Bag-of-words models

Object \rightarrow Bag of ‘words’

Bag of Words Models

Adapted from slides by Rob Fergus and Svetlana Lazebnik
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 2: Bag-of-words models

• Orderless document representation: frequencies of words from a dictionary
  Salton & McGill (1983)

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


US Presidential Speeches Tag Cloud
http://chir.ag/phernalia/preztags/
Bags of features for object recognition

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

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><strong>98.8</strong></td>
<td>97.1</td>
<td>90.2</td>
</tr>
<tr>
<td>cars (rear)</td>
<td>98.3</td>
<td><strong>98.6</strong></td>
<td>90.3</td>
</tr>
<tr>
<td>cars (side)</td>
<td><strong>95.0</strong></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><strong>98.5</strong></td>
<td>98.0</td>
<td>92.5</td>
</tr>
<tr>
<td>spotted cats</td>
<td><strong>97.0</strong></td>
<td>—</td>
<td>90.0</td>
</tr>
</tbody>
</table>
Bag of features

- First, take a bunch of images, extract features, and build up a “dictionary” or “visual vocabulary” – a list of common features.

- Given a new image, extract features and build a histogram – for each feature, find the closest visual word in the dictionary.

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

Regular grid
- Vogel & Schiele, 2003
- Fei-Fei & Perona, 2005
1. Feature extraction

Regular grid
- Vogel & Schiele, 2003
- Fei-Fei & Perona, 2005

Interest point detector
- Csurka et al. 2004
- Fei-Fei & Perona, 2005
- Sivic et al. 2005

Other methods
- Random sampling (Vidal-Naquet & Ullman, 2002)
- Segmentation-based patches (Barnard et al. 2003)
2. Learning the visual vocabulary

Slide credit: Josef Sivic
2. Learning the visual vocabulary

K-means clustering

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

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

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
From clustering to vector quantization

- Clustering is a common method for learning a visual vocabulary or codebook
  - Unsupervised learning process
  - Each cluster center produced by k-means becomes a codevector
  - Codebook can be learned on separate training set
  - Provided the training set is sufficiently representative, the codebook will be “universal”

- The codebook is used for quantizing features
  - A vector quantizer takes a feature vector and maps it to the index of the nearest codevector in a codebook
  - Codebook = visual vocabulary
  - Codevector = visual word

Example visual vocabulary

Fei-Fei et al. 2005
Image patch examples of visual words

Sivic et al. 2005

Visual vocabularies: Issues

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

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

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

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 $j = \frac{\log \frac{\text{number of documents}}{\text{number of documents in which } j \text{ appears}}}{\text{number of documents}}$

TF-IDF weighting

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

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

• Each image has ~1,000 features
• We have ~100,000 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

• Cons:
  – not as accurate as per-image-pair feature matching
  – performance degrades as the database grows

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*
Large-scale image matching

Turn 1,000,000 images of Rome...

...into 3D models

Colosseum

St. Peter’s Basilica

Trevi Fountain
Large-scale image matching

• How can we match 1,000,000 images to each other?

• Brute force approach: 500,000,000,000 pairs
  – won’t scale

• Better approach: use bag-of-words technique to find likely matches

• For each image, find the top M scoring other images, do detailed SIFT matching with those

Example bag-of-words matches
Example bag-of-words matches

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?