Object Recognition, Detection, Categorization

A single Goal with Variations

Find evidence for familiar objects in a given image

- Recognition - Find evidence for specific object(s) that you have information about (Model, Library)
- Categorization - Find evidence for objects belonging to specific classes. Say which.
  - Detection - Find evidence for some object from a specific class. Say where it is.
  - Classification - Find evidence for some object from a specific class. No need to say where.
- Semantic segmentation - Find all objects in the image and their location. Classify every pixel.
Image Variability

The image depends on many factors

- Object category
- Objects’ specific shape (within category), position vs. camera, color, reflection properties, occlusions
- Illumination sources’ location, light distribution, color

A solution based on synthesis

- Suppose we know how to synthesize an image
  \[ I(\text{Shapes, positions, illumination,...}) \]
- Then, we can try to solve the inverse problem
  \[ \arg \min_{\text{Shape, position, camera param.}} \| I_{\text{given}} - I(\text{Shape, position, camera param.}) \| \]

**Difficulties**

- Models (shape, color) & imaging process are complex, unknown
- The scene (and image) contains much irrelevant data, and solving for all and for unimportant variables is difficult
- The inverse problem is multivalued & hard (non-convex)

- Not done **fully** in practice
Simple principles/approaches/tricks that help

**Principle I:** If possible, *make the problem easier*
- Control relative position and illumination
- Limit variability
  - OCR with known font
  - Train for your own handwriting
  - Ask the person to look directly at the camera

**Simple principle II:**

**Selection:** solve separately for image parts
- Moving window
- Segmentation
- Attention
Principle III:
Use Effective Representation for image comparison:
- Informative for discrimination
- Insensitive\Invariant to irrelevant parameters.
  (illumination, pose, compression effects, ...)
- Efficient

Examples
- Intensities, normalized intensities
- Edges, Gradient directions
- Histograms
- Bag of words (e.g. textons)

Often, not enough but serves as basis

A sample of typical approaches

- Comparing intensities using subspaces
- Recognizing binary objects using tailored features
- Compensating for pose change and verifying

-------------------------------------

- Using parts and relative positions
- Using parts with no relative positions
- Brief comments about other methods
How to handle image variability?

*Answer 1:* Approximately enumerate all possibilities

**Algorithm 1: Intensity subspaces**

Basic approach (template matching, correlation):
* Model: Represent object by its intensity image (N)
* Algorithm:
  - for all S subimages, find SSD(model, subimage)
  - Decide "detection" is the SSD is small enough

Pros: optimal for signal+white noise model, Fast O(SN)
Cons: sensitive to pose, illumination

Suggestion:
- Use a set of many M models with different pose, illuminations.
- But: SSD with many models is costly O(SNM)
- Even slower if we want to recognize from library
Algorithm 1: The subspace idea

- Idea: Use efficient image representation. N = #pixels
  \[ I_i(x, y) \approx \phi_0(x, y) + \sum_{j=1}^{N'<N} c_j \phi_j(x, y) \]
  \[ I_i(x, y) \approx c_{iN'} \]
- For many such collections, \( N' \ll (x, y) \)

Why?

- Reason 1: Images are Smooth & similar
- Reason 2:
  - All images of Lambertian object under different illumination (no shadows) are in a 3D subspace. Proof: done in lecture ~3
  - With attached shadows, the effective dimension is small.

Which subspace?

Principle components analysis (PCA)

- Approximation of a set of vectors \( \{v_i\}_{1}^{m} \)
  \[ v_i \approx \hat{v}_i = \phi_0 + \sum_{j=1}^{N'<N} c_j \phi_j \]
- Let \( \{\phi_j\} \) be orthonormal.
- Approximation error
  \[ \|v_i - \hat{v}_i\|^2 = \|\sum_{j=N'+1}^{N} c_j \phi_j\|^2 = \sum_{j=N'+1}^{N} |c_j|^2 \]
- Use \( \phi_0 \neq 0, \quad \hat{\phi}_0 = \bar{v} \) (to get "good properties")
- Minimizing error:
  \[ \arg \min_{\{\phi_j\}} \sum_{j=N'+1}^{N} |c_j|^2 = \arg \max_{\{\phi_j\}} \sum_{j=1}^{N'} |c_j|^2 = \sum_{j=1}^{N'} |\phi_j^T (v_i - \phi_0)|^2 \]
Principle components analysis (cont.)

Optimizing for all images in \( \{v_i\} \) Let \( \phi_0 = \bar{v} \)

\[
\arg \max_{\{\phi_i\}} \left( \frac{1}{m} \sum_{i=1}^m \sum_{j=1}^{N'} e_{ij}^2 \right) = \sum_{i=1}^{N'} \sum_{j=1}^{N'} \left( \phi_i \right)^T \frac{1}{m} (v_i - \bar{v}) (v_i - \bar{v})^T (\phi_j)
\]

\[
= \sum_{j=1}^{N'} (\phi_j)^T \Sigma(\phi_j) \quad ; \quad (\phi_j)^T (\phi_j) = 1
\]

For \( N' = 1 \), using Lagrange multipliers,

\[
\max_{\{\phi_i\}} \left( \phi_i \right)^T \Sigma(\phi_i) - \alpha \left( \left( \phi_i \right)^T (\phi_i) - 1 \right)
\]

\[\rightarrow \quad \Sigma \phi_i - \alpha \phi_i = 0\]

\[\rightarrow \quad \phi_i \] is an e.vector of \( \Sigma \). Which one with largest e.value.

\[\rightarrow \quad \text{For next one... max} \sum_{j=1}^2 (\phi_j)^T \Sigma(\phi_j) - \alpha \left( \left( \phi_j \right)^T (\phi_j) - 1 \right) - \beta \left( \left( \phi_2 \right)^T (\phi_2) \right)\]

All the \( \{\phi_i\} \), denoted “principle components” are e.vectors of \( \Sigma \) associated with largest e.values.

Principle components analysis (cont.)

- The subspace spanned by the \( N' \) components is often a good characterization of the set \( \{v_i\} \).
- The PCA is the most common dimension reduction tool
  - Useful in many domains
    - Compression
    - Search
- If \( \Sigma \) is exact:
  - PCA = Karhunen Loeve transform,
  - coefficients \( \{e_i\} \) are uncorrelated
- Other subspaces: LDA, DCT
Recognition with PCA

PCA example: Eigenfaces

models

average + PCA basis

Recognition with PCA

- **Algorithm a: common subspace**
  - Preprocessing: get model examples \( \{v_i\} \), Find one PCA subspace for all models
  - Represent every model images by PCA coefficients \( \{c_{ij}\} \)
  - For every sub-image (k-th) find PCA coefficients \( \{d_{ij}\} \) and find nearest model using approximate SSD

\[
SSD_{ik} \approx \sum_{j=1}^{N} (c_{ij} - d_{ij})^2
\]

- **Algorithm b: Different subspaces**
  - Preprocessing: build different PCA subspace for every model
  - Represent the sub-images \( \{d_{ij}\} \) in all subspaces
  - Choose the subspace that gives best approximation
Inside the subspace

- **Limitations:** can’t handle occlusion or variable background (global), not very effective
- **Conclusions:**
  - Images of the same object are in low dimension space/manifold,
  - PCA is old but still very useful.

How to handle image variability?

**Answer 1:** Approximately enumerate all possibilities

**Answer 2:** Use Invariants
Alg 2: Recognizing binary objects using features/invariants

- Context: recognizing objects discriminated from background
- Evaluate features:
  - Tailored: perimeter, area, # holes, # concavities.
  - Generic: e.g. moments
    \[ m_{pq} = \sum_{(x,y)} x^p y^q I(x, y) \]
  - All can be made invariant to translation, rotation, etc.
  - Global
- Use the examples vector to learn (any learning algorithm)
- Pros: compact & fast, Cons: objects are not isolated

\[ Alg\ 2\ (cont):\ Moment\ invariants \]

- Basic moments
  \[ m_{pq} = \sum_{(x,y)} x^p y^q I(x, y) \]
  Meaning:
  \[ m_{00}, m_{10}, m_{01}, \ldots \]
- Invariance to translation
  Center of mass
  \[ \bar{x} = \frac{m_{10}}{m_{00}}, \quad \bar{y} = \frac{m_{01}}{m_{00}} \]
  Central moments
  \[ \mu_{00} = m_{00}, \mu_{10} = \mu_{01} = 0, \mu_{11} = m_{10} - \bar{x}m_{01} \]
- Invariance to rotation
  Matrix of central moments
  \[ \begin{bmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{bmatrix} \]
  It is diagonal in the basis of the e.vectors
  The e.values are rotation invariants
  \[ \begin{bmatrix} \lambda_1 & 0 \\ 0 & \lambda_2 \end{bmatrix} \]
More on invariants

(Pose) invariant - a property that is measured in an image and does not depend on pose.

Model: A set of invariant features
- Euclidean transformation: area, angle
- Affine transformation: length ratio (on same line)
- Perspective Transformation: Cross ratio

Limitations
- No invariants for true 3D objects
- Some grouping needed
- Invariants amplify noise

How to handle image variability?

Answer 1: Approximately enumerate all possibilities
Answer 2: Use Invariants
Answer 3: hypothesize pose (illumination) and test
Alg 3: Geometric alignment

• **Construct a model** that can be used for synthesizing image related description (e.g. an edge image)

• **Recognition (by RANSAC):** Given an image
  - Hypothesize point matches between image and model features.
  - Synthesize image hypothesis
  - Verify (evaluate hypothesis) by image comparison.
    • e.g. grad. direction.

Repeat

Alg 3: Geometric alignment

• **Past:** popular with geometric models

• **Modern variations**
  - Face recognition applications
  - Iterative alignment of 3D model
  - Part based models
  - (-) Needs a model, takes time
  - (+) Captures fine differences
How to handle image variability?

Answer 1: Approximately enumerate all possibilities
Answer 2: Use Invariants
Answer 3: hypothesize pose (illumination) and test
Answer 4: Use learning

Learning - creating classifiers from examples

- Examples - images or descriptions extracted from images (e.g. PCA).
- The examples express object variability due to pose/illumination and due to non-rigid change and change within class

- Recognition is not representation
  - Not everything must be represented
  - Example: PCA is non-optimal
  - Representing only discriminative information
Alg 4: Viola Jones face detection

• A part based alg.
• Uses Appearance
  – How the part looks like
• and Shape
  – What is the relative part position

• Based on the boosting learning process
  + Several computational tricks
• (was) Very fast (15 fps on Pentium3)

• Widely adopted and implemented in openCV, cameras

Alg 4: Viola Jones face detection (cont)

• Weak classifiers
  - based on single Haar-like features
  - Feature: brightness difference in a mask
  - Masks are defined relative to sub-window

• Faces share appearances that make these feature informative
  - Dark eyes
  - Bright nose

• 45,396 weak classifier. Which ?
AdaBoost (reminder)

- Adaboost starts with a uniform distribution of “weights” over training examples.
- Select the classifier with the lowest weighted error (i.e. a “weak” classifier)
- Increase the weights on the training examples that were misclassified.
- (Repeat)
- At the end, carefully make a linear combination of the weak classifiers obtained at all iterations.

$$h_{reg}(x) = \begin{cases} 1 & \alpha_i h_i(x) + \cdots \geq \alpha_i + \cdots \\ 0 & \text{otherwise} \end{cases}$$

Source: Qing Chen, Discover Lab, University of Ottawa

Alg 4: Viola Jones face detection (cont)

- **Strong classifier**
  - Trained from positive/negative examples
  - Every feature - fast to evaluate
  - Uses Adaboost to combine features

- **How many weak classifiers?**
  - More - slow but accurate
  - With 200 weak classifiers
  - Nice ROC, 0.7 sec.

Receiver Operating Curve
Alg 4: Viola Jones face detection (cont)

Making it faster - I

- **Integral image** for faster evaluation of the weak classifiers

  - Formal definition: \( II(x, y) = \sum_{x', y'} I(x', y') \)
  
  - Preparing the integral image: \( O(N) \)

  - Usage: calculating features in constant time

  - E.g. \( \sum_{(x, y) \in O} I(x, y) = II(4) + II(1) - II(2) - II(3) \)

Alg 4: Viola Jones face detection (cont)

Making it faster - II

- Observation: in most places, 200 weak classifiers are much more than needed to decide “no”.

- **A Cascade of classifiers**
  - The first uses 2 classifiers. Rejects 40%
  - The second uses 5 classifiers. Rejects 60%
  
  ....

- The last ones use 200 classifiers
- Design is complex.
Alg 4: Viola Jones face detection (cont)

- Experiments
  - Training 4619 (+), 10,000 (-), 24 x 24 images
  - About 92% detection, with 1/image false detection.
  - As accurate as before, and much faster

![Image of face detection experiments]

Alg 4’ (variation): Sharing weak classifiers

- 50 training samples/class
- 29 object classes

- Some work on sharing recognizers …

![Graph showing the relationship between number of object classes and number of features]

Torralba, Murphy, Freeman. CVPR 2004, PAMI 2007
**Parts – Generative approach**

- Object = a set of parts

- Model: A likelihood function depending
  - On *relative location* of parts
  - On *appearance* of part

![Figure from Fischler & Elschlager 73](image)

**Alg 5: The Constellation model**

- Model: A likelihood function depending on the *appearance* of parts and their *relative location*

- How to represent appearance
  - PCA

- How to model location
  - For all part pairs
  - Gaussians relatively to nominal relative location,

- How to choose the parts
  - The full likelihood (information gain of parts)

- Decision by maximum likelihood.
  - Contains also a model of the background

![Figure from Fischler & Elschlager 73](image)
The Constellation Model

Samples from appearance model

Pros: Can learn from weakly annotated images
Cons: convergence problems, slow runtime low accuracy

Different types of spatial constraints

- a) Constellation [13]
  - Fer tus et al. ’03
  - Fei-Fei et al. ’03
- b) Star shape [9, 14]
  - Crandall et al. ’05
  - Fergus et al. ’05
- c) b-fan (k = 2) [9]
- d) Tree [12]
  - Crandall et al. ’05
- e) Bag of features [10, 21]
  - Csurka ’04
  - Vasconcelos ’00
- f) Hierarchy [4]
  - Bouchard & Triggs ’05
- g) Sparse flexible model
  - Carneiro & Lowe ’06

source: Carneiro and Lowe
Alg 6: Histogram of Gradients (HOG+)

Another learning based algorithm.

- In each 8x8 block calculate a histogram of gradient orientations, concatenate, use as feature vectors for SVM training.
- Works nice for face, pedestrian detector

Alg 7: Discriminatively trained deformable part models

- Combination
  - Tree based spatial model
  - HOG appearance in 2-scales
  - Whole object+Parts

- Location:
  - start model
  - Deformation allowed

- Score = appearance score - deformation cost

- Training by SVM

Felzenszwalb, Mcallester, Ramanan, CVPR 2008
Some results

Alg. 8: Bag-of-features models

Many slides adapted from Fei-Fei Li, Rob Fergus, Antonio Torralba, and S.Lazebnik
Origin 1: Bag-of-words models


Origin 2: Texture recognition

- Texture is characterized by the repetition of basic elements or textons
- For a large subset (“stochastic”) textures, it is the identity of the textons, not their spatial arrangement, that matters
Origin 2: Texture recognition

Bag of features – Basic Algorithm

**Training**
1. Extract features
2. Learn “visual vocabulary”
3. Quantize features using visual vocabulary
4. Represent images as histogram of “visual words”
5. Train a (multiclass) classifier

**Classification:** Extract features, quantize them, represent image as distribution, classify.
Bag of features

1. Extract features – ideal situation

Feature extraction

Extract small regions
- Regular grid
- Interest point detector
- Random sampling
- Segmentation based patches

Represent each window
- Grad direction histograms – SIFT etc.
- PCA coefficients
- Edge maps
Bag of features

1. Extract features
2. Learn “visual vocabulary” – often k-means
3. Quantize features using visual vocabulary
4. Represent images as histogram of “visual words”
5. Train a distribution (multiclass) classifier

visual vocabulary - example

Fei-Fei et al. 2005
Bag of features

1. Extract features
2. Learn “visual vocabulary”
3. Quantize features using visual vocabulary
4. Represent images as histogram of “visual words”
5. Train a (multiclass) classifier
   - Discriminative: e.g. SVM
   - Generative: e.g. Naïve Bayes
Performance: confusion matrix (SVM)

<table>
<thead>
<tr>
<th></th>
<th>faces</th>
<th>buildings</th>
<th>trees</th>
<th>cars</th>
<th>phones</th>
<th>bikes</th>
<th>books</th>
</tr>
</thead>
<tbody>
<tr>
<td>faces</td>
<td>98</td>
<td>14</td>
<td>10</td>
<td>10</td>
<td>34</td>
<td>0</td>
<td>13</td>
</tr>
<tr>
<td>buildings</td>
<td>1</td>
<td>63</td>
<td>3</td>
<td>0</td>
<td>3</td>
<td>1</td>
<td>6</td>
</tr>
<tr>
<td>trees</td>
<td>1</td>
<td>10</td>
<td>81</td>
<td>1</td>
<td>0</td>
<td>6</td>
<td>0</td>
</tr>
<tr>
<td>cars</td>
<td>0</td>
<td>1</td>
<td>1</td>
<td>85</td>
<td>5</td>
<td>0</td>
<td>5</td>
</tr>
<tr>
<td>phones</td>
<td>0</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>55</td>
<td>2</td>
<td>3</td>
</tr>
<tr>
<td>bikes</td>
<td>0</td>
<td>4</td>
<td>1</td>
<td>0</td>
<td>1</td>
<td>91</td>
<td>0</td>
</tr>
<tr>
<td>books</td>
<td>0</td>
<td>3</td>
<td>0</td>
<td>1</td>
<td>2</td>
<td>0</td>
<td>73</td>
</tr>
<tr>
<td>error rate</td>
<td>2</td>
<td>27</td>
<td>19</td>
<td>15</td>
<td>45</td>
<td>9</td>
<td>27</td>
</tr>
</tbody>
</table>

mean rank 1.04 1.77 1.28 1.30 1.83 1.09 1.39

Sparse vs. Dense sampling

**Sparse**
- Sampling based on key points
- Different interest operator help
- More invariant & compact
- Better for finding specific objects

**Dense**
- Densely spatial sampling
- patches at all image scales.
- Better for categorization

Jurie and Triggs, Creating Efficient Codebooks for Visual Recognition, ICCV05
Which Clustering?

- What is the patch distribution?
- Depends on sampling method!
- Experiment:
  - Generate a set of patches from natural images
  - Fill the densest region with uniform size balls.

Jurie and Triggs, Creating Efficient Codebooks for Visual Recognition, ICCV05

Which Clustering?

- Which “Visual words” are important
- Most frequent - uniform, smooth, edge like, generic and noninformative.
- Least frequent - discriminative but rare - so less important
- Medium density - most informative
Which Clustering?

• So what’s wrong with K-means?
  
  • K-means will put most centers near the distribution peak,
  • where patches are non-informative

Jurie and Triggs, Creating Efficient Codebooks for Visual Recognition, ICCV05

Proposed Jurie-Triggs clustering

• Repeat several times:
  - Draw N patches uniformly & randomly from the unlabeled part of the dataset.
  - \( C_k \) = maximal density position, computed by a mean shift estimator with radius \( r \).
  - Eliminate patches that are within the radius of this center.

• Output: \( \{ C_k \} \) as code-words

Animation by Yaniv Romano
Results

Introducing location information

1. Quantize to words
2. Build a spatial pyramid
3. For every word
   1. Calculate Histogram for every cell
   2. Find similarity for every Hist./cell
   3. Weighted merge
4. Calculate overall similarity by summing over all words
The spatial pyramid

Example: (3 words)

1. Find hypotheses of relatively stable to view (“canonical”) parts
2. Infer position and appearance of other canonical parts
3. Calculate score

Algorithm

Kind of alignment algorithms
Semantic segmentation

- Croquet
- Bocce
- Snowboarding
- Polo
- Sailing
- Badminton
- Rock Climbing
- Rowing

CNN

- Deep neural networks
- Convolutional neural networks
- A significant advantage over most methods
- Requires many examples & computing power

Current state of the art

Next slides …