Lecture 16: Bag-of-words models
Large-scale image matching

Turn 1,000,000 images of Rome...
...into 3D models

Colosseum

St. Peter’s Basilica

Trevi Fountain
Image matching

• Brute force approach:
  
  • 250,000 images $\rightarrow$ $\sim$ 31 billion image pairs
    – 2 pairs per second $\rightarrow$ 1 year on 500 machines

  • 1,000,000 images $\rightarrow$ 500 billion pairs
    – 15 years on 500 machines
Image matching

• For city-sized datasets, fewer than 0.1% of image pairs actually match

• Key idea: only consider *likely* matches

• How do we know if a match is likely?

• Solution: use fast global similarity measures
  – For example, a *bag-of-words* representation
Object → Bag of ‘words’
Origin 1: Texture Recognition

Example textures (from Wikipedia)
Origin 1: Texture recognition

- Texture is characterized by the repetition of basic elements or *textons*
- For stochastic textures, it is the identity of the textons, not their spatial arrangement, that matters

Origin 1: Texture recognition

Origin 2: Bag-of-words models

- Orderless document representation: frequencies of words from a dictionary  
  Salton & McGill (1983)
Origin 2: Bag-of-words models

Origin 2: Bag-of-words models


US Presidential Speeches Tag Cloud
http://chir.ag/phernalia/preztags/
Origin 2: Bag-of-words models

- Orderless document representation: frequencies of words from a dictionary  
  Salton & McGill (1983)
Bags of features for object recognition

- Works pretty well for image-level classification and for recognizing object *instances*

Csurka et al. (2004), Willamowski et al. (2005), Grauman & Darrell (2005), Sivic et al. (2003, 2005)
Bags of features for object recognition

Caltech6 dataset

<table>
<thead>
<tr>
<th>class</th>
<th>bag of features</th>
<th>bag of features</th>
<th>Parts-and-shape model</th>
</tr>
</thead>
<tbody>
<tr>
<td>airplanes</td>
<td>98.8</td>
<td>97.1</td>
<td>90.2</td>
</tr>
<tr>
<td>cars (rear)</td>
<td>98.3</td>
<td>98.6</td>
<td>90.3</td>
</tr>
<tr>
<td>cars (side)</td>
<td>95.0</td>
<td>87.3</td>
<td>88.5</td>
</tr>
<tr>
<td>faces</td>
<td>100</td>
<td>99.3</td>
<td>96.4</td>
</tr>
<tr>
<td>motorbikes</td>
<td>98.5</td>
<td>98.0</td>
<td>92.5</td>
</tr>
<tr>
<td>spotted cats</td>
<td>97.0</td>
<td>—</td>
<td>90.0</td>
</tr>
</tbody>
</table>
Images as histograms of visual words

- Inspired by ideas from text retrieval
  - [Sivic and Zisserman, ICCV 2003]
Bag of features: outline

1. Extract features
Bag of features: outline

1. Extract features
2. Learn “visual vocabulary”
Bag of features: outline

1. Extract features
2. Learn “visual vocabulary”
3. Quantize features using visual vocabulary
Bag of features: outline

1. Extract features
2. Learn “visual vocabulary”
3. Quantize features using visual vocabulary
4. Represent images by frequencies of “visual words”
1. Feature extraction

Compute SIFT descriptor
[Low '99]

Normalize patch

Detect patches
[Mikojaczyk and Schmid '02]
[Mata, Chum, Urban & Pajdla, '02]
[Sivic & Zisserman, '03]

Slide credit: Josef Sivic
1. Feature extraction
2. Learning the visual vocabulary
2. Learning the visual vocabulary

Clustering
2. Learning the visual vocabulary

Clustering

Visual vocabulary

Clustering
K-means clustering

- Want to minimize sum of squared Euclidean distances between points $x_i$ and their nearest cluster centers $m_k$

$$D(X, M) = \sum_{cluster_k} \sum_{point i in cluster k} (x_i - m_k)^2$$

Algorithm:

- Randomly initialize K cluster centers
- Iterate until convergence:
  - Assign each data point to the nearest center
  - Recompute each cluster center as the mean of all points assigned to it
Example visual vocabulary

Fei-Fei et al. 2005
Image patch examples of visual words
Visual vocabularies: Issues

• How to choose vocabulary size?
  • Too small: visual words not representative of all patches
  • Too large: quantization artifacts, overfitting

• Generative or discriminative learning?

• Computational efficiency
  • Vocabulary trees
    (Nister & Stewenius, 2006)
3. Image representation

frequency

... codewords...
Image classification

• Given the bag-of-features representations of images from different classes, how do we learn a model for distinguishing them?
Uses of BoW representation

• Treat as feature vector for standard classifier
  – e.g k-nearest neighbors, support vector machine

• Cluster BoW vectors over image collection
  – Discover visual themes
K nearest neighbors

- For a new point, find the k closest points from training data
- Labels of the k points “vote” to classify
- Works well provided there is lots of data and the distance function is good

$k = 5$

Source: D. Lowe
Linear classifiers

• Find linear function (hyperplane) to separate positive and negative examples

\[ \mathbf{x}_i \text{ positive: } \mathbf{x}_i \cdot \mathbf{w} + b \geq 0 \]

\[ \mathbf{x}_i \text{ negative : } \mathbf{x}_i \cdot \mathbf{w} + b < 0 \]

Which hyperplane is best?
Support vector machines

• Find hyperplane that maximizes the margin between the positive and negative examples

Support vector machines

• Find hyperplane that maximizes the margin between the positive and negative examples

\[ x_i \text{ positive } (y_i = 1) : \quad x_i \cdot w + b \geq 1 \]
\[ x_i \text{ negative } (y_i = -1) : \quad x_i \cdot w + b \leq -1 \]

For support, vectors, \[ x_i \cdot w + b = \pm 1 \]
Large-scale image matching

• Bag-of-words models have been useful in matching an image to a large database of object *instances*

11,400 images of game covers (Caltech games dataset) how do I find this image in the database?
Large-scale image search

• Build the database:
  – Extract features from the database images
  – Learn a vocabulary using k-means (typical k: 100,000)
  – Compute *weights* for each word
  – Create an inverted file mapping words ➔ images
Weighting the words

• Just as with text, some visual words are more discriminative than others

  the, and, or vs. cow, AT&T, Cher

• the bigger fraction of the documents a word appears in, the less useful it is for matching
  – e.g., a word that appears in all documents is not helping us
TF-IDF weighting

• Instead of computing a regular histogram distance, we’ll weight each word by it’s inverse document frequency

inverse document frequency (IDF) of word \( j = \log \frac{\text{number of documents}}{\text{number of documents in which } j \text{ appears}} \)
TF-IDF weighting

• To compute the value of bin $j$ in image $l$:

$$\text{term frequency of } j \text{ in } l \times \text{inverse document frequency of } j$$
Inverted file

• Each image has ~1,000 features
• We have ~1,000,000 visual words
  → each histogram is extremely sparse (mostly zeros)

• Inverted file
  – mapping from words to documents

```json
"a": {2}
"banana": {2}
"is": {0, 1, 2}
"it": {0, 1, 2}
"what": {0, 1}
```
Inverted file

• Can quickly use the inverted file to compute similarity between a new image and all the images in the database
  – Only consider database images whose bins overlap the query image
Large-scale image search

• Pros:
  – Works well for CD covers, movie posters
  – Real-time performance possible

real-time retrieval from a database of 40,000 CD covers
Nister & Stewenius, *Scalable Recognition with a Vocabulary Tree*
Image matching
Faster image matching

1. For each image, find the 40 most similar vectors
   (reduces # comparisons from \( \sim n^2/2 \) to <40n)
2. Do detailed SIFT matching to verify each pair
   (this is the most time-consuming part)
3. Use *query expansion* to densify the graph
Example bag-of-words matches

Query image

Top 16 matches
Example bag-of-words matches

Query image

Top 16 matches
Example bag-of-words matches

Query image

Top 16 matches
Matching - Round 1

Initial graph after matching each image to top 10 other images
Query Expansion

Chum et al, ICCV 2007
Query Expansion

Chum et al, ICCV 2007
Query Expansion

Chum et al, ICCV 2007
Query Expansion

Chum et al, ICCV 2007
Query Expansion

Chum et al, ICCV 2007
Matching - Round 1
Matching - Round 2
Matching - Round 3
Matching - Round 4

![Graph and chart showing matching progress over rounds.

- Pairs Verified
- Matched Pairs Found

Legend:
- Red line: Pairs Verified
- Black dotted line: Matched Pairs Found

Rounds of Matching: 1 to 6]
Matching - Round 5

[Graph showing network and line chart]

- Red line: Pairs Verified
- Black dashed line: Matched Pairs Found

Rounds of Matching

1  2  3  4  5  6
Matching - Round 6
## Matching Statistics

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Size</th>
<th>Matches possible</th>
<th>Matches Tried</th>
<th>Matches Found</th>
<th>Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Dubrovnik</td>
<td>58K</td>
<td>1.6 Billion</td>
<td>2.6M</td>
<td>0.5M</td>
<td>5 hrs</td>
</tr>
<tr>
<td>Rome</td>
<td>150K</td>
<td>11.2 Billion</td>
<td>8.8M</td>
<td>2.7M</td>
<td>13 hrs</td>
</tr>
<tr>
<td>Venice</td>
<td>250K</td>
<td>31.2 Billion</td>
<td>35.5M</td>
<td>6.2M</td>
<td>27 hrs</td>
</tr>
</tbody>
</table>
What about spatial info?