Introduction

Computer vision is the analysis of digital images by a computer for such applications as:

- **Industrial**: part localization and inspection, robotics
- **Medical**: disease classification, screening, planning
- **Military**: autonomous vehicles, tank recognition
- **Intelligence Gathering**: face recognition, video analysis
- **Security**: video analysis
- **Science**: classification, measurement
- **Document Processing**: text recognition, diagram conversion
Medical Applications

CT image of a patient’s abdomen

liver
kidney
kidney
Visible Man Slice Through Lung
3D Reconstruction of the Blood Vessel Tree
CBIR of Mouse Eye Images for Genetic Studies
Robotics

• 2D Gray-tone or Color Images

“Mars” rover

• 3D Range Images

What am I?
Image Databases:

Images from my Ground-Truth collection.

- Retrieve all images that have trees.
- Retrieve all images that have buildings.
- Retrieve all images that have antelope.
Surveillance: Object and Event Recognition in Aerial Videos

Original Video Frame

Color Regions

Structure Regions
Digital Image Terminology:

- pixel (with value 94)
- its 3x3 neighborhood
- region of medium intensity
- resolution (7x7)

- binary image – 0’s and 1’s
- gray-scale (or gray-tone) image – 0 to 255
- color image – (R,G,B) at each pixel
- multi-spectral image – multiple values per pixel
- range image – depth value at each pixel
- labeled image – result of processing and labeling
Goals of Image and Video Analysis

- Segment an image into useful regions
- Perform measurements on certain areas
- Determine what object(s) are in the scene
- Calculate the precise location(s) of objects
- Visually inspect a manufactured object
- Construct a 3D model of the imaged object
- Find “interesting” events in a video
The Three Stages of Computer Vision

• low-level
  
  image → image

• mid-level
  
  image → features

• high-level (the intelligent part)
  
  features → analysis
Low-Level sharpening

blurring
Low-Level

Mid-Level (Lines and Curves)

Canny edge operator

original image

edge image

data structure

ORT line & circle extraction

circular arcs and line segments
Mid-level (Regions)

original color image

K-means clustering (followed by connected component analysis)

data structure

regions of homogeneous color
Low- to High-Level

Building Recognition
Filtering Operations Use Masks

- Masks operate on a neighborhood of pixels.
- A mask of coefficients is centered on a pixel.
- The mask coefficients are multiplied by the pixel values in its neighborhood and the products are summed.
- The result (response) goes into the corresponding pixel position in the output image.

\[
\begin{array}{cccc}
36 & 36 & 36 & 36 \\
36 & 36 & 45 & 45 \\
36 & 45 & 45 & 54 \\
36 & 45 & 54 & 54 \\
45 & 45 & 54 & 54 \\
\end{array}
\quad\quad
\begin{array}{ccc}
1/9 & 1/9 & 1/9 \\
1/9 & 1/9 & 1/9 \\
1/9 & 1/9 & 1/9 \\
\end{array}
\quad\quad
\begin{array}{cccc}
** & ** & ** & ** \\
** & ** & ** & ** \\
** & ** & 39 & ** \\
** & ** & ** & ** \\
** & ** & ** & ** \\
\end{array}
\]

Input Image

3x3 Mask (mean filter)

Output Image
Comparison: salt and pepper noise

Mean  Gaussian  Median

3x3  

5x5  

7x7  
Comparison: Gaussian noise

Mean

Gaussian

Median

3x3

5x5

7x7
Lines and Arcs Segmentation

In some image sets, lines, curves, and circular arcs are more useful than regions or helpful in addition to regions.

Lines and arcs are often used in

- object recognition
- stereo matching
- document analysis
Edge Detection

Basic idea: look for a neighborhood with strong signs of change.

Problems:

• neighborhood size

• how to detect change
Differential Operators

Differential operators

• attempt to approximate the gradient at a pixel via masks

• threshold the gradient to select the edge pixels
Example: Sobel Operator

\[ \text{Sx} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}, \quad \text{Sy} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \]

On a pixel of the image \( I \)

- let \( g_x \) be the response to \( S_x \)
- let \( g_y \) be the response to \( S_y \)

Then the gradient is

\[ \nabla I = [g_x \ g_y]^T \]

and \( g = (g_x^2 + g_y^2)^{1/2} \) is the gradient magnitude.

\[ \theta = \text{atan2}(g_y, g_x) \] is the gradient direction.
Sobel Operator on the Blocks Image

original image               gradient                  thresholded
magnitude                 magnitude
Common Masks for Computing Gradient

- **Sobel:**
  
  $\begin{bmatrix}
  -1 & 0 & 1 \\
  -2 & 0 & 2 \\
  -1 & 0 & 1 \\
  \end{bmatrix}$  
  \hspace{1cm} $\begin{bmatrix}
  1 & 2 & 1 \\
  0 & 0 & 0 \\
  -1 & -2 & -1 \\
  \end{bmatrix}$

- **Prewitt:**
  
  $\begin{bmatrix}
  -1 & 0 & 1 \\
  -1 & 0 & 1 \\
  -1 & 0 & 1 \\
  \end{bmatrix}$  
  \hspace{1cm} $\begin{bmatrix}
  1 & 1 & 1 \\
  0 & 0 & 0 \\
  -1 & -1 & -1 \\
  \end{bmatrix}$

- **Roberts**
  
  $\begin{bmatrix}
  0 & 1 \\
  -1 & 0 \\
  \end{bmatrix}$  
  \hspace{1cm} $\begin{bmatrix}
  1 & 0 \\
  0 & -1 \\
  \end{bmatrix}$

\[S_x \quad S_y\]
Canny Edge Detector

- **Smooth the image** with a Gaussian filter with spread \( \sigma \).

- Compute gradient **magnitude and direction** at each pixel of the smoothed image.

- **Zero out** any pixel response \( \leq \) the two neighboring pixels on either side of it, along the direction of the gradient.

- **Track high-magnitude contours.**

- **Keep only pixels along these contours**, so weak little segments go away.
Canny Examples

Canny $\sigma=1$  
Canny $\sigma=4$

Canny $\sigma=1$  
Roberts 2X2
Canny on Kidney Image
Canny on the Blocks image
Canny Characteristics

• The Canny operator gives single-pixel-wide images with good continuation between adjacent pixels

• It is the most widely used edge operator today; no one has done better since it came out in the late 80s. Many implementations are available.

• It is very sensitive to its parameters, which need to be adjusted for different application domains.
Segmentation into Regions

• Instead of looking for 1D features like lines and curves, some processes look for regions.

• The regions must be homogeneous in some attribute such as gray-tone, color, texture,...

• Although “region-growing” was popular in the past, clustering the pixels into subsets has become the best methodology for finding regions.
K-Means Example 1

1. Select an image: imgs/P1010012.jpg
2. Select a processor: KMCluster
3. Click process>>

Options:
Init Method: 0

Process done!
K-Means Example 2
K-Means Example 3

1. Select an image: imgs/Pa170028.jpg
2. Select a processor: KMCluster
3. Click process>>

Options:
Init Method

Process done!

640*480 (607,118): RGB(20,22,1)

(228,26): RGB(255,170,0)
High-Level Computer Vision

- Detection of classes of objects (faces, motorbikes, trees, cheetahs) in images
- Recognition of specific objects such as George Bush or machine part #45732
- Classification of images or parts of images for medical or scientific applications
- Recognition of events in surveillance videos
- Measurement of distances for robotics
High-level vision uses techniques from AI

- **Graph-Matching**: A*, Constraint Satisfaction, Branch and Bound Search, Simulated Annealing

- **Learning Methodologies**: Decision Trees, Neural Nets, SVMs, EM Classifier

- **Probabilistic Reasoning**: Belief Propagation, Graphical Models
Graph Matching for Object Recognition

• For each specific object, we have a geometric model.

• The geometric model leads to a symbolic model in terms of image features and their spatial relationships.

• An image is represented by all of its features and their spatial relationships.

• This leads to a graph matching problem.
Model-based Recognition as Graph Matching

- Let $U$ = the set of model features.
- Let $R$ be a relation expressing their spatial relationships.
- Let $L$ = the set of image features.
- Let $S$ be a relation expressing their spatial relationships.
- The ideal solution would be a subgraph isomorphism $f: U \rightarrow L$ satisfying
  - if $(u_1, u_2, ..., u_n) \in R$, then $(f(u_1), f(u_2), ..., f(u_n)) \in S$
House Example

2D model                  2D image

\[ P = \{ S_1, S_2, S_3, S_4, S_5, S_6, S_7, S_8, S_9, S_{10}, S_{11} \}. \]
\[ L = \{ S_a, S_b, S_c, S_d, S_e, S_f, S_g, S_h, S_i, S_j, S_k, S_l, S_m \}. \]

\[ R_P = \{ (S_1, S_2), (S_1, S_5), (S_1, S_6), (S_2, S_3), (S_2, S_4), (S_3, S_4), (S_3, S_9), (S_4, S_5), (S_4, S_7), (S_4, S_{11}), (S_5, S_6), (S_5, S_7), (S_5, S_{11}), (S_6, S_8), (S_6, S_{11}), (S_7, S_9), (S_7, S_{10}), (S_7, S_{11}), (S_8, S_{10}), (S_8, S_{11}), (S_9, S_{10}) \}. \]

\[ R_L = \{ (S_a, S_b), (S_a, S_j), (S_a, S_n), (S_b, S_c), (S_b, S_d), (S_b, S_n), (S_c, S_d), (S_d, S_e), (S_d, S_f), (S_d, S_g), (S_e, S_f), (S_e, S_g), (S_f, S_l), (S_f, S_m), (S_g, S_h), (S_g, S_i), (S_g, S_n), (S_h, S_i), (S_h, S_k), (S_h, S_l), (S_h, S_n), (S_i, S_j), (S_i, S_k), (S_i, S_n), (S_j, S_k), (S_k, S_l), (S_l, S_m) \}. \]

\[ f(S_1) = S_j \quad f(S_4) = S_n \quad f(S_7) = S_g \quad f(S_{10}) = S_f \]
\[ f(S_2) = S_a \quad f(S_5) = S_i \quad f(S_8) = S_l \quad f(S_{11}) = S_h \]
\[ f(S_3) = S_b \quad f(S_6) = S_k \quad f(S_9) = S_d \]

RP and RL are connection relations.
But this is too simplistic

• The model specifies all the features of the object that may appear in the image.

• Some of them don’t appear at all, due to occlusion or failures at low or mid level.

• Some of them are broken and not recognized.

• Some of them are distorted.

• Relationships don’t all hold.
TRIBORS: view class matching of polyhedral objects

- A *view-class* is a typical 2D view of a 3D object.

- Each object had 4-5 view classes (hand selected).

- The representation of a view class for matching included:
  - triplets of line segments visible in that class
  - the probability of detectability of each triplet

The first version of this program used *iterative-deepening A* search.
RIO: Relational Indexing for Object Recognition

- RIO worked with more complex parts that could have
  - planar surfaces
  - cylindrical surfaces
  - threads
Object Representation in RIO

- 3D objects are represented by a 3D mesh and set of 2D view classes.

- Each view class is represented by an attributed graph whose nodes are features and whose attributed edges are relationships.

- For purposes of indexing, attributed graphs are stored as sets of 2-graphs, graphs with 2 nodes and 2 relationships.
RIO Features

- Ellipses
- Coaxials
- Coaxials-multi
- Parallel lines (close and far)
- Junctions (L, V, Y, Z, U)
- Triples
RIO Relationships

- share one arc
- share one line
- share two lines
- coaxial
- close at extremal points
- bounding box encloses / enclosed by
Hexnut Object

How are 1, 2, and 3 related?

What other features and relationships can you find?

RELATIONS:
a: encloses
b: coaxial

FEATURES:
1: coaxials-multi
2: ellipse
3: parallel lines
Graph and 2-Graph Representations

- 1 coaxial-multi
  - encloses
  - encloses
  - encloses
  - coaxial

- 2 ellipse

- 3 parallel lines

RDF!
Preprocessing (off-line) Phase

for each model view Mi in the database

• **encode** each 2-graph of Mi to produce an index

• store Mi and associated information in the indexed bin of a hash table H
Matching (on-line) phase

1. Construct a relational (2-graph) description $D$ for the scene

2. For each 2-graph $G$ of $D$
   - encode it, producing an index to access the hash table $H$
   - cast a vote for each $M_i$ in the associated bin

3. Select the $M_i$’s with high votes as possible hypotheses

4. Verify or disprove via alignment, using the 3D meshes
The Voting Process

The diagram illustrates the voting process with a hash function applied to a shared arc. The hash function produces aaku (1,2,9,9) which is then used to retrieve a list of models: $M_1, M_5, M_{23}, M_{81}$. These models are then voted for by accumulators, with each model receiving a vote of +1.
1. The matched features of the hypothesized object are used to determine its pose.

2. The 3D mesh of the object is used to project all its features onto the image.

3. A verification procedure checks how well the object features line up with edges on the image.
Use of classifiers is big in computer vision today.

• 2 Examples:
  – Rowley’s Face Detection using neural nets
  – Yi’s image classification using EM
Object Detection: Rowley’s Face Finder

1. convert to gray scale
2. normalize for lighting
3. histogram equalization
4. apply neural net(s) trained on 16K images

What data is fed to the classifier?

32 x 32 windows in a pyramid structure
Object Class Recognition using Images of Abstract Regions

Yi Li, Jeff A. Bilmes, and Linda G. Shapiro
Department of Computer Science and Engineering
Department of Electrical Engineering
University of Washington
Problem Statement

**Given:** Some images and their corresponding descriptions

- {trees, grass, cherry trees}
- {cheetah, trunk}
- {mountains, sky}
- {beach, sky, trees, water}

**To solve:** What object classes are present in new images
Image Features for Object Recognition

- Color
- Texture
- Structure
- Context
Abstract Regions

Multiple segmentations whose regions are not labeled; a list of labels is provided for each training image.

image

labels

{sky, building}

various different segmentations

region attributes from several different types of regions
Model Initial Estimation

• Estimate the initial model of an object using all the region features from all images that contain the object
EM Classifier: the Idea

Initial Model for “trees”

[Diagram showing initial and final models for “trees”]

Final Model for “trees”

Initial Model for “sky”

[Diagram showing initial and final models for “sky”]

Final Model for “sky”
A Better Approach to Combining Different Feature Types

Phase 1:

- Treat each type of abstract region separately
- For abstract region type $a$ and for object class $o$, use the EM algorithm to construct clusters that are multivariate Gaussians over the features for type $a$ regions.
### Aggregate Scores for Color

<table>
<thead>
<tr>
<th>Components</th>
<th>1</th>
<th>2</th>
<th>3</th>
<th>4</th>
<th>5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td>beach</td>
<td>.93</td>
<td>.16</td>
<td>.94</td>
<td>.24</td>
<td>.10</td>
<td>.99</td>
<td>.32</td>
<td>.00</td>
</tr>
<tr>
<td>beach</td>
<td>.66</td>
<td>.80</td>
<td>.00</td>
<td>.72</td>
<td>.19</td>
<td>.01</td>
<td>.22</td>
<td>.02</td>
</tr>
<tr>
<td>not beach</td>
<td>.43</td>
<td>.03</td>
<td>.00</td>
<td>.00</td>
<td>.00</td>
<td>.00</td>
<td>.15</td>
<td>.00</td>
</tr>
</tbody>
</table>
Phase 2 Learning

• Let $C_i$ be row $i$ of the training matrix.

• Each such row is a feature vector for the color features of regions of image $I_i$ that relates them to the Phase 1 components.

• Now we can use a second-stage classifier to learn $P(o|I_i)$ for each object class $o$ and image $I_i$. 
Multiple Feature Case

• We calculate separate Gaussian mixture models for each different features type:

  • Color: $C_i$
  • Texture: $T_i$
  • Structure: $S_i$

• and any more features we have (motion).
Now we concatenate the matrix rows from the different region types to obtain a \textbf{multi-feature-type training matrix} and train a neural net classifier to classify images.

\begin{align*}
\begin{array}{c}
\text{color} \\
C_1^+ \\
C_2^+ \\
\cdot \\
\cdot \\
C_1^- \\
C_2^- \\
\cdot \\
\cdot \\
\end{array} & \begin{array}{c}
texture \\
T_1^+ \\
T_2^+ \\
\cdot \\
\cdot \\
T_1^- \\
T_2^- \\
\cdot \\
\cdot \\
\end{array} & \begin{array}{c}
structure \\
S_1^+ \\
S_2^+ \\
\cdot \\
\cdot \\
S_1^- \\
S_2^- \\
\cdot \\
\cdot \\
\end{array} & \begin{array}{c}
\text{everything} \\
C_1^+ \\
C_2^+ \\
\cdot \\
\cdot \\
C_1^- \\
C_2^- \\
\cdot \\
\cdot \\
\end{array} \\
\end{align*}
Groundtruth Data Set: Top Results

Asian city

Cannon beach

Italy

park
Groundtruth Data Set: Top Results

- sky
- spring flowers
- tree
- water
Groundtruth Data Set: Annotation Samples

tree(97.3), bush(91.6),
spring flowers(90.3),
flower(84.4),
park(84.3),
sidewalk(67.5),
grass(52.5), pole(34.1)

sky(99.8),
Columbia gorge(98.8),
lantern(94.2), street(89.2),
house(85.8), bridge(80.8),
car(80.5), hill(78.3),
boat(73.1), pole(72.3),
water(64.3), mountain(63.8),
building(9.5)

sky(95.1), Iran(89.3),
house(88.6),
building(80.1),
boat(71.7), bridge(67.0),
water(13.5), tree(7.7)

Italy(99.9), grass(98.5),
Sky(93.8), rock(88.8),
boat(80.1), water(77.1),
Iran(64.2), stone(63.9),
bride(59.6), European(56.3),
sidewalk(51.1), house(5.3)