High Level Computer Vision

Basic Image Processing - April 25, 2018

Bernt Schiele - schiele@mpi-inf.mpg.de
Mario Fritz - mfritz@mpi-inf.mpg.de

mpi-inf.mpg.de/hlcv
Today - Basics of Digital Image Processing

- Linear Filtering
  - Gaussian Filtering

- Multi Scale Image Representation
  - Gaussian Pyramid, Laplacian Pyramid

- Edge Detection
  - ‘Recognition using Line Drawings’
  - Image derivatives (1st and 2nd order)

- Hough Transform
  - Finding parametrized curves, generalized Hough transform

- Object Instance Identification using Color Histograms

- (Several slides are taken from Michael Black @ Brown)
Computer Vision and its Components

- computer vision: ‘reverse’ the imaging process
  - 2D (2-dimensional) digital image processing
  - ‘pattern recognition’ / 3D image analysis
  - image understanding
Image Filtering: 2D Signals and Convolution

- Image Filtering
  - to reduce noise,
  - to fill-in missing values/information
  - to extract image features (e.g. edges/corners), etc

- Simplest case:
  - linear filtering: replace each pixel by a linear combination of its neighbors

- 2D convolution (discrete):
  - discrete Image: \( I[m,n] \)
  - filter ‘kernel’: \( g[k,l] \)
  - ‘filtered’ image: \( f[m,n] \)

\[
f[m, n] = I \otimes g = \sum_{k,l} I[m - k, n - l] g[k, l]
\]

(can be expressed as matrix multiplication!)
Linear Systems

• Basic Properties:
  ‣ homogeneity \( T[aX] = aT[X] \)
  ‣ additivity \( T[X_1 + X_2] = T[X_1] + T[X_2] \)
  ‣ superposition \( T[aX_1 + bX_2] = aT[X_1] + bT[X_2] \)

• examples:
  ‣ matrix operations (additions, multiplication)
  ‣ convolutions
Filtering to Reduce Noise

• “Noise” is what we’re not interested in
  - low-level noise: light fluctuations, sensor noise, quantization effects, finite precision, …
  - complex noise (not today): shadows, extraneous objects.

• Assumption:
  - the pixel’s neighborhood contains information about its intensity
Model: Additive Noise

- Image $I = \text{Signal } S + \text{Noise } N$:

$$S + N = I$$

Nearby points tell more about the signal than distant ones.

Neighborhood for averaging.
Model: Additive Noise

- Image $I = \text{Signal } S + \text{Noise } N$
  - I.e. noise does not depend on the signal

- we consider:
  - $I_i$: intensity of i’th pixel
  - $I_i = s_i + n_i$ with $E(n_i) = 0$
    - $s_i$ deterministic
    - $n_i, n_j$ independent for $i \neq j$
    - $n_i, n_j$ i.i.d. (independent, identically distributed)

- therefore:
  - intuition: averaging noise reduces its effect
  - better: smoothing as inference about the signal
Average Filter

- Average Filter
  - replaces each pixel with an average of its neighborhood
  - Mask with positive entries that sum to 1
- if all weights are equal, it is called a BOX filter

\[
\begin{array}{|c|c|c|}
\hline
\frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\
\hline
\frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\
\hline
\frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\
\hline
\end{array}
\]
Gaussian Averaging (An Isotropic Gaussian)

- Rotationally symmetric
- Weights nearby pixels more than distant ones
  - this makes sense as ‘probabilistic’ inference

- the pictures show a smoothing kernel proportional to

\[
g(x, y) = \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)
\]
Smoothing with a Gaussian

- Effects of smoothing:
  - each column shows realizations of an image of Gaussian noise
  - each row shows smoothing with Gaussians of different width
Smoothing with a Gaussian

- Example:
Efficient Implementation

- Both, the BOX filter and the Gaussian filter are separable:
  - first convolve each row with a 1D filter
  - then convolve each column with a 1D filter

\[
(f_x \otimes f_y) \otimes I = f_x \otimes (f_y \otimes I)
\]

- remember:
  - convolution is linear - associative and commutative

- Example: separable BOX filter

\[
\begin{pmatrix}
\frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\
\frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\
\frac{1}{9} & \frac{1}{9} & \frac{1}{9}
\end{pmatrix}
\begin{pmatrix}
\frac{1}{3} & \frac{1}{3} & \frac{1}{3}
\end{pmatrix}
\otimes
\begin{pmatrix}
\frac{1}{3} \\
\frac{1}{3} \\
\frac{1}{3}
\end{pmatrix}
\]
Example: Separable Gaussian

- Gaussian in x-direction

\[ g(x) = \frac{1}{\sqrt{2\pi\sigma}} \exp \left( -\frac{x^2}{2\sigma^2} \right) \]

- Gaussian in y-direction

\[ g(y) = \frac{1}{\sqrt{2\pi\sigma}} \exp \left( -\frac{y^2}{2\sigma^2} \right) \]

- Gaussian in both directions

\[ g(x, y) = \frac{1}{2\pi\sigma^2} \exp \left( -\frac{x^2 + y^2}{2\sigma^2} \right) \]
Multi-Scale Image Representation

- In this class:
  - Gaussian Pyramids
  - Laplacian Pyramids -> later
- Example of a Gaussian Pyramid

High resolution \rightarrow Low resolution
Motivation: Search across Scales

Irani & Basri
Computation of Gaussian Pyramid

Low resolution

\[ G_4 = (G_3 \ast \text{gaussian}) \downarrow 2 \]
\[ G_3 = (G_2 \ast \text{gaussian}) \downarrow 2 \]
\[ G_2 = (G_1 \ast \text{gaussian}) \downarrow 2 \]
\[ G_1 = (G_0 \ast \text{gaussian}) \downarrow 2 \]

High resolution

\[ G_0 = \text{Image} \]

Irani & Basri
Gaussian Pyramid

Questions of interest:

- which information is preserved over ‘scales’?
- which information is lost over ‘scales’?
Fourier Transform in Pictures

- a *very* little about Fourier transform to talk about spatial frequencies...

\[
3 \sin(x) + 1 \sin(3x) + 0.8 \sin(5x) + 0.4 \sin(7x) + \ldots = \]

\[
A + B + C + D
\]
Another Example

- a bar
  - in the big images is a hair (on the zebra’s nose)
  - in smaller images, a stripe
  - in the smallest image, the animal’s nose
Today - Basics of Digital Image Processing

- Linear Filtering
  - Gaussian Filtering

- Multi Scale Image Representation
  - Gaussian Pyramid, Laplacian Pyramid

- Edge Detection
  - ‘Recognition using Line Drawings’
  - Image derivatives (1st and 2nd order)

- Hough Transform
  - Finding parametrized curves, generalized Hough transform

- Object Instance Identification using Color Histograms

- (Several slides are taken from Michael Black @ Brown)
Image Edges:
What are edges? Where do they come from?

- Edges are changes in pixel brightness
Image Edges: What are edges? Where do they come from?

- Edges are changes in pixel brightness
  - Foreground/Background Boundaries
  - Object-Object-Boundaries
  - Shadow Edges
  - Changes in Albedo or Texture
  - Changes in Surface Normals
Line Drawings: Good Starting Point for Recognition?
Example of Recognition & Localization

- David Lowe

Parameters: 3D position and orientation

"match"

3D Model
Example of Recognition & Localization

- David Lowe
  - 1. ‘filter’ image to **find brightness changes**
  - 2. ‘**fit**’ **lines** to the raw measurements
Example of Recognition & Localization

- David Lowe
  - 3. ‘project’ model into the image and ‘match’ to lines (solving for 3D pose)
Class of Models

• Common Idea & Approach (in the 1980’s)
  ‣ matching of models (wire-frame/geons/generalized cylinders...) to edges and lines

• so the ‘only’ remaining problem to solve is:
  ‣ reliably extract lines & edges that can be matched to these models...
Actual 1D profile

- Barbara Image:
  - entire image

- line 250:
  - line 250 smoothed with a Gaussian:
What are ‘edges’ (1D)

- Idealized Edge Types:

- Goals of Edge Detection:
  - **good detection**: filter responds to edge, not to noise
  - **good localization**: detected edge near true edge
  - **single response**: one per edge
Edges

- Edges:
  - correspond to fast changes
  - where the magnitude of the derivative is large

"image" of 2 step-edges

single line of "image"

smoothing
Edges & Derivatives...

1st derivative

Maxima of first derivative

2nd derivative

“zero crossings” of second derivative
Compute Derivatives

\[
\frac{d}{dx} f(x) = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h} \approx f(x + 1) - f(x)
\]

- we can implement this as a linear filter:
  - direct:
    \[
    \begin{array}{cc}
    -1 & 1 \\
    \end{array}
    \]
  - or symmetric:
    \[
    \begin{array}{ccc}
    -1 & 0 & 1 \\
    \end{array}
    \]
Edge-Detection

- based on 1st derivative:
  - smooth with Gaussian
  - calculate derivative
  - finds its maxima

\[ \frac{d}{dx} (g \otimes f) \]
Edge-Detection

- Simplification:
  - remember: derivative as well as convolution are linear operations
  - saves one operation

\[
\frac{d}{dx} (g \otimes f) = \left( \frac{d}{dx} g \right) \otimes f
\]
1D Barbara signal

- **Barbara Image:**
  - entire image

- **line 250 (smoothed):**

- **1st derivative**
1D Barbara signal: note the amplification of small variations

- Barbara Image:
  - entire image

- line 250 (smoothed):

- 1st derivative
thresholding the derivative?
Implementing 1D edge detection

- algorithmically:
  - find peak in the 1st derivative

- but
  - should be a local maxima
  - should be ‘sufficiently’ large

- hysteresis: use 2 thresholds
  - high threshold to start edge curve (maximum value of gradient should be sufficiently large)
  - low threshold to continue them (in order to bridge “gaps” with lower magnitude)
  - (really only makes sense in 2D...)
Extension to 2D Edge Detection: Partial Derivatives

- partial derivatives
  - in x direction:
    \[
    \frac{d}{dx} I(x, y) = I_x \approx I \otimes D_x
    \]
  - in y direction:
    \[
    \frac{d}{dy} I(x, y) = I_y \approx I \otimes D_y
    \]
  - often approximated with simple filters (finite differences):
    \[
    D_x = \frac{1}{3} \begin{bmatrix}
    -1 & 0 & 1 \\
    -1 & 0 & 1 \\
    -1 & 0 & 1 \\
    \end{bmatrix}
    \]
    \[
    D_y = \frac{1}{3} \begin{bmatrix}
    -1 & -1 & -1 \\
    0 & 0 & 0 \\
    1 & 1 & 1 \\
    \end{bmatrix}
    \]
Finite Differences

Is this $I_x$ or $I_y$?
Is the sign right?
Finite Differences responding to noise

- increasing noise level (from left to right)
  - noise: zero mean additive Gaussian noise
Again: Derivatives and Smoothing

- derivative in x-direction: \( D_x \otimes (G \otimes I) = (D_x \otimes G) \otimes I \)

  - in 1D:
  
  ![1D Derivative Illustration](image)

  - in 2D:
  
  ![2D Derivative Illustration](image)
What is the gradient?

\[ \left( \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right) = (k, 0) \]

\[ \left( \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right) = (0, k) \]
What is the gradient?

\[
\left( \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right) = \left( k_x, k_y \right)
\]

- gradient direction is perpendicular to edge
- gradient magnitude measures edge strength
2D Edge Detection

- calculate derivative
  - use the **magnitude** of the gradient
  - the gradient is:
    \[ \nabla I = \left( I_x, I_y \right) = \left( \frac{\partial I}{\partial x}, \frac{\partial I}{\partial y} \right) \]
  - the magnitude of the gradient is:
    \[ \| \nabla I \| = \sqrt{I_x^2 + I_y^2} \]
  - the direction of the gradient is:
    \[ \theta = \arctan(I_y, I_x) \]
2D Edge Detection

- the scale of the smoothing filter affects derivative estimates, and also the semantics of the edges recovered
  - note: strong edges persist across scales

1 pixel | 3 pixels | 7 pixels
2D Edge Detection

- there are 3 major issues:
  - the gradient magnitude at different scales is different; which should we choose?
  - the gradient magnitude is large along a thick trail; how do we identify the significant points?
  - how do we link the relevant points up into curves?
'Optimal' Edge Detection: Canny

- Assume:
  - linear filtering
  - additive i.i.d. Gaussian noise

- Edge Detection should have:
  - **good detection**: filter response to edge, not noise
  - **good localization**: detected edge near true edge
  - **single response**: one per edge

- then: optimal detector is approximately derivative of Gaussian

- detection/localization tradeoff:
  - more smoothing improves detection
  - and hurts localization
The Canny edge detector

original image (Lena)

thresholding

norm (=magnitude) of the gradient

thinning (non-maximum suppression)

thresholding
Non-maximum suppression

• Check if pixel is local maximum along gradient direction
  ‣ choose the largest gradient magnitude along the gradient direction
  ‣ requires checking interpolated pixels p and r
Butterfly Example (Ponce & Forsyth)
line drawing vs. edge detection
University of South Florida

Match “model” to measurements?
Edges & Derivatives...

- recall:
  - the zero-crossings of the second derivative tell us the location of edges
Compute 2nd order derivatives

• 1st derivative:
\[
\frac{d}{dx} f(x) = \lim_{h \to 0} \frac{f(x + h) - f(x)}{h} \approx f(x + 1) - f(x)
\]

• 2nd derivative:
\[
\frac{d^2}{dx^2} f(x) = \lim_{h \to 0} \frac{\frac{d}{dx} f(x + h) - \frac{d}{dx} f(x)}{h} \approx \frac{d}{dx} f(x + 1) - \frac{d}{dx} f(x)
\]
\[
\approx f(x + 2) - 2f(x + 1) + f(x)
\]

• mask for
  ▶ 1st derivative:
  \[
  \begin{pmatrix}
  -1 & 1 \\
  \end{pmatrix}
  \]
  ▶ 2nd derivative:
  \[
  \begin{pmatrix}
  1 & -2 & 1 \\
  \end{pmatrix}
  \]
The Laplacian

- The Laplacian:

\[ \nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \]

- just another linear filter:

\[ \nabla^2 (G \otimes f) = \nabla^2 G \otimes f \]
Second Derivative of Gaussian

• in 1D:

• in 2D (‘mexican hat’):
1D edge detection

- using Laplacian

Laplacian of Gaussian operator

\[
\left( \frac{d^2}{dx^2} g \right) \otimes f
\]
Approximating the Laplacian

- Difference of Gaussians (DoG) at different scales:
The Laplacian Pyramid

\[ L_i = G_i - \text{expand}(G_{i+1}) \]

Gaussian Pyramid

\[ G_i = L_i + \text{expand}(G_{i+1}) \]

Laplacian Pyramid

\[ L_n = G_n \]

\[ L_0 \]

\[ L_1 \]

\[ L_2 \]

\[ G_0 \]

\[ G_1 \]

\[ G_2 \]

\[ G_n \]

Irani & Basri
Edge Detection with Laplacian

- sigma = 4
- sigma = 2
Edge Detection Today

- Still topic of active research after 40 years
- Today dominated by learning-based methods
- Quantitative Evaluation eg. on Berkeley Segmentation Data Set
  - 500 images & 5 Annotations per image

References

- **gPb-owt-ucm**: P. Arbelaez, M. Maire, C. Fowlkes and J. Malik: Contour Detection and Hierarchical Image Segmentation @ IEEE TPAMI, 2011

- **SE**: P. Dollar, C. Lawrence Zitnick: Fast Edge Detection using Structured Forests @ICCV’13 & IEEE TPAMI 2015

- **HED**: S. Xie, Z. Tu: Holistically-Nested Edge Detection @ ICCV 2015
Today - Basics of Digital Image Processing

- Linear Filtering
  - Gaussian Filtering
- Multi Scale Image Representation
  - Gaussian Pyramid, Laplacian Pyramid
- Edge Detection
  - ‘Recognition using Line Drawings’
  - Image derivatives (1st and 2nd order)
- Hough Transform
  - Finding parametrized curves, generalized Hough transform
- Object Instance Identification using Color Histograms

(Several slides are taken from Michael Black @ Brown)
Discussion

- edge detection + contour extraction
  - edges are defined as discontinuities in the image
  - we can assemble them, to obtain corresponding object contours
  - but contours do not necessarily correspond to object boundaries

- problem:
  - there is basically no knowledge used how object contours look like
  - obviously humans use such knowledge to segment objects
  - in principle: if we knew which object is in the image it would be much simpler to segment the object
Hough Transformation

- detection of straight lines
  - use the ‘knowledge’ that many contours belong to straight lines

- representation of a line: \( y = a \times + b \)
  - 2 parameters: \( a \) and \( b \) - determine all points of a line
  - this corresponds to a transformation: \((a,b) \rightarrow (x,y)\)
    - \( y = a \times + b \)
  - inverse interpretation: transformation of \((x,y) \rightarrow (a,b)\)
    - \( b = (-x)a + y \)
  - usage: points, for which the magnitude of the first derivate is large, lie potentially on a line
**Hough Transformation**

- for a particular point \((x,y)\) determine all lines which go through this point:
  - the parameters of all those lines are given by: \(b = (-x)a + y\)
  - i.e. those lines are given by a line in the parameter space \((a,b)\)
Hough Transformation

- implementation:
  - the parameter space \((a,b)\) has to be discretized
  - for each candidate \((x,y)\) for a line, store the line
    \[ b = (-a) x + y \]
  - in principle each candidate \((x,y)\) votes for the discretized parameters
  - the maxima in the parameter space \((a,b)\) correspond to lines in the image

- problem of this particular parameterization
  - the parameter ‘a’ can become infinite (for vertical lines)
  - problematic for the discretization
Hough Transformation

- choose another parameterization:

\[ x \cos(\theta) + y \sin(\theta) = \rho \]

- for this parameterization the domain is limited:
  - \( \rho \) is limited by the size of the image
  - and \( \theta \in [0, 2\pi] \)
Examples

- Hough transform for a square (left) and a circle (right)
Examples
Hough Transform

- the same idea can be used for other parameterized contours
  - Example:
    - circle: \((x-a)^2 + (y-b)^2 = r^2\)
    - 3 parameters: center point \((a, b)\) and radius \(r\)

- Limitation:
  - the parameter space should not become too large
  - not all contours can be parameterized
Generalized Hough Transform

• Generalization for an arbitrary contour
  ▸ choose reference point for the contour (e.g. centre)
  ▸ for each point on the contour remember where it is located w.r.t. to the reference point
  ▸ e.g. if the center is the reference point: remember radius $r$ and angle relative to the tangent of the contour
  ▸ recognition: whenever you find a contour point, calculate the tangent angle and ‘vote’ for all possible reference points
Today - Basics of Digital Image Processing

- Linear Filtering
  - Gaussian Filtering
- Multi Scale Image Representation
  - Gaussian Pyramid, Laplacian Pyramid
- Edge Detection
  - ‘Recognition using Line Drawings’
  - Image derivatives (1st and 2nd order)
- Hough Transform
  - Finding parametrized curves, generalized Hough transform
- Object Instance Identification using Color Histograms

(Several slides are taken from Michael Black @ Brown)
Object Recognition (reminder)

- Different Types of Recognition Problems:
  - **Object Identification**
    - recognize your apple, your cup, your dog
    - sometimes called: “instance recognition”
  - **Object Classification**
    - recognize any apple, any cup, any dog
    - also called: generic object recognition, object categorization, …
    - typical definition: ‘basic level category’
Object Identification

- Example Database for Object Identification:
  - COIL-100 - Columbia Object Image Library
  - contains 100 different objects, some form the same object class (e.g. cars, cups)
Challenges = Modes of Variation

- Viewpoint changes
  - Translation
  - Image-plane rotation
  - Scale changes
  - Out-of-plane rotation

- Illumination
- Clutter
- Occlusion
- Noise
Appearance-Based Identification / Recognition

• Basic assumption
  ‣ Objects can be represented by a collection 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
Global Representation

• Idea
  ‣ Represent each view (of an object) by a global descriptor.
  ‣ For recognizing objects, just match the (global) descriptors.
  ‣ Modes of variation can be taken care of by:
    - built into the descriptor
      • e.g. a descriptor can be made invariant to image-plane rotations, translation
    - incorporate in the training data or the recognition process.
      • e.g. viewpoint changes, scale changes, out-of-plane rotation
    - robustness of descriptor or recognition process (descriptor matching)
      • e.g. illumination, noise, clutter, partial occlusion
Case Study: Use Color for Recognition

- Color:
  - Color stays constant under geometric transformations
  - Local feature
    - Color is defined for each pixel
    - Robust to partial occlusion

- Idea
  - Directly use object colors for identification / recognition
  - Better: use statistics of object colors
Color Histograms

- Color statistics
  - Given: tri-stimulus R,G,B for each pixel
  - Compute 3D histogram
    - \( H(R,G,B) = \#(\text{pixels with color } (R,G,B)) \)

[Swain & Ballard, 1991]
Color Histograms

- Robust representation
  - presence of occlusion, rotation

[Swain & Ballard, 1991]
Color

- One component of the 3D color space is intensity
  - If a color vector is multiplied by a scalar, the intensity changes, but not the color itself.
  - This means colors can be normalized by the intensity.
    - Intensity is given by: $I = R + G + B$:
  - „Chromatic representation“
    \[
    r = \frac{R}{R + G + B}
    \]
    \[
    g = \frac{G}{R + G + B}
    \]
    \[
    b = \frac{B}{R + G + B}
    \]
Color

- Observation:
  - Since \( r + g + b = 1 \), only 2 parameters are necessary
  - E.g. one can use \( r \) and \( g \)
  - and obtains \( b = 1 - r - g \)

\[
\begin{align*}
r + g + b &= 1 \\
\Rightarrow b &= 1 - r - g
\end{align*}
\]
Recognition using Histograms

- Histogram comparison
  - Database of known objects
  - Test image of unknown object

[Images of a test image and known objects with corresponding histograms]
Recognition using Histograms

- Database with multiple training views per object
Histogram Comparison

- Comparison measures
  - Intersection
    \[ \cap (Q, V) = \sum_i \min(q_i, v_i) \]

- Motivation
  - Measures the common part of both histograms
  - Range: [0, 1]
  - For unnormalized histograms, use the following formula
    \[ \cap (Q, V) = \frac{1}{2} \left( \frac{\sum_i \min(q_i, v_i)}{\sum_i q_i} + \frac{\sum_i \min(q_i, v_i)}{\sum_i v_i} \right) \]
Histogram Comparison

- Comparison Measures
  - Euclidean Distance
    \[d(Q, V) = \sum_i (q_i - v_i)^2\]

- Motivation
  - Focuses on the differences between the histograms
  - Range: \([0, \infty]\)
  - All cells are weighted equally.
  - Not very discriminant
Histogram Comparison

- Comparison Measures
  - Chi-square
    \[ \chi^2(Q, V) = \sum_i \frac{(q_i - v_i)^2}{q_i + v_i} \]

- Motivation
  - Statistical background:
    - Test if two distributions are different
    - Possible to compute a significance score
  - Range: \([0, \infty]\)
  - Cells are not weighted equally!
    - therefore more discriminant
    - may have problems with outliers (therefore assume that each cell contains at least a minimum of samples)
Histogram Comparison

- Which measure is best?
  - Depends on the application…
  - Both Intersection and $\chi^2$ give good performance.
    - Intersection is a bit more robust.
    - $\chi^2$ is a bit more discriminative.
    - Euclidean distance is not robust enough.
  - There exist many other measures
    - e.g. statistical tests: Kolmogorov-Smirnov
    - e.g. information theoretic: Kullback-Leiber divergence, Jeffrey divergence, ...
Recognition using Histograms

- Simple algorithm
  1. Build a set of histograms $H = \{M_1, M_2, M_3, \ldots\}$ for each known object
     - more exactly, for each view of each object
  2. Build a histogram $T$ for the test image.
  3. Compare $T$ to each $M_k \in H$
     - using a suitable comparison measure
  4. Select the object with the best matching score
     - or reject the test image if no object is similar enough.

“Nearest-Neighbor” strategy
Color Histograms

• Recognition (here object identification)
  ‣ Works surprisingly well
  ‣ In the first paper (1991), 66 objects could be recognized almost without errors

[Swain & Ballard, 1991]
Discussion: Color Histograms

- **Advantages**
  - Invariant to object translations
  - Invariant to image rotations
  - Slowly changing for out-of-plane rotations
  - No perfect segmentation necessary
  - Histograms change gradually when part of the object is occluded
  - Possible to recognize deformable objects
    - e.g. pullover

- **Problems**
  - The pixel colors change with the illumination („color constancy problem“)
    - Intensity
    - Spectral composition (illumination color)
  - Not all objects can be identified by their color distribution.