowe

Sharpening filter:
accentuates differences
with local average

Original

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0 \\
\end{array}
\]

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{array}
\]

Source: D. Lowe

Filtering examples: sharpening

before

after

Filtering application: Hybrid Images

Aude Oliva & Antonio Torralba & Philippe G Schyns, SIGGRAPH 2006
Application: Hybrid Images


Gaussian Filter

Laplacian Filter

unit impulse

Gaussian Laplacian of Gaussian

Main idea: image filtering

• Compute a function of the local neighborhood at each pixel in the image
  – Function specified by a “filter” or mask saying how to combine values from neighbors.

• Uses of filtering:
  – Enhance an image (denoise, resize, etc)
  – Extract information (texture, edges, etc)
  – Detect patterns (template matching)

Why are gradients important?

Kristen Grauman

Derivatives and edges

An edge is a place of rapid change in the image intensity function.

Krista J. Gibson
Derivatives with convolution

For 2D function, \( f(x,y) \), the partial derivative is:

\[
\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \frac{f(x + \varepsilon, y) - f(x, y)}{\varepsilon}
\]

For discrete data, we can approximate using finite differences:

\[
\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x+1,y) - f(x,y)}{1}
\]

To implement above as convolution, what would be the associated filter?

Partial derivatives of an image

The gradient of an image:

\[
\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}
\]

The gradient points in the direction of most rapid change in intensity

\[ \nabla f = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, \quad \nabla f = \begin{bmatrix} 1 \\ 0 \end{bmatrix} \]

The gradient direction (orientation of edge normal) is given by:

\[ \theta = \tan^{-1} \left( \frac{\partial f / \partial y}{\partial f / \partial x} \right) \]

Main idea: image filtering

- Compute a function of the local neighborhood at each pixel in the image
  - Function specified by a “filter” or mask saying how to combine values from neighbors.

- Uses of filtering:
  - Enhance an image (denoise, resize, etc)
  - Extract information (texture, edges, etc)
  - Detect patterns (template matching)

Template matching

- Filters as templates:
  Note that filters look like the effects they are intended to find — “matched filters”

- Use normalized cross-correlation score to find a given pattern (template) in the image.
- Normalization needed to control for relative brightnesses.
Template matching

Scene

Template (mask)

A toy example

Template matching

Scene

Detected template

Correlation map

Where's Waldo?

Template

Detected template

Correlation map

Where's Waldo?

Detected template

Correlation map
Template matching

What if the template is not identical to some subimage in the scene?

Detected template

Match can be meaningful, if scale, orientation, and general appearance is right.

…but we can do better!...

Summary so far

• Compute a function of the local neighborhood at each pixel in the image
  – Function specified by a “filter” or mask saying how to combine values from neighbors.
• Uses of filtering:
  – Enhance an image (denoise, resize, etc)
  – Extract information (texture, edges, etc)
  – Detect patterns (template matching)

Plan for today

• 1. Basics in feature extraction: filtering
• 2. Invariant local features
• 3. Specific object recognition methods

Local features:
detection and description

Basic goal
Local features: main components

1) Detection: Identify the interest points

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

\[ x_1 = [x_1^{(1)}, \ldots, x_1^{(l)}] \]
\[ x_2 = [x_2^{(1)}, \ldots, x_2^{(l)}] \]

3) Matching: Determine correspondence between descriptors in two views

Goal: interest operator repeatability

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

- 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

Detecting corners

- What points would you choose?
Detecting corners
Compute “cornerness” response at every pixel.

Detecting local invariant features
• Detection of interest points
  – Harris corner detection
  – Scale invariant blob detection: LoG

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

Corners as distinctive interest points
\[ M = \sum w(x, y) \begin{bmatrix} I_x I_x & I_x I_y \\ I_x I_y & I_y I_y \end{bmatrix} \]

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:
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?

**Corner response function**

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

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

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

Corner score (other variants possible)

\[
f = \frac{\lambda_1 \lambda_2}{\lambda_1 + \lambda_2}
\]

Since \(M\) is symmetric, we have

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

\[
MX_i = \lambda_i x_i
\]

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

**Harris corner detector**

1. Compute \(M\) matrix for each image window to get their corneress 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
Harris Detector: Steps

Find points with large corner response: \( f > \text{threshold} \)

Harris Detector: Steps

Take only the points of local maxima of \( f \)

Properties of the Harris corner detector

Rotation invariant? Yes

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

Scale invariant? No

All points will be classified as edges

Corner!

Scale invariant interest points

How can we independently select interest points in each image, such that the detections are repeatable across different scales?
Automatic Scale Selection

How to find corresponding patch sizes, with only one image in hand?

Automatic Scale Selection

- Function responses for increasing scale (scale signature)

Automatic Scale Selection

Intuition:

- Find scale that gives local maxima of some function \( f \) in both position and scale.

Automatic Scale Selection

- Function responses for increasing scale (scale signature)
Automatic Scale Selection
• Function responses for increasing scale (scale signature)

What can be the "signature" function?

Blob detection in 2D
Laplacian of Gaussian: Circularly symmetric operator for blob detection in 2D
\[ \nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \]

Blob detection in 2D: scale selection
Laplacian-of-Gaussian = "blob" detector
\[ \nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2} \]

Blob detection in 2D
We define the characteristic scale as the scale that produces peak of Laplacian response
Example

Original image at 1/3 the size

Scaled down image

Original image

Scaled down image
Scale-invariant interest points

Interest points are local maxima in both position and scale.

Squared filter response maps

List of \((x, y, \sigma)\)

Recap so far: interest points

- Interest point detection
  - Harris corner detector
  - Laplacian of Gaussian, automatic scale selection

Technical detail

We can approximate the Laplacian with a difference of Gaussians; more efficient to implement.

\[ L = \sigma^2 \left( G_x(x, y, \sigma) + G_y(x, y, \sigma) \right) \]  

(Laplacian)

\[ \text{DoG} = G(x, y, k\sigma) - G(x, y, \sigma) \]  

(Difference of Gaussians)

\[ I(k\sigma) = I(x) \]

\[ I(\sigma) - I(k\sigma) \]
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

Geometric transformations

Photometric transformations

Raw patches as local descriptors

Scale Invariant Feature Transform (SIFT) descriptor [Lowe 2004]
- Use histograms to bin pixels within sub-patches according to their orientation.

Scale Invariant Feature Transform (SIFT) descriptor [Lowe 2004]
- Interest points and their scales and orientations (random subset of 50)

http://www.vlfeat.org/overview/sift.html
Making descriptor rotation invariant

• Rotate patch according to its dominant gradient orientation
• This puts the patches into a canonical orientation.

SIFT descriptor [Lowe 2004]

• 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, e.g. http://www.vlfeat.org/overview/sift.html

Example

NASA Mars Rover images

Example

NASA Mars Rover images

with SIFT feature matches

Figure by Noah Snavely

SIFT properties

• Invariant to
  – Scale
  – Rotation

• Partially invariant to
  – Illumination changes
  – Camera viewpoint
  – Occlusion, clutter

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

Kristen Grauman
Matching local features

To generate candidate matches, find patches that have the most similar appearance (e.g., lowest SSD)
Simplest approach: compare them all, take the closest (or closest k, or within a thresholded distance)

Ambiguous matches

At what SSD value do we have a good match?
To add robustness to matching, can consider ratio: distance to best match / distance to second best match
If low, first match looks good.
If high, could be ambiguous match.

Matching SIFT Descriptors

- Nearest neighbor (Euclidean distance)
- Threshold ratio of nearest to 2\textsuperscript{nd} nearest descriptor

SIFT (preliminary) matches

http://www.vlfeat.org/overview/sift.html

http://www.vlfeat.org/overview/sift.html
Value of local (invariant) features

- Complexity reduction via selection of distinctive points
- Describe images, objects, parts without requiring segmentation
- Local character means robustness to clutter, occlusion
- Robustness: similar descriptors in spite of noise, blur, etc.

Applications of local invariant features

- Wide baseline stereo
- Motion tracking
- Panoramas
- Mobile robot navigation
- 3D reconstruction
- Recognition
- …

Automatic mosaicing

http://www.cs.ubc.ca/~mbrown/autostitch/autostitch.html

Wide baseline stereo

[Image from T. Tuytelaars ECCV 2006 tutorial]

Photo tourism [Snavely et al.]

Recognition of specific objects, scenes

Lowe 1999

Many current applications - 2017
Summary so far

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

• Invariant descriptors
  – Rotation according to dominant gradient direction
  – Histograms for robustness to small shifts and translations (SIFT descriptor)

Plan for today

• 1. Basics in feature extraction: filtering
• 2. Invariant local features
• 3. Recognizing object instances

Recognizing or retrieving specific objects

Example I: Visual search in feature films

Visually defined query

“Find this clock”

“Groundhog Day” [Ramnis, 1993]

Example II: Search photos on the web for particular places

Find these landmarks

...in these images and 1M more

Why is it difficult?

Want to find the object despite possibly large changes in scale, viewpoint, lighting and partial occlusion

We can’t expect to match such varied instances with a single global template...
Instance recognition: key new ideas

- **Visual words**
  - quantization, index, bags of words
- **Spatial verification**
  - affine; RANSAC, Hough

Indexing local features

- Each patch / region has a descriptor, which is a point in some high-dimensional feature space (e.g., SIFT)

Indexing local features

- When we see close points in feature space, we have similar descriptors, which indicates similar local content.

Visual words

- Map high-dimensional descriptors to tokens/words by quantizing the feature space

Visual words: main idea

- Extract some local features from a number of images …

For text documents, an efficient way to find all pages on which a word occurs is to use an index...

We want to find all images in which a feature occurs.

To use this idea, we'll need to map our features to "visual words".
Visual words: main idea

Each point is a local descriptor, e.g. SIFT vector.

Example: each group of patches belongs to the same visual word.
Inverted file index

- Database images are loaded into the index mapping words to image numbers

New query image is mapped to indices of database images that share a word.

Instance recognition: remaining issues

- How to summarize the content of an entire image? And gauge overall similarity?
- How large should the vocabulary be? How to perform quantization efficiently?
- Is having the same set of visual words enough to identify the object/scene? How to verify spatial agreement?

Analogy to documents

- Sensory, brain, visual, perception, retinal, cerebral cortex, eye, cell, optical nerve, image Hubel, Wiesel
- China, trade, surplus, commerce, exports, imports, US, yuan, bank, domestic, foreign, increase, trade, value

Bags of visual words

- Summarize entire image based on its distribution (histogram) of word occurrences.
- Analogous to bag of words representation commonly used for documents.
Comparing bags of words

- Rank frames by normalized scalar product between their (possibly weighted) occurrence counts—nearest neighbor search for similar images.

\[
sim(d_j, q) = \frac{d_j \cdot q}{\|d_j\| \|q\|}
\]

\[
= \frac{\sum_{i=1}^{V} d_j(i) \cdot q(i)}{\sqrt{\sum_{i=1}^{V} d_j(i)^2} \cdot \sqrt{\sum_{i=1}^{V} q(i)^2}}
\]

for vocabulary of \(V\) words

Instance recognition: remaining issues

- How to summarize the content of an entire image? And gauge overall similarity?
- How large should the vocabulary be? How to perform quantization efficiently?
- Is having the same set of visual words enough to identify the object/scene? How to verify spatial agreement?

Vocabulary size

Results for recognition task with 6347 images

Influence on performance, sparsity?

Nister & Stewenius, CVPR 2006

Vocabulary Trees: hierarchical clustering for large vocabularies

- Tree construction:

[Slide credit: David Nister
[Nister & Stewenius, CVPR’06]]
Vocabulary trees: complexity

Number of words given tree parameters: branching factor and number of levels
Word assignment cost vs. flat vocabulary

Visual words/bags of words

+ flexible to geometry / deformations / viewpoint
+ compact summary of image content
+ provides vector representation for sets
+ very good results in practice

- background and foreground mixed when bag covers whole image
- optimal vocabulary formation remains unclear
- basic model ignores geometry – must verify afterwards, or encode via features

Instance recognition: remaining issues

- How to summarize the content of an entire image? And gauge overall similarity?
- How large should the vocabulary be? How to perform quantization efficiently?
- Is having the same set of visual words enough to identify the object/scene? How to verify spatial agreement?

Spatial Verification

Both image pairs have many visual words in common.

Which matches better?

Only some of the matches are mutually consistent.
Spatial Verification: two basic strategies

- RANSAC

- Generalized Hough Transform

Outliers affect least squares fit

RANSAC

- RANdom Sample Consensus

- Approach: we want to avoid the impact of outliers, so let’s look for “inliers”, and use those only.

- Intuition: if an outlier is chosen to compute the current fit, then the resulting line won’t have much support from rest of the points.

RANSAC for line fitting

Repeat $N$ times:

- Draw $s$ points uniformly at random
- Fit line to these $s$ points
- Find inliers to this line among the remaining points (i.e., points whose distance from the line is less than $t$)
- If there are $d$ or more inliers, accept the line and refit using all inliers

RANSAC for line fitting example
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize-and-verify loop

Source: R. Raguram
Lana Lazebnik
RANSAC for line fitting example

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize-and-verify loop

Uncontaminated sample

1. Randomly select minimal subset of points
2. Hypothesize a model
3. Compute error function
4. Select points consistent with model
5. Repeat hypothesize-and-verify loop

That is an example fitting a model (line)...

What about fitting a transformation (translation)?

RANSAC example: Translation

Putative matches

Source: Rick Szeliski

RANSAC example: Translation

Select one match, count inliers

Source: Rick Szeliski
RANSAC example: Translation

Select one match, count inliers

RANSAC example: Translation

Find “average” translation vector

RANSAC: General form

- **RANSAC loop:**
  1. Randomly select a seed group of points on which to base transformation estimate
  2. Compute model from seed group
  3. Find inliers to this transformation
  4. If the number of inliers is sufficiently large, recompute estimate of model on all of the inliers
- Keep the model with the largest number of inliers

Fitting an affine transformation

\[
\begin{bmatrix}
x'_i \\
y'_i
\end{bmatrix} =
\begin{bmatrix}
m_1 & m_2 & x_i \\
m_3 & m_4 & y_i
\end{bmatrix} \begin{bmatrix}
x_i \\
y_i \\
f_i
\end{bmatrix}
\]

Approximates viewpoint changes for roughly planar objects and roughly orthographic cameras.

RANSAC verification

For matching specific scenes/objects, common to use an affine transformation for spatial verification.
Spatial Verification: two basic strategies

- **RANSAC**
  - Typically sort by BoW similarity as initial filter
  - Verify by checking support (inliers) for possible affine transformations
    - e.g., "success" if find an affine transformation with > N inlier correspondences

- **Generalized Hough Transform**
  - Let each matched feature cast a vote on location, scale, orientation of the model object
  - Verify parameters with enough votes

Voting

- It's not feasible to check all combinations of features by fitting a model to each possible subset.
- **Voting** is a general technique where we let the features vote for all models that are compatible with it.
  - Cycle through features, cast votes for model parameters.
  - Look for model parameters that receive a lot of votes.
- Noise & clutter features will cast votes too, but typically their votes should be inconsistent with the majority of "good" features.

Hough Transform for line fitting

- Given points that belong to a line, what is the line?
- How many lines are there?
- Which points belong to which lines?
- **Hough Transform** is a voting technique that can be used to answer all of these questions.
  
  Main idea:
  1. Record vote for each possible line on which each edge point lies.
  2. Look for lines that get many votes.

Difficulty of line fitting

Finding lines in an image: Hough space

Connection between image (x,y) and Hough (m,b) spaces

- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
  - given a set of points (x,y), find all (m,b) such that $y = mx + b$
Finding lines in an image: Hough space

Connection between image \((x,y)\) and Hough \((m,b)\) spaces
- A line in the image corresponds to a point in Hough space
- To go from image space to Hough space:
  - given a set of points \((x_i, y_i)\), find all \((m, b)\) such that \(y = mx + b\)
- What does a point \((x_0, y_0)\) in the image space map to?
  - Answer: the solutions of \(b = -x_0m + y_0\)
  - this is a line in Hough space

Finding lines in an image: Hough algorithm

How can we use this to find the most likely parameters \((m,b)\) for the most prominent line in the image space?
- Let each edge point in image space vote for a set of possible parameters in Hough space
- Accumulate votes in discrete set of bins; parameters with the most votes indicate line in image space.

Voting: Generalized Hough Transform

- If we use scale, rotation, and translation invariant local features, then each feature match gives an alignment hypothesis (for scale, translation, and orientation of model in image).

Gen Hough Transform details (Lowe’s system)
- Training phase: For each model feature, record 2D location, scale, and orientation of model (relative to normalized feature frame)
- Test phase: Let each match btw a test SIFT feature and a model feature vote in a 4D Hough space
  - Use broad bin sizes of 30 degrees for orientation, a factor of 2 for scale, and 0.25 times image size for location
  - Vote for two closest bins in each dimension
  - Find all bins with at least three votes and perform geometric verification
- Estimate least squares affine transformation
- Search for additional features that agree with the alignment

Objects recognized, Recognition in spite of occlusion

Background subtract for model boundaries

Example result

Gen Hough vs RANSAC

**GHT**
- Single correspondence -> vote for all consistent parameters
- Represents uncertainty in the model parameter space
- Linear complexity in number of correspondences and number of voting cells; beyond 4D vote space impractical
- Can handle high outlier ratio

**RANSAC**
- Minimal subset of correspondences to estimate model -> count inliers
- Represents uncertainty in image space
- Must search all data points to check for inliers each iteration
- Scales better to high-d parameter spaces

Kristen Grauman

Video Google System

1. Collect all words within query region
2. Inverted file index to find relevant frames
3. Compare word counts
4. Spatial verification

Sivic & Zisserman, ICCV 2003

- Demo online at:
  - [http://www.robots.ox.ac.uk/~vgg/research/vedge/index.html](http://www.robots.ox.ac.uk/~vgg/research/vedge/index.html)

Object retrieval with large vocabularies and fast spatial matching, Philbin et al., CVPR 2007

Query Results from 5k Flickr images (demo available for 100k set)

[Philbin CVPR'07]

Instance recognition applications

- Snap, pick, pay

[https://www.usatoday.com/videos/tech/2014/10/31/18261641/](https://www.usatoday.com/videos/tech/2014/10/31/18261641/)

Auto-annotate by connecting to content on Wikipedia!

World-scale mining of objects and events from community photo collections, Quack et al., CIVR 2008

Moulin Rouge
Tour Montparnasse
Colosseum
Old Town Square (Prague)

Slide credit: Kristen Grauman
**Example Applications**

Mobile tourist guide
- Self-localization
- Object/building recognition
- Photo/video augmentation

**Web Demo: Movie Poster Recognition**

50,000 movie posters indexed
Query-by-image from mobile phone available in Switzerland


**Recognition via feature matching+spatial verification**

**Pros:**
- Effective when we are able to find reliable features within clutter
- Great results for matching specific instances

**Cons:**
- Scaling with number of models
- Spatial verification as post-processing – not seamless, expensive for large-scale problems
- Not suited for category recognition.

**Summary (Part 3)**

- **Matching local invariant features**
  - To find specific objects and scenes.
- **Bag of words** representation: quantize feature space to make discrete set of visual words
  - Summarize image by distribution of words
  - Index individual words
- **Inverted index**: pre-compute index to enable faster search at query time
- **Recognition of instances via alignment**: matching local features followed by spatial verification
  - Robust fitting: RANSAC, GHT

**Coming up**

- Today - sign sheet if not registered / on wait list
- Read assigned papers, review 2
  - Don’t be afraid of the ImageNet IJCV paper!
- Assignment 1 out now, due Sept 22