What is computer vision?

- The goal of computer vision is to write computer programs that can interpret images and understand the scene. The holy grail is to mimic human vision system.

Can computer match human perception?

- Yes and no (but mostly no!)
  - computers can be better at “easy” things
  - humans are much better at “hard” things
Components of a computer vision system:
- Camera
- Lighting
- Computer
- Scene
- Scene Interpretation

Camera trial #1
- Put a piece of film in front of an object.

Pinhole camera
- Add a barrier to block off most of the rays.
- It reduces blurring
- The pinhole is known as the aperture
- The image is inverted
Shrinking the aperture

Why not making the aperture as small as possible?
- Less light gets through
- Diffraction effect

Adding a lens

A lens focuses light onto the film
- There is a specific distance at which objects are “in focus”
- Other points project to a “circle of confusion” in the image

Lenses

Thin lens equation: \( \frac{1}{d_o} + \frac{1}{d_i} = \frac{1}{f} \)
- Any object point satisfying this equation is in focus
- Thin lens applet:
  http://www.phy.ntnu.edu.tw/java/Lens/lens_e.html

Exposure = aperture + shutter speed

- Aperture of diameter D restricts the range of rays (aperture may be on either side of the lens)
- Shutter speed is the amount of time that light is allowed to pass through the aperture
Exposure

- Two main parameters:
  - Aperture (in f stop)
  - Shutter speed (in fraction of a second)

Exposure

- Two main parameters:
  - Aperture (in f stop)
  - Shutter speed (in fraction of a second)

Reciprocity

The same exposure is obtained with an exposure twice as long and an aperture area half as big.
- Hence square root of two progression of f stops vs. power of two progression of shutter speed
- Reciprocity can fail for very long exposures

Effects of shutter speeds

- Slower shutter speed => more light, but more motion blur
- Faster shutter speed freezes motion

Depth of field

Changing the aperture size affects depth of field. A smaller aperture increases the range in which the object is approximately in focus.
Depth of field

Changing the aperture size affects depth of field. A smaller aperture increases the range in which the object is approximately in focus.

Film camera

- Aperture & shutter
- Lens & motor
- Film

Digital camera

- Aperture & shutter
- Lens & motor
- Sensor array

- A digital camera replaces film with a sensor array
- Each cell in the array is a light-sensitive diode that converts photons to electrons

From Photography, London et al.
**CCD v.s. CMOS**
- CCD is less susceptible to noise (special process, higher fill factor)
- CMOS is more flexible, less expensive (standard process), less power consumption

**SLR (Single-Lens Reflex)**
- Reflex (R in SLR) means that we see through the same lens used to take the image.
- Not the case for compact cameras

**SLR view finder**

**Color**
So far, we’ve only talked about monochrome sensors. Color imaging has been implemented in a number of ways:
- Field sequential
- Multi-chip
- Color filter array
- X3 sensor
Field sequential

Prokudin-Gorskii (early 1900’s)

Lantern projector

http://www.loc.gov/exhibits/empire/
Prokudin-Gorskii (early 1990’s)

Color filter array

Bayer’s pattern

Multi-chip

wavelength dependent

Color filter arrays (CFAs)/color filter mosaics
Demosaicking CFA’s

Demosaicking CFA’s

bilinear interpolation

\[ G_{44} = (G_{34} + G_{43} + G_{45} + G_{54}) / 4 \]
\[ R_{44} = (R_{33} + R_{35} + R_{53} + R_{55}) / 4 \]

Original input linear interpolation

Digital camera review website

• A cool video of digital camera illustration
• http://www.dpreview.com/

Now, we have images

• We can think of an image as a function, \( f: \mathbb{R}^2 \rightarrow \mathbb{R} \):
  - \( f(x, y) \) gives the intensity at position \( (x, y) \)

• What about color images?
Write a program to interpret images

Low-level vision (early vision)
- Considers local properties of an image
  "There's an edge!"

Mid-level vision
- Grouping and segmentation
  "There's an object and a background!"

Computer vision programs
High-level vision

- Recognition

“It's a chair!”

Detection

- Edges
- Lines
- Corners

Low-level vision

Image filtering

- Convolution with a mask
Image filtering (motion blur)

- Convolution with a mask

![Image](image1.png)  ![Image](image2.png)

<table>
<thead>
<tr>
<th>1</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
</tbody>
</table>

Image filtering (sharpening)

- Convolution with a mask

![Image](image3.png)  ![Image](image4.png)

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>9</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>-1</td>
<td>-1</td>
<td>-1</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Demo with PaintShop Pro

Gaussian filters

- One-dimensional Gaussian

\[ G_1(x) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} \]

- Two-dimensional Gaussian

\[ G_2(x, y) = \frac{1}{\sigma \sqrt{2\pi}} e^{-\frac{x^2 + y^2}{2\sigma^2}} \]
Gaussian filters

Computing Discrete Convolutions

\[ \text{Out}(x, y) = \sum_i \sum_j f(i, j) \cdot I_n(x - i, y - j) \]

- If \( I_n \) is \( n \times n \), \( f \) is \( m \times m \), takes time \( O(m^2 n^2) \)
- OK for small filter kernels, bad for large ones

Example: smoothing

Canny edge detector

- Smooth
- Find derivative
- Thresholding
- Thinning
Canny edge detector

• First, smooth with a Gaussian of some width $\sigma$

Original Image  
blurred Image

Canny edge detector

• Next, find “derivative”

• What is derivative in 2D? Gradient:

$$\nabla f(x,y) = \left( \frac{\partial f}{\partial x}, \frac{\partial f}{\partial y} \right)$$

Canny edge detector

Horizontal gradient  
Vertical gradient

Canny edge detector

Original Image  
Smoothed Gradient Magnitude
Canny edge detector

• thresholding

Original Image  Threshold Gradient Magnitude

Canny edge detector

• Thinning

Original Image  Edges

Canny edge detector

• Nonmaximum suppression
  - Eliminate all but local maxima in magnitude of gradient
  - At each pixel look along direction of gradient: if either neighbor is bigger, set to zero
  - In practice, quantize direction to horizontal, vertical, and two diagonals
  - Result: “thinned edge image”

Detecting lines

• What is the difference between line detection and edge detection?
  - Edges = local
  - Lines = nonlocal
• Line detection usually performed on the output of an edge detector

Canny demo
Hough transform

- General idea: transform from image coordinates to parameter space of feature
  - Need parameterized model of features
  - For each pixel, determine all parameter values that might have given rise to that pixel; vote
  - At end, look for peaks in parameter space

Hough transform for lines

- Generic line: $y = ax + b$
- Parameters: $a$ and $b$

Hough transform for lines

1. Initialize table of buckets, indexed by $a$ and $b$, to zero
2. For each detected edge pixel $(x, y)$:
   a. Determine all $(a, b)$ such that $y = ax + b$
   b. Increment bucket $(a, b)$
3. Buckets with many votes indicate probable lines
Issues

- Slope / intercept parameterization not ideal
  - Non-uniform sampling of directions
  - Can’t represent vertical lines
- Angle / distance parameterization
  - Line represented as \((r, \theta)\) where \(x \cos \theta + y \sin \theta = r\)

Detection of corners

- Also known as features, interesting points, salient points or keypoints. Points that you can easily point out their correspondences in multiple images using only local information.

Moravec corner detector (1980)

- We should easily recognize the point by looking through a small window
- Shifting a window in *any direction* should give a large change in intensity
Moravec corner detector

Change of intensity for the shift \([u,v]\):

\[ E(u,v) = \sum_{x,y} w(x,y) \left[ I(x+u, y+v) - I(x, y) \right]^2 \]

Window function \(w(x,y) = 1\) in window, 0 outside

Four shifts: \((u,v) = (1,0), (1,1), (0,1), (-1, 1)\)

Look for local maxima in \(\min\{E\}\)
Problems of Moravec detector

- Noisy response due to a binary window function
- Only a set of shifts at every 45 degree is considered
- Only minimum of $E$ is taken into account

$\Rightarrow$ Harris corner detector (1988) solves these problems.

Harris corner detector

Noisy response due to a binary window function

- Use a Gaussian function

$$w(x, y) = \exp \left( -\frac{(x^2 + y^2)}{2\sigma^2} \right)$$

Window function $w(x, y)$ - [Gaussian]

Harris corner detector

Only a set of shifts at every 45 degree is considered

- Consider all small shifts by Taylor’s expansion

$$E(u, v) = \sum_{x, y} w(x, y) \left[ I(x + u, y + v) - I(x, y) \right]^2$$

$$= \sum_{x, y} w(x, y) \left[ I_x u + I_y v + O(u^2, v^2) \right]^2$$

$E(u, v) = Au^2 + 2Cuv + Bv^2$

$A = \sum_{x, y} w(x, y) I_x^2(x, y)$

$B = \sum_{x, y} w(x, y) I_y^2(x, y)$

$C = \sum_{x, y} w(x, y) I_x(x, y) I_y(x, y)$
Harris corner detector

Equivalently, for small shifts \([u,v]\) we have a \textit{bilinear} approximation:

\[
E(u, v) \cong [u \ v]M [u \ v]^	op
\]

, where \(M\) is a \(2 \times 2\) matrix computed from image derivatives:

\[
M = \sum_{x,y} w(x, y) \begin{bmatrix} I_x^2 & I_x I_y \\
I_x I_y & I_y^2 \end{bmatrix}
\]

High-level idea: what shape of the error function will we prefer for features?

Intensity change in shifting window: eigenvalue analysis

\[
E(u, v) \cong [u \ v]M [u \ v]^	op
\]

\(\lambda_1, \lambda_2\) – eigenvalues of \(M\)

Ellipse \(E(u,v) = \text{const}\)

\(E(\text{flat})\)  \(E(\text{edge})\)  \(E(\text{corner})\)

\(\lambda_{\text{max}}^{1/2}\)

\(\lambda_{\text{min}}^{1/2}\)

\(O(\text{fastest change})\)

\(O(\text{slowest change})\)

\(\mathbf{u}^T \mathbf{M} \mathbf{u}\) represents a quadratic function; Thus, we can analyze \(E\)'s shape by looking at the property of \(M\). Only minimum of \(E\) is taken into account

A new corner measurement by investigating the shape of the error function
**Harris corner detector**

Classification of image points using eigenvalues of M:

- **Corner**: \( \lambda_1 \) and \( \lambda_2 \) are large; \( \lambda_1 \approx \lambda_2 \); \( E \) increases in all directions
- **Edge**: \( \lambda_2 >> \lambda_1 \)
- **Flat**: \( \lambda_1 \) and \( \lambda_2 \) are small; \( E \) is almost constant in all directions

Measure of corner response:

\[
R = \text{det} M - k(\text{trace} M)^2
\]

\( k \) - empirical constant, \( k = 0.04-0.06 \)

**Summary of Harris detector**

1. Compute x and y derivatives of image
\[
I_x = G_x^\sigma * I \quad I_y = G_y^\sigma * I
\]
2. Compute products of derivatives at every pixel
\[
I_{x^2} = I_x * I_x \quad I_{y^2} = I_y * I_y \quad I_{xy} = I_x * I_y
\]
3. Compute the sums of the products of derivatives at each pixel
\[
S_{x^2} = G_\sigma * I_{x^2} \quad S_{y^2} = G_\sigma * I_{y^2} \quad S_{xy} = G_\sigma * I_{xy}
\]
Summary of Harris detector

4. Define the matrix at each pixel

\[ M(x, y) = \begin{bmatrix} S_{xx}(x, y) & S_{xy}(x, y) \\ S_{yx}(x, y) & S_{yy}(x, y) \end{bmatrix} \]

5. Compute the response of the detector at each pixel

\[ R = \det M - k(\text{trace}M)^2 \]

6. Threshold on value of R; compute nonmax suppression.
Local maximum of R

Harris corner detector

Mid-level vision

Segmentation and clustering

- Defining regions
  - Should they be compact? Smooth boundary?
- Defining similarity
  - Color, texture, motion, ...
- Defining similarity of regions
  - Minimum distance, mean, maximum
Clustering based on color

- Let’s make a few concrete choices:
  - Arbitrary regions
  - Similarity based on color only
  - Similarity of regions = distance between mean colors

**k-means Clustering**

1. Pick number of clusters $k$
2. Randomly scatter $k$ “cluster centers” in color space
3. Repeat:
   a. Assign each data point to its closest cluster center
   b. Move each cluster center to the mean of the points assigned to it
k-means Clustering

k-means Clustering
**k-means Clustering**

Results of Clustering

- Original Image
- k-means, k=5
- k-means, k=11

Sample clusters with k-means clustering based on color
Other Distance Measures

- Suppose we want to have compact regions
- New feature space: 5D
  (2 spatial coordinates, 3 color components)
- Points close in this space are close both in color and in actual proximity

Interactive segmentation

Matting

High-level vision
Recognition

Recognition problems

- What is it?
  - Object detection

- Who is it?
  - Recognizing identity

- What are they doing?
  - Activities

- All of these are **classification** problems
  - Choose one class from a list of possible candidates

Face detection

- How to tell if a face is present?

One simple method: skin detection

- Skin pixels have a distinctive range of colors
  - Corresponds to region(s) in RGB color space
  - For visualization, only R and G components are shown above

Skin classifier

- A pixel \( X = (R,G,B) \) is skin if it is in the skin region
- But how to find this region?
Skin detection

- Learn the skin region from examples
  - Manually label pixels in one or more “training images” as skin or not skin
  - Plot the training data in RGB space
    - skin pixels shown in orange, non-skin pixels shown in blue
      - some skin pixels may be outside the region, non-skin pixels inside. Why?

Skin classifier
- Given X = (R,G,B): how to determine if it is skin or not?

Skin classification techniques

Skin classifier
- Given X = (R,G,B): how to determine if it is skin or not?
  - Nearest neighbor
    - find labeled pixel closest to X
    - choose the label for that pixel
  - Data modeling
    - fit a model (curve, surface, or volume) to each class
  - Probabilistic data modeling
    - fit a probability model to each class

Probability

- Basic probability
  - X is a random variable
  - \( P(X) \) is the probability that X achieves a certain value
  - Conditional probability: \( P(X \mid Y) \)
    - probability of X given that we already know Y

Probabilistic skin classification

- Now we can model uncertainty
  - Each pixel has a probability of being skin or not skin
    - \( P(\sim \text{skin} \mid R) = 1 - P(\text{skin} \mid R) \)

Skin classifier
- Given X = (R,G,B): how to determine if it is skin or not?
  - Choose interpretation of highest probability
    - set X to be a skin pixel if and only if \( R_1 < X \leq R_2 \)
  - Where do we get \( P(\text{skin} \mid R) \) and \( P(\sim \text{skin} \mid R) \)?
Learning conditional PDF’s

- We can calculate $P(R \mid \text{skin})$ from a set of training images
  - It is simply a histogram over the pixels in the training images
    - each bin $R_i$ contains the proportion of skin pixels with color $R_i$

This doesn’t work as well in higher-dimensional spaces. Why not?

Approach: fit parametric PDF functions
  - common choice is rotated Gaussian
    - center $c = \bar{X}$
    - covariance $\sum (X - \bar{X})(X - \bar{X})^T$
      - orientation, size defined by eigenvectors, eigenvalues

Bayes rule

$$P(X \mid Y) = \frac{P(Y \mid X)P(X)}{P(Y)}$$

- In terms of our problem:
  - what we measure
  - domain knowledge
  - $P(\text{skin} \mid R) = \frac{P(R \mid \text{skin})P(\text{skin})}{P(R)}$

Bayesian estimation

$$P(R \mid \text{skin}) = \frac{\# \text{skin pixels with color } R}{\# \text{skin pixels}}$$

- We can calculate $P(R \mid \text{skin})$ from a set of training images
  - It is simply a histogram over the pixels in the training images
    - each bin $R_i$ contains the proportion of skin pixels with color $R_i$

But this isn’t quite what we want

- Why not? How to determine if a pixel is skin?
- We want $P(\text{skin} \mid R)$ not $P(R \mid \text{skin})$
- How can we get it?

Bayesian estimation

- Goal is to choose the label (skin or ~skin) that maximizes the posterior
  - this is called Maximum A Posteriori (MAP) estimation

The prior: $P(\text{skin})$

- Could use domain knowledge
  - $P(\text{skin})$ may be larger if we know the image contains a person
  - for a portrait, $P(\text{skin})$ may be higher for pixels in the center
- Could learn the prior from the training set. How?
  - $P(\text{skin})$ may be proportion of skin pixels in training set

Bayesian estimation = minimize probability of misclassification

= $P(\text{~skin})$
Skin detection results

Viola/Jones: features

"Rectangle filters"
Differences between sums of pixels in adjacent rectangles

\[ h_t(x) = \begin{cases} 
+1 & \text{if } f_t(x) > \theta_t \\
-1 & \text{otherwise} 
\end{cases} \]

60,000 \times 100 = 6,000,000 Unique Features

\[ Y(x) = \sum_\alpha \alpha_t h_t(x) \]
Select 200 by Adaboost

Detection = \{ face, \quad \text{if } Y(x) > 0 \\
non-face, \quad \text{otherwise} \}

Robust Realtime Face Detection, IJCV 2004, Viola and Jones

Viola/Jones: handling scale

Larger Scale

Smallest Scale

50,000 Locations/Scales

Viola/Jones results:

Run-time: 15fps (384x288 pixel image on a 700 Mhz Pentium III)
**Application**

Smart cameras: auto focus, red eye removal, auto color correction

**Application**

Lexus LS600 Driver Monitor System

**Face recognition**

- Suppose you want to recognize a *particular* face
- How does *this* face differ from average face
- Consider variation from average face
- Not all variations equally important
  - Variation in a single pixel relatively unimportant
- If image is high-dimensional vector, want to find directions in this space with high variation

**PCA**

- Principal Components Analysis (PCA): approximating a high-dimensional data set with a lower-dimensional subspace
Using PCA for Recognition

- Store each person as coefficients of projection onto first few principal components
  \[ \text{image} = \sum_{i=1}^{K} a_i \text{Eigenface}_i \]
- Compute projections of target image, compare to database (“nearest neighbor classifier”)

Choosing the dimension $K$

- How many eigenfaces to use?
- Look at the decay of the eigenvalues
  - the eigenvalue tells you the amount of variance “in the direction” of that eigenface
  - ignore eigenfaces with low variance
Advanced topics

High dynamic range imaging/display

Image warping/morphing

someone not that famous

someone very famous

video
Tracking

Feature tracking

Image stitching

Move matching using scene planes

MatchMove

Matchmove

Move matching using scene planes
Matchmove

Move matching using scene planes

Photo tourism

Video matching

Matrix  MOCO (Motion control camera)
Matting and compositing

Matting

Image manipulation

Image manipulation
Image-based modeling

Structured Light and Ranging Scanning

3D photography (active)

3D photography (active)
3D photography (passive)

Image-based rendering

View interpolation

View interpolation
Making face

- Gollum
- Spacetime face

Video rewrite

- Trainable videorealistic speech animation

Inpainting (wire removal)

- Inpainting

Texture synthesis/replacement

- Texture replacement
Semi-automatic matting painting

Image analogies

Video editing

Flow-based video editing

Face Detection and Recognition

Motion Estimation

Application

Andy Serkis, Gollum, Lord of the Rings
Reality?

Retouching

Iraq War, LA Times, April 2003

Bush campaign’s TV AD, 2004
Texture synthesis and inpainting

Production pipeline

Domestic example

The Liberty Times
2007.12.17
Related courses

What's next?