Object Recognition: The Problem

Given: A database \( D \) of "known" objects and an image \( I \):

1. Determine which (if any) objects in \( D \) appear in \( I \)
2. Determine the pose (rotation and translation) of the object

Object recognition

- **Feature Extraction**: Image (window)
- **Classification**: Feature Vector
- **Object Identity**: Pose Est. (where is it 3D), Segmentation (where is it 2D), Recognition (what is it)

Feature:
- Image as a feature vector
- PCA (Eigenfaces)
- Linear Subspace (3-D) per class
- Fisherfaces

Classifier:
- Nearest Neighbor
- Closest Subspace
- Bayesian classifier
- Support Vector Machine

Bayesian Classification

An Example

- "Sorting incoming Fish on a conveyor according to species using optical sensing"

Species
- Sea bass
- Salmon

Pattern Classification
• Adopt the lightness and add the width of the fish

Fish

\[ x^T = [x_1, x_2] \]

Lightness

Width

• However, our satisfaction is premature because the central aim of designing a classifier is to correctly classify novel input

Issue of generalization!
Introduction

• The sea bass/salmon example

  • State of nature, prior
    • State of nature is a random variable
    • The catch of salmon and sea bass is equiprobable
      - \( P(\omega_1), P(\omega_2) \) Prior probabilities
      - \( P(\omega_1) = P(\omega_2) \) (uniform priors)
      - \( P(\omega_1) + P(\omega_2) = 1 \) (exclusivity and exhaustivity)

• Decision rule with only the prior information
  - Decide \( \omega_1 \) if \( P(\omega_1) > P(\omega_2) \) otherwise decide \( \omega_2 \)

• Use of the class–conditional information
  - \( P(x | \omega_1) \) and \( P(x | \omega_2) \) describe the difference in lightness between populations of sea-bass and salmon

• Posterior, likelihood, evidence
  - \( P(\omega_j | x) = \frac{(P(x | \omega_j) * P(\omega_j))}{P(x)} \) (BAYES RULE)
    - In words, this can be said as:
      Posterior = (Likelihood * Prior) / Evidence
    - Where in case of two categories
      \[ P(x) = \sum_{j=1}^{2} P(x | \omega_j)P(\omega_j) \]
Intuitive decision rule given the posterior probabilities:

Given $x$:
- If $P(\omega_1 | x) > P(\omega_2 | x)$, True state of nature = $\omega_1$
- If $P(\omega_1 | x) < P(\omega_2 | x)$, True state of nature = $\omega_2$

Why do this?: Whenever we observe a particular $x$, the probability of error is:
- $P(\text{error} | x) = P(\omega_1 | x)$ if we decide $\omega_2$
- $P(\text{error} | x) = P(\omega_2 | x)$ if we decide $\omega_1$

Since decision rule is optimal for each feature value $X$, there is not better rule for all $x$.

Bayesian Decision Theory – Continuous Features

Generalization of the preceding ideas:
- Use of more than one feature
- Use more than two states of nature
- Allowing actions and not only decide on the state of nature
- Introduce a loss of function (more general than the probability of error)
- Allowing actions other than classification primarily allows the possibility of rejection
- Refusing to make a decision in close or bad cases!
- Letting loss function state how costly each action taken is

Bayesian Decision Theory – Continuous Features

Finding skin:
- Skin has a very small range of (intensity independent) colours, and little texture
  - Compute an intensity-independent colour measure, check if colour is in this range, check if there is little texture (median filter)
  - See this as a classifier - we can set up the tests by hand, or learn them.
  - get class conditional densities (histograms), priors from data (counting)
- Classify:
  - if $p(\text{skin}|x) > \theta$, classify as skin
  - if $p(\text{skin}|x) < \theta$, classify as not skin
  - if $p(\text{skin}|x) = \theta$, choose classes uniformly and at random
Support Vector Machines

- Bayes classifiers and generative approaches in general try to model the posterior, $p(k|x)$
- Instead, try to obtain the decision boundary directly
  - potentially easier, because we need to encode only the geometry of the boundary, not any irrelevant wiggles in the posterior.
  - Not all points affect the decision boundary

Set $S$ of points $x_i \in \mathbb{R}^n$, each $x_i$ belongs to one of two classes $y_i \in \{-1,1\}$

The goal is to find a hyperplane that divides $S$ in these two classes

$$S$$ is separable if there exists $w, b \in \mathbb{R}^n$ such that

$$y_i (w \cdot x_i + b) \geq 1$$

$$d_i = \frac{w \cdot x_i + b}{w} \quad w = \frac{1}{\|w\|}$$

Separating hyperplanes:

$$w \cdot x + b = 0$$

Problem 1:

Minimize $\frac{1}{2} w \cdot w$

Subject to $y_i (w \cdot x_i + b) \geq 1$, $i = 1, 2, \ldots, N$

Support Vector Machines

Variability:
- Camera position
- Illumination
- Internal parameters

Within-class variations
Appearance manifold approach

- for every object (Nayar et al. '96)
  1. sample the set of viewing conditions
  2. Crop & scale images to standard size
  3. Use as feature vector
- apply a PCA over all the images
- keep the dominant PCs
- 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?

An example: input images

An example: basis images

An example: surfaces of first 3 coefficients

Parameterized Eigenspace

Recognition
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.

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.

A Rough Recognition Spectrum

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.

Motion
Structure-from-Motion (SFM)

Goal: Take as input two or more images or video w/o any information on camera position/motion, and estimate camera position and 3-D structure of scene.

Two Approaches

1. Discrete motion (wide baseline)
   1. Orthographic (affine) vs. Perspective
   2. Two view vs. Multi-view
   3. Calibrated vs. Uncalibrated
2. Continuous (Infinitesimal) motion

Discrete Motion: Some Counting

Consider \( M \) images of \( N \) points, how many unknowns?

1. Affix coordinate system to location of first camera
   location: \((M-1)\times6\) Unknowns
2. 3-D Structure: \(3N\) Unknowns
3. Can only recover structure and motion up to scale. Why?

Total number of unknowns: \((M-1)\times6+3N-1\)
Total number of measurements: \(2MN\)
Solution is possible when \((M-1)\times6+3N-1 \leq 2MN\)

Epipolar Constraint: Calibrated Case

\[ \mathbf{E} \mathbf{p} = \mathbf{0} \]

where \( \mathbf{E} = \begin{bmatrix} 0 & -t_z & t_y \\ t_z & 0 & -t_x \\ -t_y & t_x & 0 \end{bmatrix} \)

The Eight-Point Algorithm (Longuet-Higgins, 1981)

Let \( \mathbf{E} \) denote the Essential Matrix Here

\[ \begin{bmatrix} F_{11} & F_{12} & F_{13} \\ F_{21} & F_{22} & F_{23} \\ F_{31} & F_{32} & F_{33} \end{bmatrix} \begin{bmatrix} u' \\ v' \end{bmatrix} = 0 \]

\[ \begin{bmatrix} u'w \end{bmatrix}, \begin{bmatrix} u'w' \end{bmatrix}, \begin{bmatrix} u'w'' \end{bmatrix}, \begin{bmatrix} u'w''' \end{bmatrix}, \begin{bmatrix} u'w'''' \end{bmatrix} \]

Set \( F_{33} \) to 1

\[ \begin{bmatrix} F_{11} \ F_{12} \ F_{13} \\ F_{21} \ F_{22} \ F_{23} \\ 0 \ 0 \ 1 \end{bmatrix} = 1 \]

Solve For \( \mathbf{F} \)

Solve For \( \mathbf{R} \) and \( \mathbf{t} \)

Sketch of Two View SFM Algorithm

Input: Two images
1. Detect feature points
2. Find 8 matching feature points (easier said than done)
3. Compute the Essential Matrix \( \mathbf{E} \) using Normalized 8-point Algorithm
4. Compute \( \mathbf{R} \) and \( \mathbf{T} \) (recall that \( \mathbf{E} = \mathbf{RS} \) where \( \mathbf{S} \) is skew symmetric matrix)
5. Perform stereo matching using recovered epipolar geometry expressed via \( \mathbf{E} \).
6. Reconstruct 3-D geometry of corresponding points.

Feature points

Select strongest features (e.g. 1000/image)
Finding Corners

Intuition:
• Right at corner, gradient is ill defined.
• Near corner, gradient has two different values.

\[
M = \begin{bmatrix}
\frac{\partial I}{\partial x} & \frac{\partial I}{\partial y}
\end{bmatrix}
\int \int \left[ \frac{\partial I}{\partial x} \right] \left[ \frac{\partial I}{\partial y} \right] dxdy
\]

\(M\) should have large eigenvalues (e.g., Harris & Stephens '88; Shi & Tomasi '94)

Find points that differ as much as possible from all neighboring points

Feature = local maxima of \(F(\lambda_1, \lambda_2)\)

Detecting Feature points

Feature matching

Evaluate normalized cross correlation (or sum of squared differences) for all features with similar coordinates
e.g. \((x', y') = [x + \Delta x, x + \Delta y, y + \Delta y, y + \Delta y] \]

Keep mutual best matches
Still many wrong matches!

Comments

• Greedy Algorithm:
  – Given feature in one image, find best match in second image irrespective of other matches.
  – OK for small motions, little rotation, small search window
• Otherwise
  – Must compare descriptor over rotation
  – Can’t consider \(O(n^2)\) potential pairings (way too many), so
    • Manual correspondence (e.g., façade, photogrametry).
    • Use random sampling (RANSAC).
    • More descriptive features (line segments, SIFT, larger regions, color).
    • Use video sequence to track, but perform SFM w/ first and last image.

Continuous Motion

• Consider a video camera moving continuously along a trajectory (rotating & translating).
• How do points in the image move
• What does that tell us about the 3-D motion & scene structure?