Recognition (Part 2)

Computer Vision I
CSE 252A
Lecture 19

What can we say about the distribution of features in the feature space?

- Illumination variation, no shadowing 3-D linear subspace
- Illumination variation, shadowing – illumination cone.
- Approximated by a 9-D subspace (Basri, Jacobs)
- Appearance Manifold

3-D Linear subspace

The set of images of a Lambertian surface with no shadowing is a subset of 3-D linear subspace.

\[ L = \{ x \mid x = Bs, \forall s \in \mathbb{R}^3 \} \]

where \( B \) is a \( n \) by 3 matrix.

The Illumination Cone

Theorem: The set of images of any object in fixed posed, but under all lighting conditions, is a convex cone in the image space.

(Belhumeur and Kriegman, IJCV, 98)
**Appearance manifold approach**

- For every object
  1. Sample the set of viewing conditions
  2. Crop & scale images to standard size
  3. Use as feature vector
- Apply principal component analysis over all the images
- Keep the dominant principal components
- Set of views for one object is represented as a manifold in the projected space
- Recognition: What is nearest manifold for a given test image?

(Nayar et al. '96)

---

**Linear Projection to Lower Dimension**

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

  where $W$ is an $n$ by $m$ matrix.
- Recognition is performed using nearest neighbor in $\mathbb{R}^m$.
- How do we choose a good $W$?

---

**Singular value decomposition and its relationship to eigen decomposition**

- Any $m$ by $n$ matrix $A$ may be factored such that
  
  \[ A = U \Sigma V^T \]

  where

  - $U$: $m$ by $m$, orthogonal matrix
    - Columns of $U$ are the eigenvectors of $A^T A$
  - $V$: $n$ by $n$, orthogonal matrix,
    - Columns are the eigenvectors of $A A^T$
  - $\Sigma$: $m$ by $n$, diagonal with non-negative entries ($\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_s$) with $s = \min(m,n)$ are called the called the singular values
    - Singular values are the square roots of eigenvalues of both $A A^T$ and $A^T A$
  - Result of SVD algorithm: $\sigma_1 \geq \sigma_2 \geq \ldots \geq \sigma_s$

---

**Principal Component Analysis (PCA)**

Assume we have a set of $n$ feature vectors $x_i$ ($i = 1, \ldots, n$) in $\mathbb{R}^m$. Write

\[ \mu = \frac{1}{n} \sum x_i \]

\[ \Sigma = \frac{1}{n} \sum (x_i - \mu)(x_i - \mu)^T \]

The unit eigenvectors of $\Sigma$ — which we write as $v_1, v_2, \ldots, v_s$ where the order is given by the size of the eigenvalues and $v_1$ has the largest eigenvalue — give a set of features with the following properties:

- They are independent.
- Projection onto the basis $(v_1, \ldots, v_s)$ gives the $k$-dimensional set of linear features that preserves the most variance.

**SVD Properties**

- In Matlab, $[U \; S \; V] = \text{svd}(A)$, and you can verify that: $A = U S V^T$
- $r = \text{Rank}(A) = \#$ of non-zero singular values.
- $U$, $V$ give us orthonormal bases for the subspaces of $A$:
  - $1\text{st}$ $r$ rows of $U$: Column space of $A$
  - Last $m-r$ rows of $U^T$: Left nullspace of $A$
  - $1\text{st}$ $r$ columns of $V$: Row space of $A$
  - Last $n-r$ columns of $V^T$: (Right) nullspace of $A$
- For $d \leq r$, the first $r$ columns of $V$ provide the best $d$-dimensional basis for rows of $A$ in least squares sense.
### Economy SVD

- Any \( m \times n \) matrix \( A \) may be factored such that:
  \[
  A = U \Sigma V^T
  \]
  
- If \( n \geq m \), then one can view \( \Sigma \) as:
  \[
  [\Sigma' \ 0]
  \]
  
- Where \( \Sigma' = \text{diag}(\sigma_1, \sigma_2, \ldots, \sigma_k) \) with \( k = \min(m,n) \), and right part of matrix is \( (m-n \times n) \) of zeros.
  
- Alternatively, you can write:
  \[
  A = U \Sigma' V'^T
  \]
  
- In MATLAB, economy SVD is:
  
### Performing PCA with SVD

- Singular values of \( A \) are the square roots of eigenvalues of both \( AA^T \) and \( A^TA \).

- And:
  \[
  \sum a_i a_i^T = a_1 a_1^T + a_2 a_2^T + \cdots + a_n a_n^T = AA^T
  \]
  
- Covariance matrix is:
  \[
  \Sigma = \frac{1}{n-1} \sum_{i=1}^{n} (\mathbf{x}_i - \mu)(\mathbf{x}_i - \mu)^T
  \]
  
- So, subtract mean image \( \mu \) from each input image, create mean-deviation form of the data matrix (each image is a row), and perform economy SVD on the mean-deviation form of the data matrix.

### Eigenfaces

#### Modeling
1. Given a collection of \( n \) labeled training images,
2. Compute mean image and covariance matrix.
3. Compute \( k \) Eigenvectors (note that these are images) of covariance matrix corresponding to \( k \) largest Eigenvalues. (Or perform using SVD.)
4. Project the training images to the \( k \)-dimensional Eigenspace.

#### Recognition
1. Given a test image, project to Eigenspace.
2. Perform classification to the projected training images.

### Eigenfaces: Training Images

[ Turk, Pentland 01](image)

### Eigenfaces

- Mean Image
- Basis Images
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 \( y^T = (x - \mu)^T V' \).
- From \( y \in \mathbb{R}^m \), the reconstruction of the point is \( (x - \mu)^T = y^T V'^T \).
- The error of the reconstruction, or the distance from \( x \) to the subspace spanned by \( V' \) is:
  \[
  || (x - \mu)^T - (x - \mu)^T V' V'^T ||
  \]

Reconstruction using Eigenfaces

- Given image on left, project to Eigenspace, then reconstruct an image (right).

Face detection using “distance to face space”

- Scan a window \( \omega \) across the image, and classify the window as face/not face as follows:
  - Project window to subspace, and reconstruct as described earlier.
  - Compute distance between \( \omega \) and reconstruction.
  - Local minima of distance over all image locations less than some threshold are taken as locations of faces.
  - Repeat at different scales.
  - Possibly normalize windows intensity so that \( ||\omega|| = 1 \).

Underlying assumptions

- Background is not cluttered (or else only looking at interior of object)
- Lighting in test image is similar to that in training image.
- No occlusion
- Size of training image (window) same as window in test image.

Difficulties with PCA

- Projection may suppress important detail
  - smallest variance directions may not be unimportant
- Method does not take discriminative task into account
  - typically, we wish to compute features that allow good discrimination
  - not the same as largest variance or minimizing reconstruction error.

Limitations of these approaches

- Object must be segmented from background
  (How would one do this in non-trivial situations?)
- Occlusion?
- The variability (dimension) in images is large, so is sampling feasible?
- How can one generalize to classes of objects?
Appearance-Based Vision: Lessons

Strengths
• Posing the recognition metric in the image space rather than a derived representation is more powerful than expected.
• Modeling objects from many images is not unreasonable given hardware developments.
• The data (images) may provide a better representations than abstractions for many tasks.

Appearance-Based Vision: Lessons

Weaknesses
• Segmentation or object detection is still an issue.
• To train the method, objects have to be observed under a wide range of conditions (e.g. pose, lighting, shape deformation).
• Limited power to extrapolate or generalize (abstract) to novel conditions.

Model-Based Vision

• Given 3-D models of each object
• Detect image features (often edges, line segments, conic sections)
• Establish correspondence between model &image features
• Estimate pose
• Consistency of projected model with image.

A Rough Recognition Spectrum

Appearance-Based Recognition (Eigenface, Fisherface)

Geometric Invariants

Image Abstractions/ Volumetric Primitives

Local Features Spatial Relations

Aspect Graphs

3-D Model-Based Recognition

Function

Recognition by Hypothesize and Test

• General idea
  – Hypthesize object identity and pose
  – Recover camera parameters (widely known as backprojection)
  – Render object using camera parameters
  – Compare to image
• Issues
  – where do the hypotheses come from?
  – How do we compare to image (verification)?

• Simplest approach
  – Construct a correspondence for all object features to every correctly sized subset of image points
  – These are the hypotheses
  – Expensive search, which is also redundant.

Pose consistency

• Correspondences between image features and model features are not independent.
• A small number of correspondences yields a camera matrix --- the others correspondences must be consistent with this.

• Strategy:
  – Generate hypotheses using small numbers of correspondences (e.g. triples of points for a calibrated perspective camera, etc., etc.)
  – Backproject and verify
Voting on Pose

- Each model leads to many correct sets of correspondences, each of which has the same pose
  - Vote on pose, in an accumulator array
Invariance

- Properties or measures that are independent of some group of transformation (e.g., rigid, affine, projective, etc.)
- For example, under affine transformations:
  - Collinearity
  - Parallelism
  - Intersection
  - Distance ratio along a line
  - Angle ratios of tree intersecting lines
  - Affine coordinates

Invariance - 1

- There are geometric properties that are invariant to camera transformations
- Easiest case: view a plane object in scaled orthography.
- Assume we have three base points \( P_i \) (\( i = 1..3 \)) on the object
  - then any other point on the object can be written as
  \[
  P_k = P_1 + \mu_a (P_2 - P_1) + \mu_b (P_3 - P_1)
  \]
  - Now image points are obtained by multiplying by a plane affine transformation, so
  \[
  P'_k = A P_k = A (P_1 + \mu_a (P_2 - P_1) + \mu_b (P_3 - P_1))
  \]

Geometric hashing

- Vote on identity and correspondence using invariants
  - Take hypotheses with large enough votes
- Building a table:
  - Take all triplets of points in on model image to be base points \( P_1, P_2, P_3 \)
  - Take ever fourth point and compute \( \mu \)'s
  - Fill up a table, indexed by \( \mu \)'s, with
    - the base points and fourth point that yield those \( \mu \)'s
    - the object identity

Algorithm 18.3: Geometric hashing: voting on identity and point labels

For all groups of three image points \( T(J) \)
  For every other image point \( p \)
    Compute the \( \mu \)'s from \( p \) and \( T(J) \)
    Obtain the table entry at these values
    if there is one, it will label the three points in \( T(J) \)
    with the name of the object and the names of those particular points.
  Cluster those labels.
  if there are enough labels, backproject and verify end
end
Verification

- Edge score
  - are there image edges near predicted object edges?
  - very unreliable; in texture, answer is usually yes

- Oriented edge score
  - are there image edges near predicted object edges with the right orientation?
  - better, but still hard to do well

- Texture
  - e.g. does the spanner have the same texture as the wood?

Matching using Local Image features

Simple approach

- Detect corners in image (e.g. Harris corner detector).
- Represent neighborhood of corner by a feature vector produced by Gabor Filters, K-jets, affine-invariant features, etc.).
- Modeling: Given an training image of an object w/o clutter, detect corners, compute feature descriptors, store these.
- Recognition time: Given test image with possible clutter, detect corners and compute features. Find models with same feature descriptors (hashing) and vote.

Probabilistic interpretation

- Write
  \[ P[\text{patch of type } i \text{ appears in image | pattern is present}] = p_{ij} \]
  \[ P[\text{patch of type } i \text{ appears in image | no pattern is present}] = p_{ii} \]

- Assume
  \[ p_{ij} = \mu \text{ if the pattern can produce this patch and } 0 \text{ otherwise} \]
  \[ p_{ii} = \lambda < \mu \text{ for all } i \]

- Likelihood of image given pattern

  \[ P[\text{interpretation | pattern}] = \lambda^n \mu^{100-n} \]
Employ spatial relations

![Diagram showing spatial relations](image1)

Figure from "Local grayvalue invariants for image retrieval," by C. Schmid and R. Mohr, IEEE Trans. Pattern Analysis and Machine Intelligence, 1997 copyright 1997, IEEE

Example

![Example image](image2)

Finding faces using relations

- Strategy: compare
  - Face is eyes, nose, mouth, etc. with appropriate relations between them
  - Build a specialised detector for each of these (template matching) and look for groups with the right internal structure
  - Once we’ve found enough of a face, there is little uncertainty about where the other bits could be

Finding faces using relations

- Strategy: compare
  - Face is eyes, nose, mouth, etc. with appropriate relations between them
  - Build a specialised detector for each of these (template matching) and look for groups with the right internal structure
  - Once we’ve found enough of a face, there is little uncertainty about where the other bits could be

![Graph showing finding faces](image3)

Notice that once some facial features have been found, the position of the rest is quite strongly constrained.

Figure from, "Finding faces in cluttered scenes using random labelled graph matching," by Leung, T.; Burl, M and Perona, P. Proc. Int. Conf. on Computer Vision, 1995 copyright 1995, IEEE