Computer Vision - Lecture 9
Subspace Representations for Recognition

1.12.2009

Bastian Leibe
RWTH Aachen
http://www.umic.rwth-aachen.de/multimedia
leibe@umic.rwth-aachen.de

Course Outline

- Image Processing Basics
- Segmentation & Grouping
  - Graph-theoretic methods (remainder)
- Recognition
  - Global Representations
  - Subspace representations
- Object Categorization I
  - Sliding Window based Object Detection
- Local Features & Matching
- Object Categorization II
  - Part based Approaches
- 3D Reconstruction
- Motion and Tracking

Announcements

- Registering for the exam...
  - There has been some confusion regarding the procedure.
- Procedure
  - Master CS students had to register on Campus until last Friday.
  - Master MI students can start registering today.
  - Bachelor CS students MUST NOT register at the ZPA.
  - Diplom CS students do not have to register electronically (but can take the lecture as part of the exam "Praktische Informatik" or "Vertiefungslinie")
  - Foreign students
  - Sorry about this mess. It’s not us who made the rules... 😞

Announcements

- Exercise sheet 4 will be made available this afternoon
  - Subspace methods for object recognition [today’s topic]
  - Sliding-window object detection [Thursday’s topic]
  - The exercise will be next Tuesday.
  - Submit your results by Monday night.
- Application: Face detection & identification

Recap: Image Segmentation

- Goal: identify groups of pixels that go together

Recap: Images as Graphs

- Fully-connected graph
  - Node (vertex) for every pixel
  - Link between every pair of pixels, (p,q)
  - Affinity weight $w_{pq}$ for each link (edge)
  - $w_{pq}$ measures similarity
  - Similarity is inversely proportional to difference (in color and position...)
Recap: Normalized Cut (NCut)

- A minimum cut penalizes large segments
- This can be fixed by normalizing for size of segments
- The normalized cut cost is:
  \[ NCut(A, B) = \frac{cut(A, B)}{assoc(A, V)} - \frac{cut(A, B)}{assoc(B, V)} \]
  where \( assoc(A, V) \) is the sum of weights of all edges in \( V \) that touch \( A \)
- The exact solution is NP-hard but an approximation can be computed by solving a generalized eigenvalue problem.

Recap: NCuts: Overall Procedure

1. Construct a weighted graph \( G=(V,E) \) from an image.
2. Connect each pair of pixels, and assign graph edge weights.
   \[ W(i, j) = \text{Prob. that } i \text{ and } j \text{ belong to the same region.} \]
3. Solve \( (D-W)y = \lambda D y \) for the smallest few eigenvectors. This yields a continuous solution.
4. Threshold eigenvectors to get a discrete cut.
   - This is where the approximation is made (we’re not solving NP).
5. Recursively subdivide if NCut value is below a pre-specified value.

Recap: Energy Formulation

- Joint probability
  \[ P(x, y) = \prod_i \Phi(x_i, y_i) \prod_{i,j} \Psi(x_i, x_j) \]
- Maximizing the joint probability is the same as minimizing the log
  \[ \log P(x, y) = \sum_i \log \Phi(x_i, y_i) + \sum_{i,j} \log \Psi(x_i, x_j) \]
  \[ E(x, y) = \sum_i \Phi(x_i, y_i) + \sum_{i,j} \Psi(x_i, x_j) \]
- This is similar to free-energy problems in statistical mechanics (spin glass theory). We therefore draw the analogy and call \( E \) an energy function.
- \( \phi \) and \( \psi \) are called potentials.

Recap: Graph Cuts Energy Minimization

Regional bias example

Suppose \( I' \) and \( I'' \) are given "expected" intensities of object and background

\[ D_{I'}(s) \propto \exp \left( -\|I' - I'' \|^2 / 2\sigma^2 \right) \]
\[ D_{I''}(s) \propto \exp \left( -\|I'' - I' \|^2 / 2\sigma^2 \right) \]
Recap: Graph Cuts Energy Minimization

“expected” intensities of object and background can be re-estimated using EM-style optimization.

\[
D_s(s) = \exp\left(-\frac{1}{2} I_s - I' \| I' \| / 2 \sigma^2\right) \\
D_t(t) = \exp\left(-\frac{1}{2} I_t - I' \| I' \| / 2 \sigma^2\right)
\]

How Does it Work? The s-t-Mincut Problem

The s-t-Mincut Problem

What is an s-t-cut?

An s-t-cut \((S, T)\) divides the nodes between source and sink.

What is the cost of an s-t-cut?

Sum of cost of all edges going from \(S\) to \(T\)

\[5 + 2 + 9 = 16\]

What is the st-mincut?

Sum of cost of all edges going from \(S\) to \(T\)

\[2 + 1 + 4 = 7\]

How to Compute the s-t-Mincut?

Solve the dual maximum flow problem.

Compute the maximum flow between source and sink.

Min-cut/Max-flow Theorem

In every network, the maximum flow equals the cost of the st-mincut.

History of Maxflow Algorithms

<table>
<thead>
<tr>
<th>Year</th>
<th>Algorithm(s)</th>
<th>Authors</th>
<th>Notes</th>
</tr>
</thead>
<tbody>
<tr>
<td>1956</td>
<td>Ford-Fulkerson</td>
<td>O(nm)</td>
<td></td>
</tr>
<tr>
<td>1970</td>
<td>Dinic</td>
<td>O(nm^2)</td>
<td></td>
</tr>
<tr>
<td>1972</td>
<td>Edmonds &amp; Karp</td>
<td>O(nm^2)</td>
<td></td>
</tr>
<tr>
<td>1974</td>
<td>Dinic</td>
<td>O(nm^2)</td>
<td></td>
</tr>
<tr>
<td>1975</td>
<td>Dinic</td>
<td>O(nm^2)</td>
<td></td>
</tr>
<tr>
<td>1977</td>
<td>Dinic</td>
<td>O(nm^2)</td>
<td></td>
</tr>
<tr>
<td>1980</td>
<td>Goldsmith &amp; Karloff</td>
<td>O(nm log^2 n)</td>
<td></td>
</tr>
<tr>
<td>1985</td>
<td>Stoer &amp; Tarjan</td>
<td>O(m log^2 n)</td>
<td></td>
</tr>
<tr>
<td>1985</td>
<td>Goldberg &amp; Tarjan</td>
<td>O(nm log log n)</td>
<td></td>
</tr>
<tr>
<td>1987</td>
<td>Ahuja &amp; Orlin</td>
<td>O(mn log n)</td>
<td></td>
</tr>
<tr>
<td>1987</td>
<td>Ahuja &amp; Orlin</td>
<td>O(nm log^2 n)</td>
<td></td>
</tr>
<tr>
<td>1989</td>
<td>Ahuja et al.</td>
<td>O(nm log n)</td>
<td></td>
</tr>
<tr>
<td>1990</td>
<td>Ahuja et al.</td>
<td>O(nm log^2 n)</td>
<td></td>
</tr>
<tr>
<td>1990</td>
<td>Ahuja et al.</td>
<td>O(nm log n)</td>
<td></td>
</tr>
<tr>
<td>1992</td>
<td>King et al.</td>
<td>O(nm log^2 n)</td>
<td></td>
</tr>
<tr>
<td>1993</td>
<td>Goldberg &amp; Stein</td>
<td>O(nm log^{1.5} n)</td>
<td></td>
</tr>
<tr>
<td>1994</td>
<td>King et al.</td>
<td>O(nm log^{1.5} n)</td>
<td></td>
</tr>
<tr>
<td>1995</td>
<td>Goldberg &amp; Stein</td>
<td>O(nm log^{1.5} n)</td>
<td></td>
</tr>
<tr>
<td>1995</td>
<td>Goldberg &amp; Stein</td>
<td>O(nm log^{1.5} n)</td>
<td></td>
</tr>
<tr>
<td>1995</td>
<td>Goldberg &amp; Stein</td>
<td>O(nm log^{1.5} n)</td>
<td></td>
</tr>
<tr>
<td>1997</td>
<td>Goldberg &amp; Stein</td>
<td>O(nm log^{1.5} n)</td>
<td></td>
</tr>
</tbody>
</table>
Maxflow in Computer Vision

- Specialized algorithms for vision problems
  - Grid graphs
  - Low connectivity (m ~ O(n))
- Dual search tree augmenting path algorithm
  [Boykov and Kolmogorov PAMI 2004]
  - Finds approximate shortest augmenting paths efficiently
  - High worst-case time complexity
- Dual search tree augmenting path algorithm
  [Boykov and Kolmogorov PAMI 2004]
  - Finds approximate shortest augmenting paths efficiently
  - High worst-case time complexity
  - Empirically outperforms other algorithms on vision problems
  - Efficient code available on the web
  http://www.cs.ucl.ac.uk/staff/V.Kolmogorov/software.html

When Can s-t Graph Cuts Be Applied?

- s-t graph cuts can only globally minimize binary energies that are submodular.
- Non-submodular cases can still be addressed with some optimality guarantees.
  - Current research topic...

GraphCut Applications: “GrabCut”

- Interactive Image Segmentation [Boykov & Jolly, ICCV’01]
  - Rough region cues sufficient
  - Segmentation boundary can be extracted from edges
- Procedure
  - User marks foreground and background regions with a brush.
  - This is used to create an initial segmentation which can then be corrected by additional brush strokes.

GrabCut: Data Model

- Obtained from interactive user input
  - User marks foreground and background regions with a brush
  - Alternatively, user can specify a bounding box

GrabCut: Coherence Model

- Pairwise potentials: (“contrast sensitive Potts model”)
  \[ \psi(x, y) = \sum_{(i,j) \in C} \delta[x_i \neq x_j] e^{-\beta (A_{i,j} - \gamma)^2} \]
  - Penalty if labels of adjacent pixels are different
  - Contrast between pixels

Iterated Graph Cuts

- Energy after each iteration
- Result
- Color model (Mixture of Gaussians)
**Perceptual and Sensory Augmented Computing**

**Computer Vision WS 09/10**

**GrabCut: Example Results**

- This will be included in the next version of MS Office!

**Applications: Interactive 3D Segmentation**

**Improving Efficiency of Segmentation**

- Problem: Images contain many pixels
  - Even with efficient graph cuts, an MRF formulation has too many nodes for interactive results.
- Efficiency trick: Superpixels
  - Group together similar-looking pixels for efficiency of further processing.
  - Cheap, local oversegmentation
  - Important to ensure that superpixels do not cross boundaries
- Several different approaches possible
  - Superpixel code available here

**Superpixels for Pre-Segmentation**

**Summary: Graph Cuts Segmentation**

**Pros**
- Powerful technique, based on probabilistic model (MRF).
- Applicable for a wide range of problems.
- Very efficient algorithms available for vision problems.
- Becoming a de-facto standard for many segmentation tasks.

**Cons/Issues**
- Graph cuts can only solve a limited class of models
  - Submodular energy functions
  - Can capture only part of the expressiveness of MRFs
- Only approximate algorithms available for multi-label case

**Topics of This Lecture**

- Subspace Methods for Recognition
  - Motivation
- Principal Component Analysis (PCA)
  - Derivation
  - Object recognition with PCA
  - Eigenimages/Eigenfaces
  - Limitations
- Fisher’s Linear Discriminant Analysis (FLD/LDA)
  - Derivation
  - Fisherfaces for recognition
Recap: Appearance-Based Recognition

- Basic assumption
  - Objects can be represented by a set of images ("appearances").
  - For recognition, it is sufficient to just compare the 2D appearances.
  - No 3D model is needed.

⇒ Fundamental paradigm shift in the 90's

Representations for Recognition

- More generally, we want to obtain representations that are well-suited for
  - Recognizing a certain class of objects
  - Identifying individuals from that class (identification)

- How can we arrive at such a representation?
- Approach 1:
  - Come up with a brilliant idea and tweak it until it works.
  - Can we do this more systematically?

Example: The Space of All Face Images

- When viewed as vectors of pixel values, face images are extremely high-dimensional.
  - 100x100 image = 10,000 dimensions
- However, relatively few 10,000-dimensional vectors correspond to valid face images.
- We want to effectively model the subspace of face images.

Subspace Methods

- Images represented as points in a high-dim. vector space
- Valid images populate only a small fraction of the space
- Characterize subspace spanned by images
Subspace Methods

Principal Component Analysis

• Given: $N$ data points $x_1, \ldots, x_N$ in $\mathbb{R}^d$
• We want to find a new set of features that are linear combinations of original ones:
  \[ u(x_i) = u^T (x_i - \mu) \]
  ($\mu$: mean of data points)

• What unit vector $u$ in $\mathbb{R}^d$ captures the most variance of the data?

Remember: Fitting a Gaussian

• Mean and covariance matrix of data define a Gaussian model

Topics of This Lecture

• Subspace Methods for Recognition
  • Activation
• Principal Component Analysis (PCA)
  • Derivation
  • Object recognition with PCA
  • Eigenimages/Eigenfaces
  • Limitations
• Fisher’s Linear Discriminant Analysis (FLD/ LDA)
  • Derivation
  • Fishenfaces for recognition

Interpretation of PCA

• Compute eigenvectors of covariance $\Sigma$.
• Eigenvectors: main directions
• Eigenvalue: variance along eigenvector

Result: coordinate transform to best represent the variance of the data
Properties of PCA

- It can be shown that the mean square error between $x_i$ and its reconstruction using only $m$ principle eigenvectors is given by the expression:

$$\sum_{j=1}^{m} \lambda_j - \sum_{j=1}^{k} \lambda_j = \sum_{j=k+1}^{k} \lambda_j$$

- Interpretation
  - PCA minimizes reconstruction error
  - PCA maximizes variance of projection
  - Finds a more “natural” coordinate system for the sample data.

Projection and Reconstruction

- An $n$-pixel image $x \in \mathbb{R}^n$ can be projected to a low-dimensional feature space $y \in \mathbb{R}^m$ by

$$y = W^T x$$

- From $y \in \mathbb{R}^m$, the reconstruction of the point is $W^T y$

- The error of the reconstruction is $\|x - W^T W x\|$

Example: Object Representation

Object Detection by Distance TO Eigenspace

- Scan a window $\omega$ over the image and classify the window as object or non-object as follows:
  - Project window to subspace and reconstruct as earlier.
  - Compute the distance between $x$ and the reconstruction (reprojection error).
  - Local minima of distance over all image locations $\omega$ object locations
  - Repeat at different scales
  - Possibly normalize window intensity such that $|\omega|=1$.

Principal Component Analysis

- Objects are represented as coordinates in an $n$-dim. eigenspace.

- Example:
  - 3D space with points representing individual objects or a manifold representing parametric eigenspace (e.g., orientation, pose, illumination).

- Estimate parameters by finding the NN in the eigenspace

Obj. Identification by Distance IN Eigenspace
Parametric Eigenspace

- Object identification / pose estimation
  - Find nearest neighbor in eigenspace [Murase & Nayar, IJCV’95]

Applications: Recognition, Pose Estimation

- Applications: Visual Inspection

Eigenfaces: Key Idea

- Assume that most face images lie on a low-dimensional subspace determined by the first $k$ ($k < d$) directions of maximum variance
- Use PCA to determine the vectors $\mathbf{u}_1, \ldots, \mathbf{u}_k$ that span that subspace:
  \[
  \mathbf{x} \approx \mu + w_1 \mathbf{u}_1 + w_2 \mathbf{u}_2 + \ldots + w_k \mathbf{u}_k
  \]
- Represent each face using its “face space” coordinates $(w_1, \ldots, w_k)$
- Perform nearest-neighbor recognition in “face space”

Eigenfaces Example

- Training images $\mathbf{x}_1, \ldots, \mathbf{x}_N$

Mean: $\mu$

Top eigenvectors: $\mathbf{u}_1, \ldots, \mathbf{u}_k$
Eigenfaces Example 2 (Better Alignment)

Eigenfaces Example

• Face x in “face space” coordinates:

\[ x \rightarrow \left[ u_1^T (x - \mu), \ldots, u_k^T (x - \mu) \right] = \begin{bmatrix} w_1 \end{bmatrix}, \ldots, \begin{bmatrix} w_k \end{bmatrix} \]

• Reconstruction:

\[ x = \mu + w_1 u_1 + w_2 u_2 + w_3 u_3 + w_4 u_4 + \ldots \]

Important Footnote

• Don’t really implement PCA this way!
  1. How big is Σ?
     a. n×n, where n is the number of pixels in an image!
     b. However, we only have m training examples, typically m<n, so Σ will at most have rank m!
  2. You only need the first k eigenvectors

Summary: Recognition with Eigenfaces

• Process labeled training images:
  1. Find mean µ and covariance matrix Σ
  2. Find k principal components (eigenvectors of Σ) \( u_1, \ldots, u_k \)
  3. Project each training image \( x \) onto subspace spanned by principal components:
      \( \begin{bmatrix} w_1 \end{bmatrix}, \ldots, \begin{bmatrix} w_k \end{bmatrix} \)

• Given novel image \( x \):
  1. Project onto subspace:
      \( \begin{bmatrix} w_1 \end{bmatrix}, \ldots, \begin{bmatrix} w_k \end{bmatrix} \)
  2. Optional: check reconstruction error \( x - \hat{x} \) to determine whether image is really a face
  3. Classify as closest training face in k-dimensional subspace

Singular Value Decomposition (SVD)

• Any m×n matrix A may be factored such that

\[ A = U \Sigma V^T \]

\[ [m \times n] = [m \times m][m \times n][n \times n] \]

• \( U \): m×m, orthogonal matrix
  1. Columns of \( U \) are the eigenvectors of \( AA^T \)
• \( V \): n×n, orthogonal matrix
  1. Columns are the eigenvectors of \( A^T A \)
• \( \Sigma \): m×n, diagonal with non-negative entries \( \sigma_1, \sigma_2, \ldots, \sigma_s \) with \( s = \text{min}(m,n) \) are called the singular values.
  1. Singular values are the square roots of the eigenvalues of both \( AA^T \) and \( A^T A \), Columns of \( U \) are corresponding eigenvectors!

Result of SVD algorithm: \( \sigma_1, \sigma_2, \ldots, \sigma_n \)
SVD Properties

- Matlab: \[ [u \, s \, v] = \text{svd}(A) \]
  - where \( A = u \cdot s \cdot v' \)
- \( r = \text{rank}(A) \)
- Number of non-zero singular values
- \( U, V \) give us orthonormal bases for the subspaces of \( A \):
  - first \( r \) columns of \( U \): column space of \( A \)
  - last \( n-r \) columns of \( U \): left nullspace of \( A \)
  - first \( r \) columns of \( V \): row space of \( A \)
  - last \( m-r \) columns of \( V \): nullspace of \( A \)
- For \( d \leq r \), the first \( d \) columns of \( U \) provide the best \( d \)-dimensional basis for columns of \( A \) in least-squares sense

Performing PCA with SVD

- Singular values of \( A \) are the square roots of eigenvalues of both \( A A^T \) and \( A^T A \)
- Columns of \( U \) are the corresponding eigenvectors.
- And \( \sum_{i=1}^{k} a_i a_i^T = [a_1 \ldots a_k] [a_1^T \ldots a_k^T] = AA^T \)
- Covariance matrix \( \Sigma = \frac{1}{n} \sum_{i=1}^{n} (x_i - \mu)(x_i - \mu)^T \)
- So, ignoring the factor \( 1/n \), subtract mean image \( \mu \) from each input image, create data matrix, and perform (thin) SVD on the data matrix.

Thin SVD

- Any \( m \times n \) matrix \( A \) may be factored such that \( A = U \Sigma V^T \)
- If \( m > n \), then one can view \( \Sigma \) as:
  \[
  \Sigma = \begin{bmatrix}
  \sigma_1 & 0 & \cdots & 0 \\
  0 & \sigma_2 & \cdots & 0 \\
  \vdots & \vdots & \ddots & \vdots \\
  0 & 0 & \cdots & \sigma_r \\
  \end{bmatrix}
  \]
- Where \( \Sigma = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_r) \) with \( s = \min(m, n) \), and lower matrix is \((n-m) \times m\) of zeros.
- Alternatively, you can write:
  \[
  A = U \Sigma V^T
  \]
- In Matlab, thin SVD is:\[ [U \, S \, V] = \text{svds}(A, k) \] This is what you should use!

Limitations

- PCA assumes that the data has a Gaussian distribution (mean \( \mu \), covariance matrix \( \Sigma \))
- The shape of this dataset is not well described by its principal components
- Global appearance method: not robust to misalignment, background variation
- Easy fix (with considerable manual overhead)
  - Need to align the training examples
- The direction of maximum variance is not always good for classification
Topics of This Lecture

- Subspace Methods for Recognition
  - Motivation
- Principal Component Analysis (PCA)
  - Derivation
  - Eigenimages/Eigenfaces
  - Limitations
- Fisher’s Linear Discriminant Analysis (FLD/LDA)
  - Derivation
  - Fisherfaces for recognition

Restrictions of PCA

- PCA minimizes projection error
- PCA is "unsupervised" no information on classes is used
- Discriminating information might be lost

Fisher’s Linear Discriminant Analysis (FLD)

- FLD is an enhancement to PCA
  - Constructs a discriminant subspace that minimizes the scatter between images of the same class and maximizes the scatter between different class images
  - Also sometimes called LDA...

Mean Images

- Let $X_1, X_2, \ldots, X_c$ be the classes in the database and let each class $X_i$, $i = 1, 2, \ldots, c$ have $k$ images $x_{ij}$, $j = 1, 2, \ldots, k$.
- We compute the mean image $\mu_i$ of each class $X_i$ as:
$$\mu_i = \frac{1}{k} \sum_{j=1}^{k} x_{ij}$$
- Now, the mean image $\mu$ of all the classes in the database can be calculated as:
$$\mu = \frac{1}{c} \sum_{i=1}^{c} \mu_i$$

Scatter Matrices

- We calculate the within-class scatter matrix as:
$$S_W = \sum_{i=1}^{c} \sum_{j=1}^{k} (x_{ij} - \mu_i)(x_{ij} - \mu_i)^T$$
- We calculate the between-class scatter matrix as:
$$S_B = \sum_{i=1}^{c} N_i (\mu_i - \mu)(\mu_i - \mu)^T$$

Visualization

- Good separation
Fisher’s Linear Discriminant Analysis (FLD)

- Maximize distance between classes
- Minimize distance within a class
- Criterion: $J(w) = \frac{w^T S_b w}{w^T S_w w}$
- $S_b$ — between-class scatter matrix
- $S_w$ — within-class scatter matrix
- Vector $w$ is a solution of a generalized eigenvalue problem: $S_b w = \lambda S_w w$
- Classification function: $g(x) = w^T x + w_0 \geq 0$

Fisherfaces: Experiments

- Variation in lighting

Face Recognition Difficulty: Lighting

- The same person with the same facial expression, and seen from the same viewpoint, can appear dramatically different when light sources illuminate the face from different directions.

Fisherfaces: Experiments

- Variation in lighting

FLD Computation

- Maximization of $J(w) = \frac{w^T S_b w}{w^T S_w w}$
- Is given by solution of generalized eigenvalue problem $S_b w = \lambda S_w w$
- Defining $v = S_w^{-1} S_b w$ we get $S_w v = \lambda v$
- For the $c$-class case, we obtain at most $c-1$ projections.

Application: Fisherfaces

- Idea:
  - Using Fisher’s linear discriminant to find class-specific linear projections that compensate for lighting/facial expression.

Face Recognition Difficulty: Lighting

- Singularity problem:
  - The within-class scatter is always singular for face recognition, since #training images << #pixels
  - This problem is overcome by applying PCA first

Fisherfaces: Experiments

- Variation in lighting

Application: Fisherfaces

- Idea:
  - Using Fisher’s linear discriminant to find class-specific linear projections that compensate for lighting/facial expression.

Face Recognition Difficulty: Lighting

- Singularity problem:
  - The within-class scatter is always singular for face recognition, since #training images << #pixels
  - This problem is overcome by applying PCA first

Fisherfaces: Experiments

- Variation in lighting
Fisherfaces: Experimental Results

- Variation in facial expression, eye wear, lighting

Example Application: Fisherfaces

- Visual discrimination task
  - Training data:
    - \( C_1 \): Subjects with glasses
    - \( C_2 \): Subjects without glasses
  - Test:
    - glasses?

References and Further Reading

- Background information on PCA/FLD can be found in
- Important Papers (available on webpage)
  - M. Turk, A. Pentland
  - P.N. Belhumeur, J.P. Hespanha, D.J. Kriegman