

- Surface Features
  - Lecture Notes
    - Colour
    - Texture
- Traditional Object Recognition
  - Lecture Notes

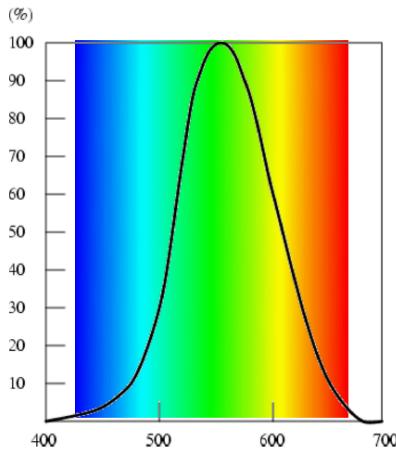
# Surface Features

## Lecture Notes

### Colour

#### The perception of brightness

- Luminous efficiency function  $v(\lambda)$ : relates radiometry & photometry
- C.I.E.(Commission Internationale de l'Eclairage) → standards



#### Link radiometry-photometry (Watt to lumen)

- **Photometry:** subjective impressions  
**Radiometry:** objective, physical measurements  
 at 555 nm :  $1\text{lm} = 1/683 \text{ W} = 1.46 \text{ mW}$
- for light with spectral composition  $c(\lambda)$  (radiant flux)  

$$l = k \int_0^{\infty} c(\lambda)v(\lambda)d\lambda$$
 where k is 683 lumens/watt

#### The study of colour...

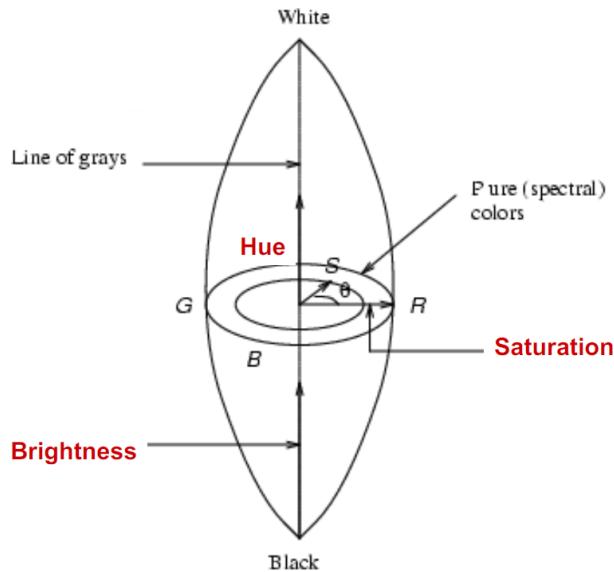
Use:

- pleasing to the eye (visualisation of results)

- characterising colours (features e.g. for recognition)
- generating colours (displays, light for inspection)
- understanding human vision

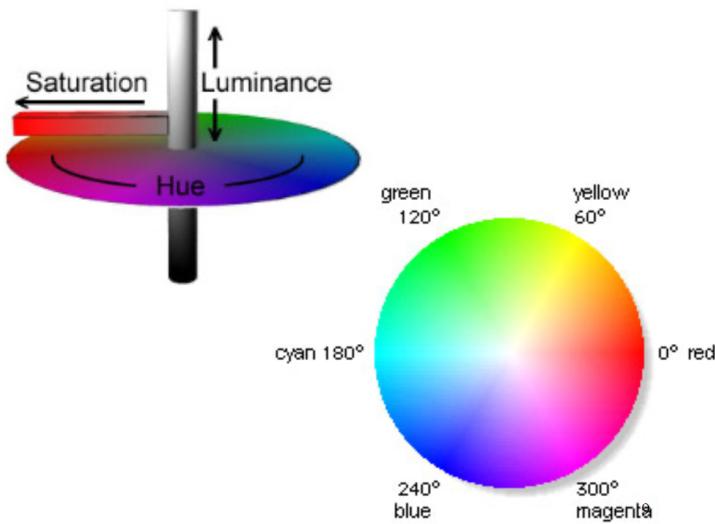
## The perceptual attributes of light (humans)

1. brightness
2. hue
3. saturation



## The C.I.E. color space

Human perception of light = 3-dimensional



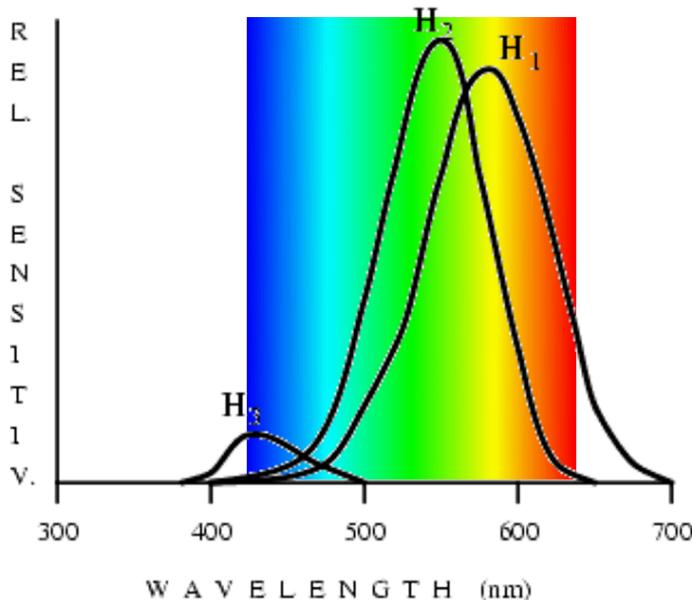
## The history of colour

- Newton → spectrum
- Young → tristimulus model

later: physiological underpinning: 3 cone types

## The retinal cones

- 3 types : blue, green, yellow-green



## Prediction of colour sensation

- source with spectral radiant flux  $C(\lambda)$  produces responses  $R_i$  with  $i = 1$ (red),  $2$ (green),  $3$ (blue)  
$$R_i(c) = \int H_i(\lambda)C(\lambda)d\lambda, \quad i = 1, 2, 3$$
- Hence, our perception of multi-spectral sources is quite impoverished:  
an entire distribution over  $\lambda$  is projected onto only 3 numbers  $R_i$ .
- 2 sources with equal  $R_i$ 's  $\Rightarrow$  observed as same colour!
- luminance  $\perp$  chrominance
- 10% of population have abnormal colour vision
- several birds have 4 cone types (incl. UV)
- colour constancy

## Tristimulus representation of colour

- Camera tristimulus values display
- 3 primaries  $P_j(\lambda)$ ,  $j = 1, 2, 3$
- CIE primaries:  
$$\lambda_1 = 700, \lambda_2 = 546.1, \lambda_3 = 435.8$$
- applications: practical primaries e.g. TV: EBU and NTSC

## The matching of colour

- source  $C(\lambda)$  matched by primaries  $\sum_{j=1}^3 m_j P_j(\lambda)$   
 $R_i(c) = \int H_i(\lambda) C(\lambda) d\lambda = \int H_i(\lambda) \sum_{j=1}^3 m_j P_j(\lambda) d\lambda = \sum_{j=1}^3 m_j \int H_i(\lambda) P_j(\lambda) d\lambda$   
denote  $l_{i,j} = \int H_i(\lambda) P_j(\lambda) d\lambda$ , **can be determined “off-line”**

## The math

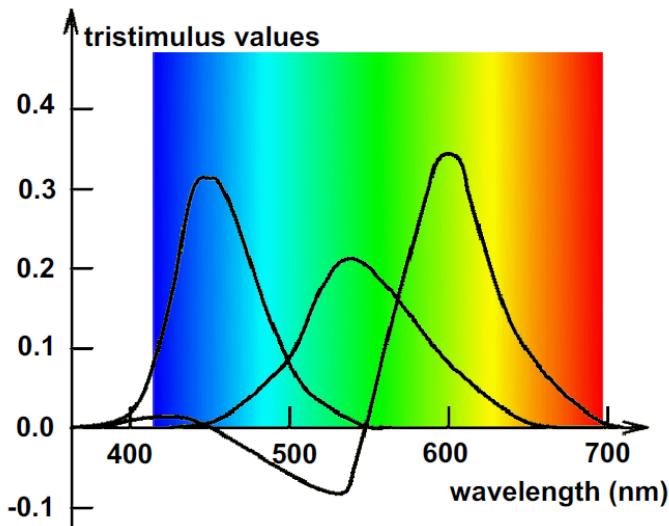
- extremely simple: linear equations  
 $\sum_{j=1}^3 m_j l_{i,j} = R_i$
- implies inverting the matrix: independent primaries!
- also linear transform between the  $m_j$ 's for different choices of primaries:  
 $Lm = R$  and  $L'm' = R \Rightarrow m' = L'^{-1}Lm$

## Tristimulus values

- “white” considered a reference: specify relative values w.r.t.  $m_j$  s for white:  $w_j$
- Tristimulus values:  $T_j = \frac{m_j}{w_j}$
- The scaling preserves the linearity
- CIE tristimulus values: R, G, B  
(for CIE white = flat spectrum &  $w_1 = w_2 = w_3$ )
- Note that for white  $T_1 = T_2 = T_3 = 1$

## Spectral matching curves

- Spectral matching curves  $T_j(\lambda)$
- values for monochromatic sources  
 $R_i(C_\lambda) = H_i(\lambda) = \sum_{j=1}^3 m_j l_{i,j} = \sum_{j=1}^3 w_j l_{i,j} T_j(\lambda)$
- for the CIE primaries:



## Interpretation of these curves

- negative values: colours that cannot be produced
- In such case:  $\text{mix}(\text{target}, \text{neg.primary}) = \text{mix}(\text{pos.primaries})$
- for **any** primary triple **some colours cannot be produced**
- for arbitrary source  $C(\lambda)$   

$$T_j(C) = \int C(\lambda)T_j(\lambda)d\lambda$$

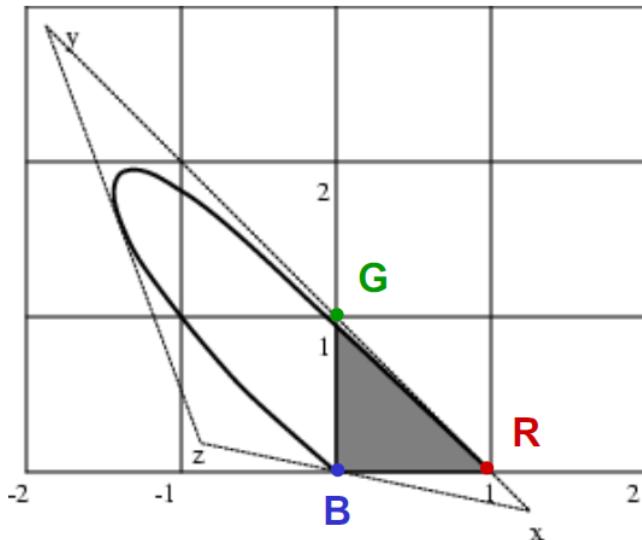
## Chromaticity coordinates

- tristimulus values still contain brightness info  
pure chrominance info: normalising the tristimulus values
- **chromaticity coordinates:**  $t_j = \frac{T_j}{T_1+T_2+T_3}$
- $t_1 + t_2 + t_3 = 1$  allows to eliminate one
- 2 chromaticity coordinates specify saturation and hue
- Note that for white:  $t_1 = t_2 = t_3 = 1/3$

## CIE chromaticity diagram

- chromaticity coordinates  $(r, g)$  for CIE primaries:  

$$r = \frac{R}{R+G+B} \quad g = \frac{G}{R+G+B}$$
- The corresponding colour space:



## CIE x-y coordinates

- In order to get rid of the negative values: virtual tristimulus colour system  $X, Y, Z$   
(no such physically realizable primaries exist!)
- linear transf. from  $R, G, B$  to  $X, Y, Z$  coordinates:

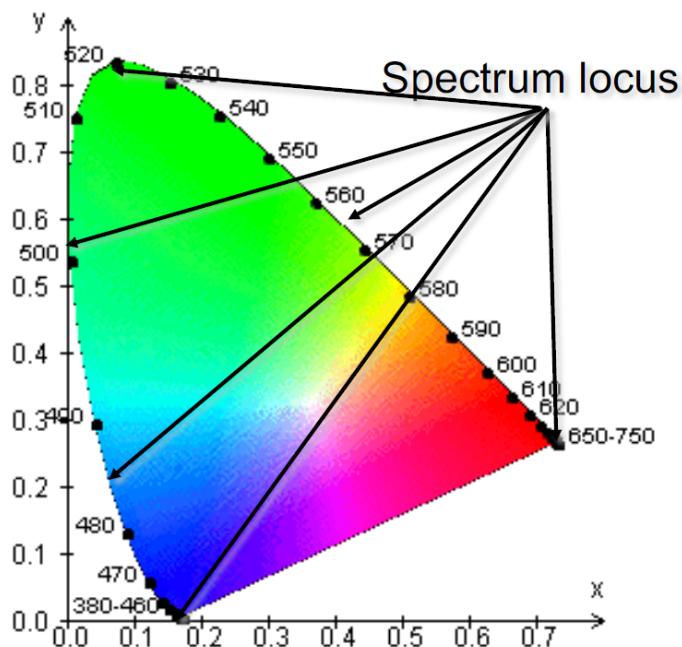
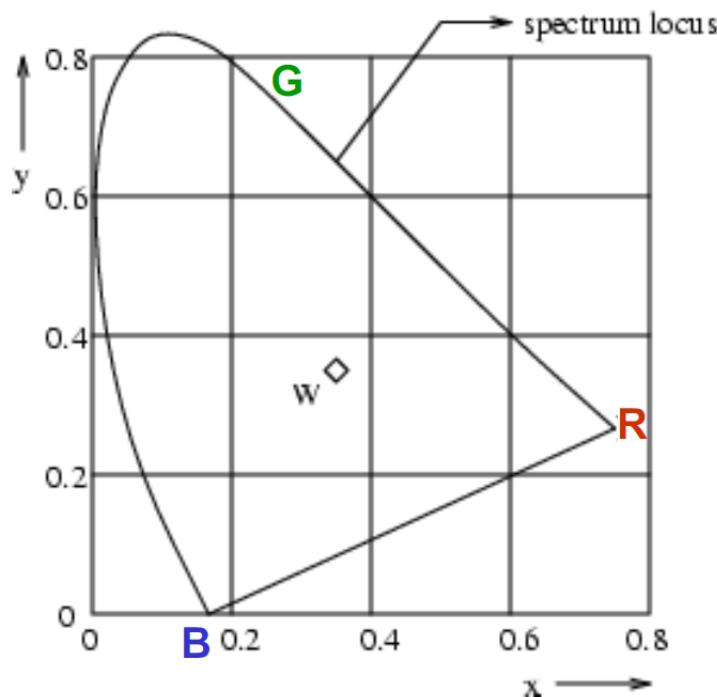
$$\begin{pmatrix} X \\ Y \\ Z \end{pmatrix} = \begin{pmatrix} 0.490 & 0.310 & 0.200 \\ 0.177 & 0.813 & 0.011 \\ 0.000 & 0.010 & 0.990 \end{pmatrix} \begin{pmatrix} R \\ G \\ B \end{pmatrix}$$

- chosen as to make Y represent luminance

with (R=G=B=1) mapped to X=Y=Z=1

$$x = \frac{X}{X+Y+Z} \quad y = \frac{Y}{X+Y+Z}$$

### CIE x,y colour triangle



TV primaries

the EBU primaries have coordinates

$$R_r : \quad x = 0.64 \quad y = 0.33$$

$$G_r : \quad x = 0.29 \quad y = 0.60$$

$$B_r : \quad x = 0.15 \quad y = 0.06$$

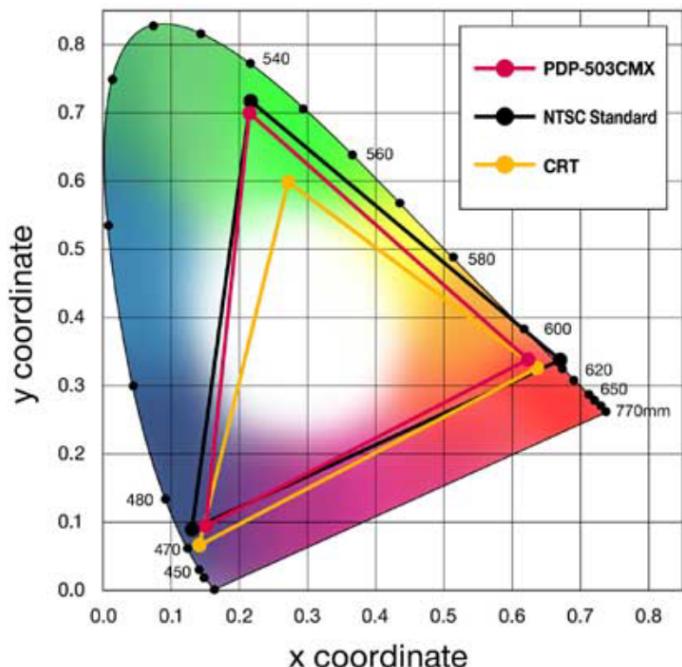
the NTSC primaries have coordinates

$$R_N : \quad x = 0.67 \quad y = 0.33$$

$$G_N : \quad x = 0.21 \quad y = 0.71$$

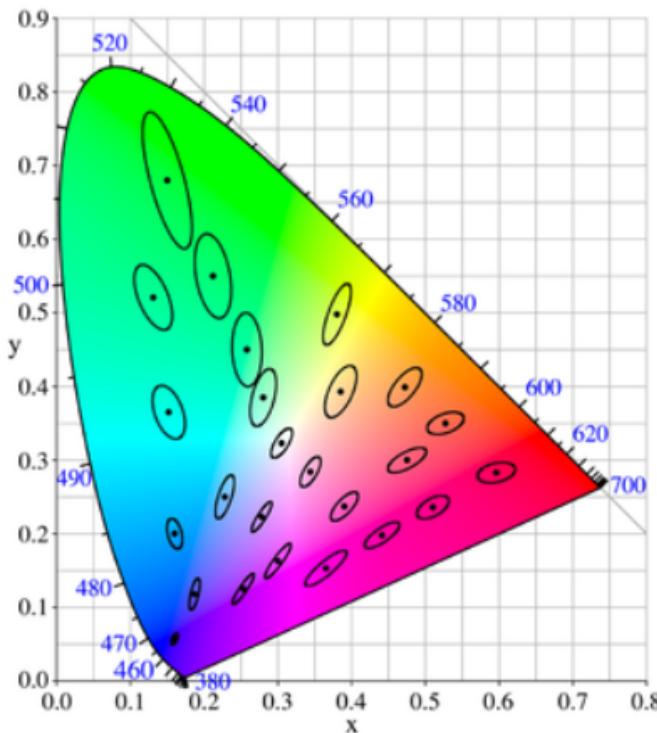
$$B_N : \quad x = 0.14 \quad y = 0.08$$

## CIE Chromaticity Coordinates



## Notes

- Minimize colours outside the triangle!
- Area dubious criterion:
  - projective transf. between chromaticity coordinates
  - distance in triangle no faithful indication of perceptual difference



- pure spectrum colours (on the spectrum locus) are rare in nature

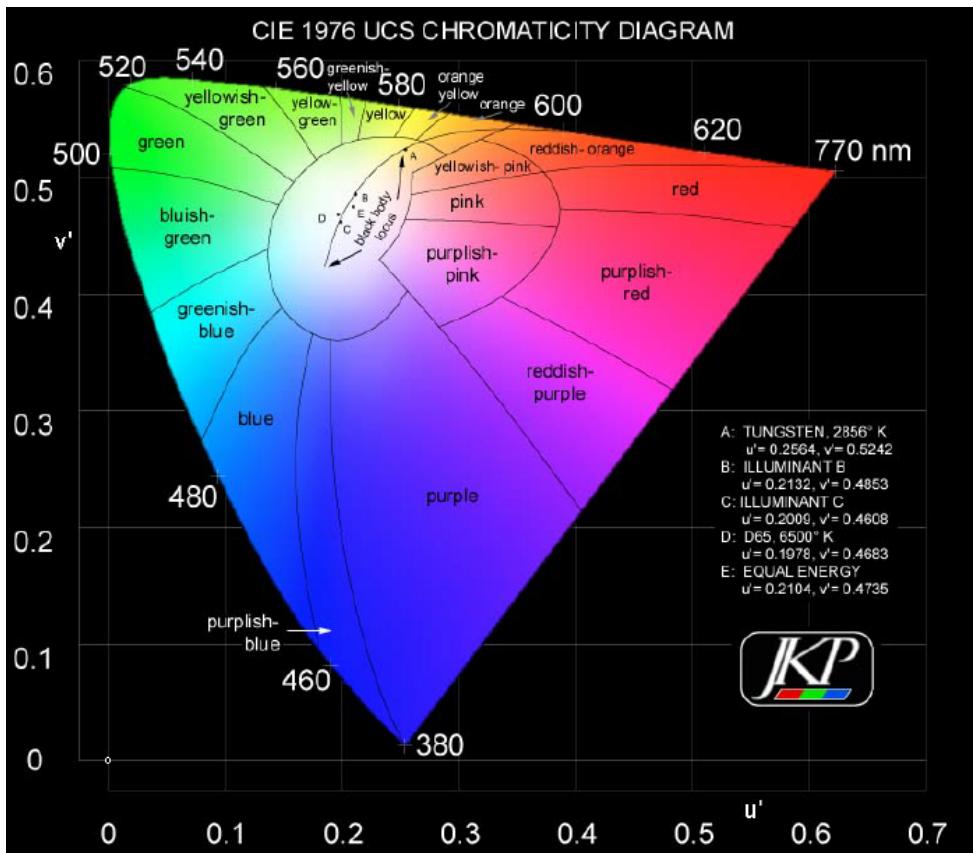
### Chromaticity coordinate transitions

- So, colour coordinates need for their definition  
3 primaries + white: 4 points  
4 points define a projective frame:  
primaries  $\rightarrow (0,0), (1,0), (0,1)$   
white  $\rightarrow (0.33,0.33)$   
a chromaticity coordinate transformation can be shown to be projective, i.e. non-linear

### CIE u-v color coordinates

- u-v diagram +/- faithfully represents perceptual distance:

$$u = \frac{4x}{-2x+12y+3} \quad v = \frac{6x}{-2x+12y+3}$$



## Using colour as a feature

### Koffka ring with colours

### Colour constancy

- Patches keep their colour appearance even if they reflect differently (e.g. the hat)
- Patches change their colour appearance if they reflect identically but surrounding patches reflect differently (e.g. the background)
- There is more to colour perception than 3 cone responses
- Edwin Land performed in-depth experiments (psychophysics)

### Colour constancy - notes

- The colour of a surface is the result of the product of spectral reflectance and spectral light source composition
- Our visual system can from a single product determine the two factors, it seems
- The colour of the light source can be guessed via that of specular reflections, but the visual system does not critically depend on this trick

### Illumination invariant colour features

- Extracting the true surface colour under varying illumination - as the **HVS** can - is very difficult
- A less ambitious goal is to extract colour features that do not change with illumination

## 1. Spectral or 'internal' changes

- Let  $I_R, I_G, I_B$  represent the irradiances at the camera for red, green, blue
- A simple model: the irradiances change by  
 $\alpha, \beta, \gamma : (I'_R, I'_G, I'_B) = (\alpha I_R, \beta I_G, \gamma I_B)$
- Consider irradiances at 2 points:  
 $I_{R1}, I_{G1}, I_{B1}$  and  $I_{R2}, I_{G2}, I_{B2}$   
 $I'_{R1}/I'_{R2} = (\alpha I_{R1})/(\alpha I_{R2}) = I_{R1}/I_{R2}$
- For a camera with a non-linear response:  
 $(\alpha I_{R1})^\gamma / (\alpha I_{R2})^\gamma = (I_{R1})^\gamma / (I_{R2})^\gamma$

## 2. Geometric or 'external' changes

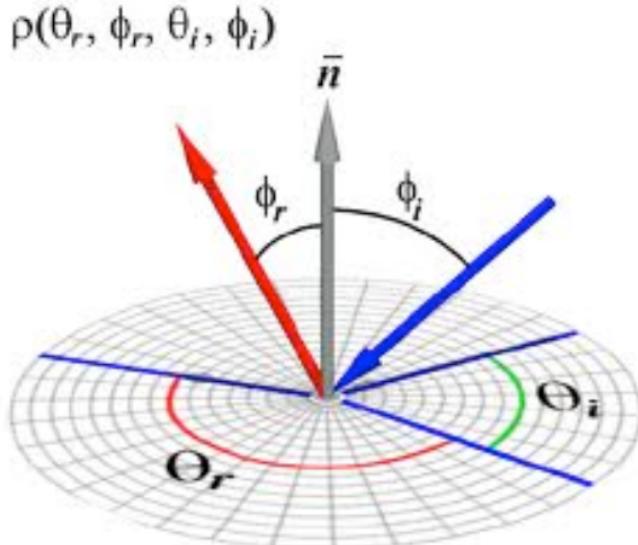
- $(I'_R, I'_G, I'_B) = (s(x,y)I_R, s(x,y)I_G, s(x,y)I_B)$   
 $I'_R/I'_G = I_R/I_G$  and  $I'_R/I'_B = I_R/I_B$  are invariant

## 3. Spectral + geometric changes

- $\frac{I'_{R1} I'_{G2}}{I'_{R2} I'_{G1}} = \frac{\alpha s(x_1, y_1) I_{R1} \beta s(x_2, y_2) I_{G2}}{\alpha s(x_2, y_2) I_{R2} \beta s(x_1, y_1) I_{G1}} = \frac{I_{R1} I_{G2}}{I_{R2} I_{G1}}$
- for points on both sides of a colour edge  $s(x_1, y_1) \approx s(x_2, y_2)$   
and hence  $I_{R1}/I_{R2}$  is invariant

## The elusive BRDF

- Bidirectional Reflection Distribution Function... for different wavelengths



- A 4D function, specifying the radiance for an outgoing direction given an irradiance for an incoming direction, relative to the normal and ideally for 1 wavelength at a time

## Mini-dome to study reflectance

## Texture

## Example textures

### Texture characteristics

- oriented vs. isotropic
- regular vs. stochastic
- coarse vs. fine

### Fourier features

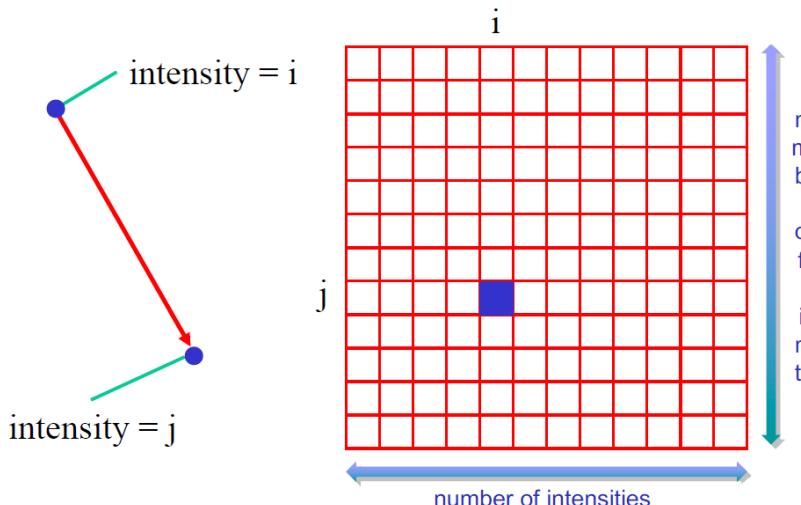
- Based on the integration of regions of the Fourier power spectrum  
$$\int_A \int |F(u, v)|^2 dudv$$
- Intuitively appealing
  - peaks if periodic
  - mostly low/high freq. if coarse resp. fine
  - the sine patterns each have an orientation
- coarseness:  $r_1^2 < u^2 + v^2 < r_2^2$
- directionality:  $\theta_1 \leq \arctan\left(\frac{u}{v}\right) < \theta_2$
- The Fourier Transform collects information globally over the entire image  
Not good for segmentation or inspection

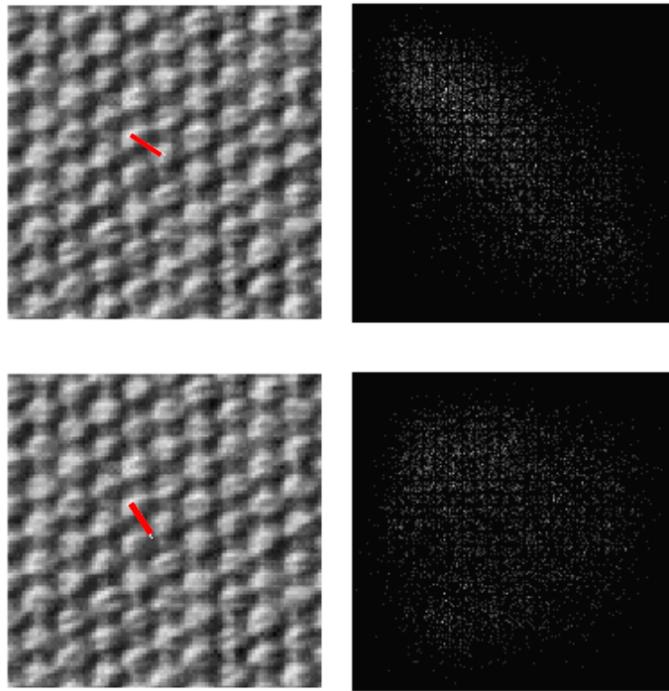
### Histograms: principle

- Intensity probability distribution
- Captures global brightness information in a compact, but incomplete way
- Doesn't capture spatial relationships

### Cooccurrence matrix

- probability distributions for intensity pairs
- Contains information on some aspects of the spatial configurations





- Features calculated from the matrix:

feature	expression
energy	$\sum_i \sum_j C^2(i, j)$
entropy	$-\sum_i \sum_j C(i, j) \log C(i, j)$
contrast	$\sum_i \sum_j (i - j)^2 C(i, j)$
homogeneity	$\sum_i \sum_j C(i, j) / (1 +  i - j )$
max. probability	$\max_{i,j} C(i, j)$

## Laws filters

- This fixed filter set yields simple convolutions but has proven very effective in some cases

Feature	1D filter
L3	[ 1 2 1 ]
E3	[ -1 0 1 ]
S3	[ -1 2 -1 ]
L5	[ 1 4 6 4 1 ]
E5	[ -1 -2 0 2 1 ]
S5	[ -1 0 2 0 -1 ]
W5	[ -1 2 0 -2 1 ]
R5	[ 1 -4 6 -4 1 ]
L7	[ 1 6 15 20 15 6 1 ]
E7	[ -1 -4 -5 0 5 4 1 ]
S7	[ -1 -2 1 4 1 -2 -1 ]
W7	[ -1 0 3 0 -3 0 1 ]
R7	[ 1 -2 -1 4 -1 -2 1 ]
O7	[ -1 6 -15 20 -15 6 -1 ]

## Gabor filters

- Gaussian envelope multiplied by cosine

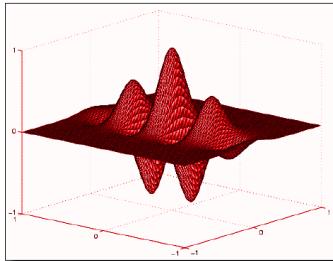
$$g(x, y) = \exp\left(-\frac{x^2+y^2}{4\Delta_{x,y}^2}\right) \cos(2\pi u^*x + \varphi)$$

- The filter's Fourier power spectrum

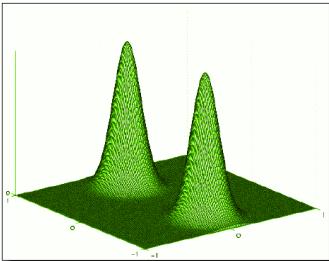
$$G(u, v) = \frac{1}{4\pi\Delta_{u,v}^2} \left( \exp\left[-\frac{(u-u^*)^2+v^2}{4\Delta_{u,v}^2}\right] + \exp\left[-\frac{(u+u^*)^2+v^2}{4\Delta_{u,v}^2}\right] \right)$$

- Good localisation in both domains:

Spatial domain



Frequency domain



$$f = f(x, y) \Leftrightarrow F = F(u, v)$$

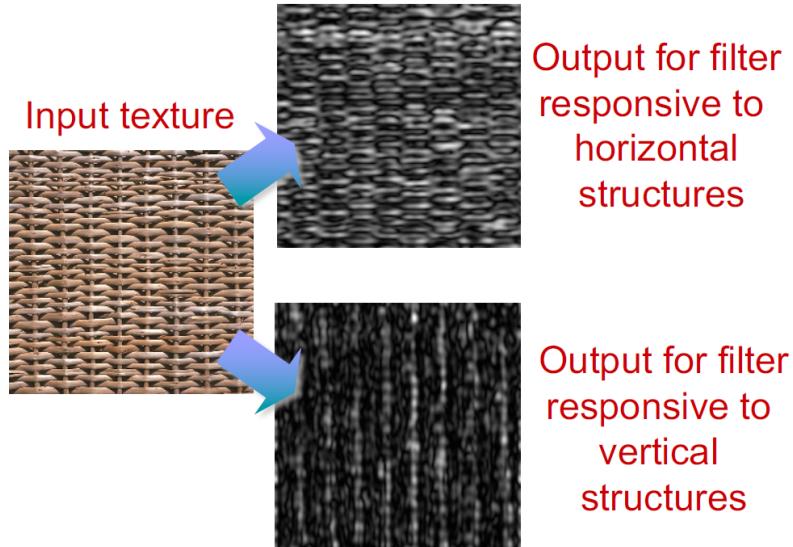
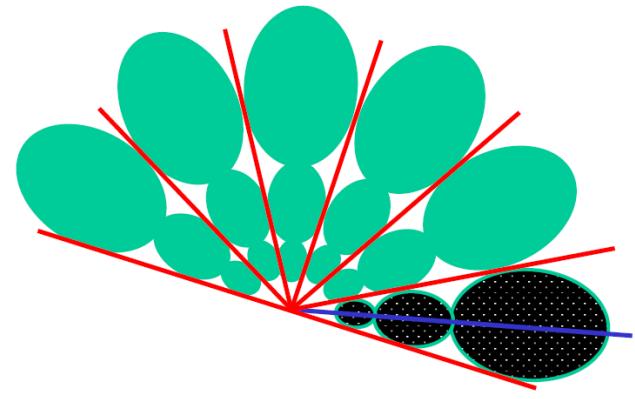
$$\bullet \quad x_{av} = \frac{\int_{-\infty}^{\infty} x f \bar{f} dx}{\int_{-\infty}^{\infty} f \bar{f} dx} \Leftrightarrow u_{av} = \frac{\int_{-\infty}^{\infty} u F \bar{F} du}{\int_{-\infty}^{\infty} F \bar{F} du}$$

$$(\Delta x)^2 = \frac{\int_{-\infty}^{\infty} (x - x_{av})^2 f \bar{f} dx}{\int_{-\infty}^{\infty} f \bar{f} dx} \Leftrightarrow (\Delta u)^2 = \frac{\int_{-\infty}^{\infty} (u - u_{av})^2 F \bar{F} du}{\int_{-\infty}^{\infty} F \bar{F} du}$$

The Heisenberg uncertainty principle:  $\Delta x \Delta u = \frac{1}{4}\pi$

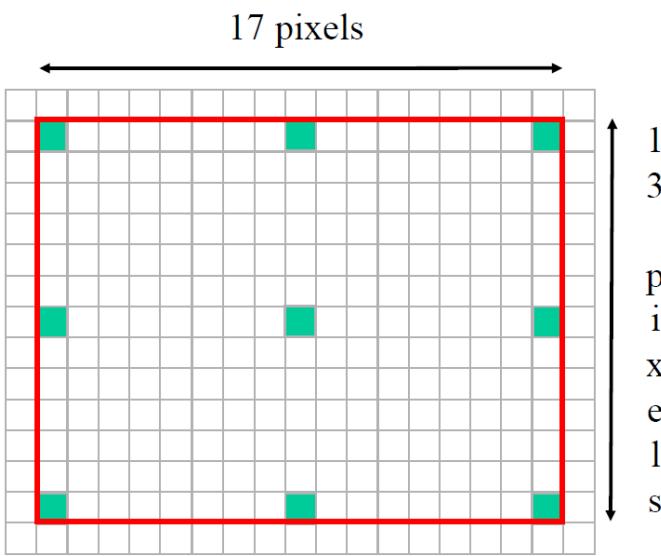
- Covering the Fourier domain with responses

- to probe for directionality
- to look at different scales



## Eigenfilters (Ade, ETH)

- Filters adapted to the texture
  1. shift mask over training image
  2. collect intensity statistics
  3. PCA → eigenvectors → 'eigenfilters'
  4. energies of eigenfilter outputs
- but small filters may reduce efficacy
- hence large, but sparse filters
- Example applications: textile inspection  
Filters with size of one period  
(period found as peak in autocorrelation)



- Covariance matrix needed for PCA

$$\begin{pmatrix} 174.1 & -77.6 & 101.6 & -60.8 & 72.7 & -71.5 & 116.5 & -77.9 & 91.4 \\ -77.6 & 173.9 & -78.2 & 71.5 & -61.7 & 73.1 & -76.4 & 116.4 & -78.4 \\ 101.6 & -78.2 & 173.5 & -70.4 & 71.7 & -62.1 & 95.3 & -77.0 & 116.3 \\ -60.8 & 71.5 & -70.4 & 173.9 & -76.5 & 101.0 & -59.4 & 71.8 & -70.1 \\ 72.7 & -61.7 & 71.7 & -76.5 & 173.7 & -77.1 & 70.6 & -60.3 & 72.1 \\ -71.5 & 73.1 & -62.1 & 101.0 & -77.1 & 173.4 & -69.3 & 70.9 & -60.7 \\ 116.5 & -76.4 & 95.3 & -59.4 & 70.6 & -69.3 & 173.4 & -75.3 & 99.8 \\ -77.9 & 116.4 & -77.0 & 71.8 & -60.3 & 70.9 & -75.3 & 173.2 & -75.9 \\ 91.4 & -78.4 & 116.3 & -70.1 & 72.1 & -60.7 & 99.8 & -75.9 & 172.8 \end{pmatrix}$$

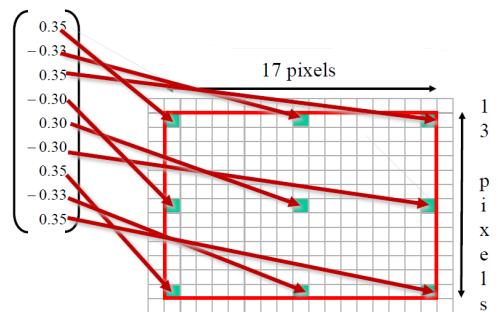
$$\begin{pmatrix} 0.35 \\ -0.33 \\ 0.35 \\ -0.30 \\ 0.30 \\ -0.30 \\ 0.35 \\ -0.33 \\ 0.35 \end{pmatrix} \quad \begin{pmatrix} 0.31 \\ 0.12 \\ 0.31 \\ 0.51 \\ -0.20 \\ 0.49 \\ 0.34 \\ 0.12 \\ 0.31 \end{pmatrix} \quad \begin{pmatrix} 0.10 \\ 0.58 \\ 0.10 \\ -0.19 \\ 0.41 \\ -0.19 \\ 0.11 \\ 0.59 \\ 0.09 \end{pmatrix} \quad \begin{pmatrix} 0.46 \\ -0.00 \\ -0.43 \\ 0.30 \\ 0.03 \\ -0.28 \\ 0.41 \\ -0.02 \\ -0.49 \end{pmatrix} \quad \begin{pmatrix} -0.10 \\ -0.15 \\ -0.09 \\ 0.33 \\ 0.83 \\ 0.33 \\ -0.13 \\ -0.14 \\ -0.06 \end{pmatrix}$$

$$\begin{pmatrix} 0.27 \\ 0.11 \\ -0.11 \\ -0.62 \\ 0.00 \\ 0.63 \\ 0.11 \\ -0.12 \\ -0.27 \end{pmatrix} \quad \begin{pmatrix} -0.43 \\ -0.06 \\ -0.54 \\ -0.10 \\ 0.00 \\ 0.09 \\ 0.55 \\ 0.05 \\ 0.40 \end{pmatrix} \quad \begin{pmatrix} 0.08 \\ -0.69 \\ 0.02 \\ -0.09 \\ -0.00 \\ 0.10 \\ -0.02 \\ 0.69 \\ -0.09 \end{pmatrix} \quad \begin{pmatrix} -0.50 \\ 0.00 \\ 0.50 \\ 0.00 \\ 0.02 \\ 0.00 \\ 0.47 \\ -0.01 \\ -0.50 \end{pmatrix}$$

- Let's look at the filter with the highest variance.

We will see that this filter captures the underlying intensity pattern very well.

Basically, this vector consists of a value of about 1/3, at some places positive, at others negative.



- Convolving the image with this filter gives the biggest swings (i.e. variance !) as the pos. and neg. values go from an intensity top to intensity valley



- *Too many output images for this section, see lecture notes.*

# Traditional Object Recognition

## Lecture Notes

### Background

- Current state-of-the-art: Deep Learning
- Today: Traditional approaches
  - Pre-deep learning
  - Algorithmic basics
  - Important to understand the field
  - Important for various other applications

### Outline

- Specific object recognition  
Identifying the same object in different images despite variations in pose and illumination
- Object category recognition  
Identifying the object category despite variations in pose, illumination and intra-class variation

### Specific objects vs. category-level objects

- A specific object = an instance of an object class  
e.g. "my car" instead of "a car"

### Example application

- search photos on the web for particular places
- Large-Scale Retrieval [Philbin CVPR'07]
- Mobile tourist guide [Quack, Leibe, Van Gool, CIVR'08]

### Challenges

Main challenges:

- Pose / viewpoint changes
- Illumination variation
- Occlusion
- Clutter

## Determine a numerical representation

### Once upon a time...

- comparing image features with features of objects in a database, trying to figure out **type + pose**
- Representation: Local geometric features  
Largely contour based approach
- Slow: hypothesise-and-verify process

## Model-based approaches

### Next-step in the evolution: Invariant-based recognition of planar shapes

- The crucial advantage of invariants is that they decouple object type and pose issues
- Ex. given here for completely visible planar shapes
  - under affine distortions
  - using invariant signatures of the outlines

## Example

- $\int |\bar{x} - \bar{x}_1 \bar{x}^{(1)}| dt$  as a function of  $\int \text{abs}(|\bar{x}^{(1)} \bar{x}^{(2)}|^{1/3}) dt$

## Appearance based methods

- The model of an object is simply its image(s).
- A simple example: Template matching
  - Shift the template over the image and compare  
(e.g. Normalized Cross-Correlation or Sum of Squared Diff.)
  - The problem is variation in the appearance because of changes in viewpoint / lighting → Zillions of templates

## The power of Principal Component Anal.

- PCA represents data in a lower-dimensional space keeping most of the variance
- It was seen to be powerful for similar patterns like faces, that exhibit a lot of redundancy  
 $I = \mu + \sum_j^J \alpha_j U_j$        $I : [\alpha_1, \dots, \alpha_J]$
- Representation: PCA weights of the image – global image representations

## Appearance manifold approach [Nayar et al. '96]

- Training

for every object :

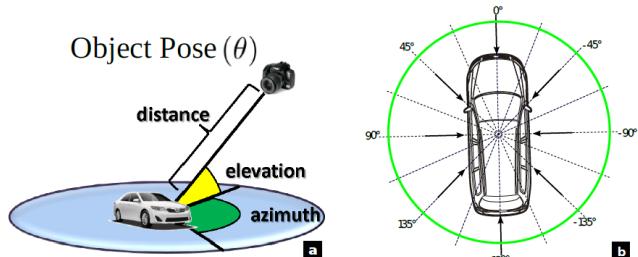
- sample the set of viewing conditions (mainly viewpoints in this ex.)
- use these images as feature vectors (after manual bounding-box fitting around the object, rescaling, brightness normalization)
- apply a PCA over all the images (directly on the images)
- keep the dominant PCs (10-20 enough already)
- sequence of views for 1 object represent a manifold in the space of projections (fit data pts with splines, then densely resample if desired)

- Recognition

what is the nearest manifold to a test image?

## Object-pose manifold

- The images were put on a turntable, and imaged from a fixed distance and under a fixed elevation angle
- hence pose changes were limited to changing azimuth angles



- Appearance changes projected on PCs:

1D pose changes in this case by spinning around vertical axis and only projection on 3 principal components shown

- Sufficient characterization for recognition and pose estimation
- Real-time system

## Eigenfaces for compact face representation

### A slight diversion: 3D reconstruction application

- Using the PCA representation identify the 3D face that yields the image when projected

## Comparison between model-based and appearance-based techniques

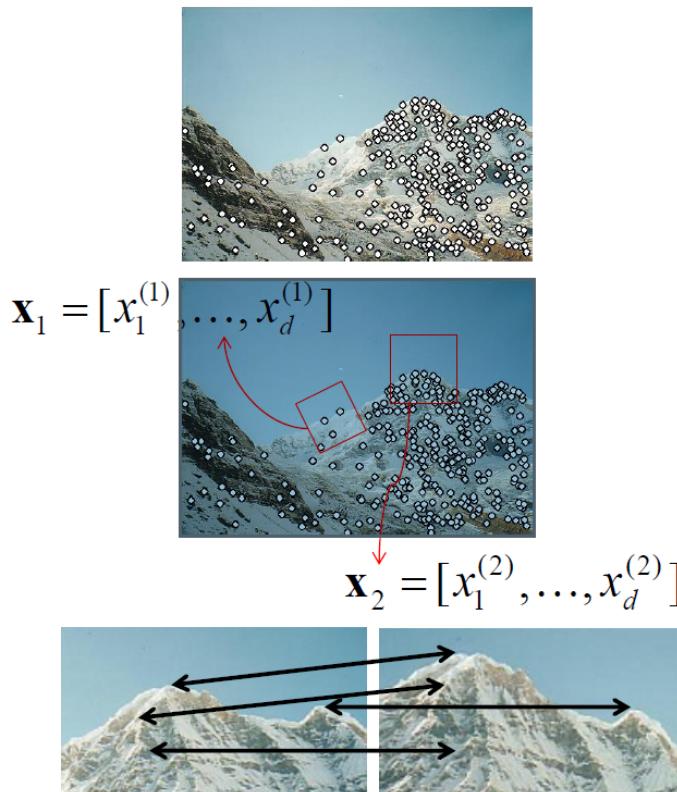
- Pure model-based

- Compact model
- Can deal with clutter
- *Slow analysis-by-synthesis*
- *Models difficult to produce*

- For limited object classes
- Pure appearance-based
  - Large models
  - Cannot deal with clutter
  - Efficient
  - Models easy to produce
  - For wide classes of objects

## Local features: main components

1. Detection: Identify interest points
2. Description: Extract feature vector descriptors around them
3. Matching: Determine correspondence between descriptors in two views



## Euclidean invariant feature [Schmid and Mohr '97]

- Training
  1. look for corners (with the Harris detector)
  2. take circular regions around these points, of different radii (cope a bit with scale changes)
  3. calculate, from the regions, features, invariant under planar rotation
  4. do this from different viewpoints, where the invariance cuts down the number of views needed (here no in-plane rotations necessary)
- Testing
  1. compare these features with those found for images in the database -> find matches

2. look for consistent placement of candidate matches / geometric constraints (epipolar geometry)
  3. decide which object based on the number of remaining matches (best matching image: type+appr.pose)
- Example (rotation) invariant:  $G_x G_x + G_y G_y$   
where  $G_x$  and  $G_y$  represent horizontal and vertical derivatives of intensity weighted by a Gaussian profile ('Gaussian derivatives')
  - 2nd example invariant:  $G_{xx} + G_{yy}$   
Where  $G_{xx}$  and  $G_{yy}$  represent 2nd order Gaussian derivatives  
(Compute features for circles at different scales, i.e. take scale into account explicitly)

## Example

- Training examples for one object in the database  
deal with cluttered background  
need less training images  
*problems with uniform objects*

## Hybrid techniques

- Rather compact model
- Can deal with clutter and partial occlusion
- Efficient
- Models easy to produce  
(take images, fewer than in pure appearance-based method)
- For rather wide class of objects  
(almost as wide as in pure appearance based, but there is a problem with untextured objects)
- A lot of room for improving the invariance properties

## Supporting the matching step

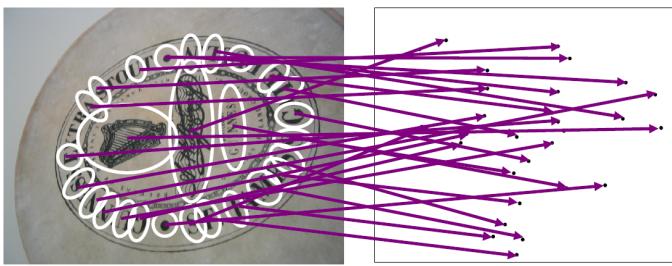
1. Too slow if naively done  
→ Hierarchical vocabulary tree for speed-up
2. Will often fail when only based on descriptor matching

## Indexing local features

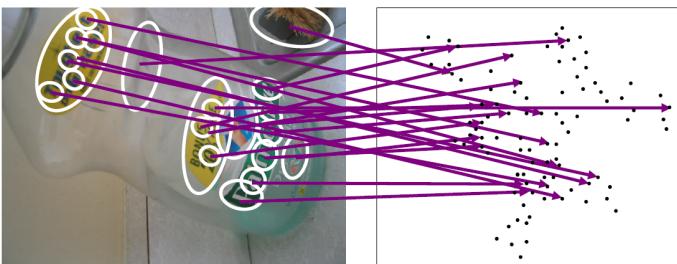
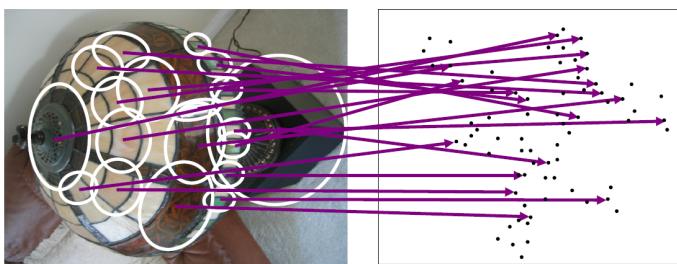
- With potentially thousands of interest pts + their descriptors per image, and hundreds to millions of images to search, how to efficiently find those relevant to a new test image?
- Quantize/cluster the descriptors into 'visual words'  
And match words hierarchically: vocabulary tree Use Inverted file indexing schemes

## Visual words: main idea

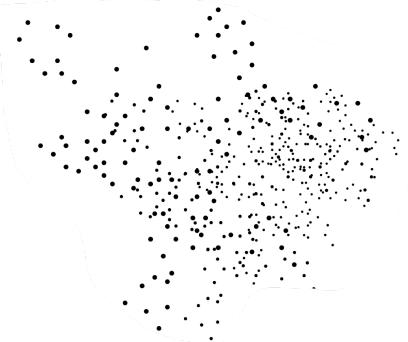
- Extract some local features from several images



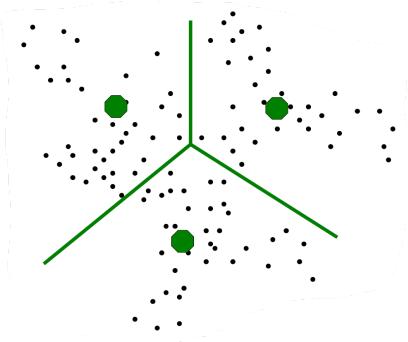
e.g., SIFT descriptor space:  
each point is 128-dimensional



- Each point is a local descriptor, e.g. SIFT vector.



- Vector quantize feature space = cluster the features

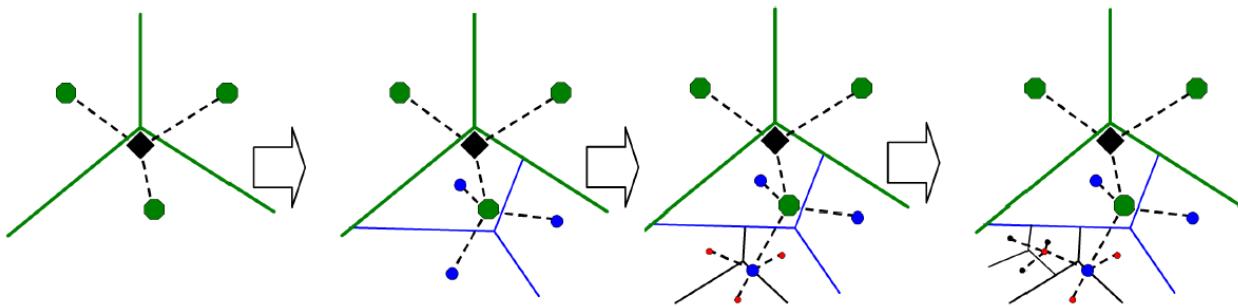


## K-means clustering

1. randomly initialize K cluster centers
2. assign each feature to nearest cluster center
3. recompute cluster center (mean)
4. iterate from 2, until convergence

## Hierarchical K-means clustering

- Allows to use larger vocabularies and thereby yields better results
- In the example k=3, but typically it is chosen higher, e.g. k=10 and 6 layers could be used for search in about 1M images



## Visual words

Ex: each group of patches belongs to same visual word

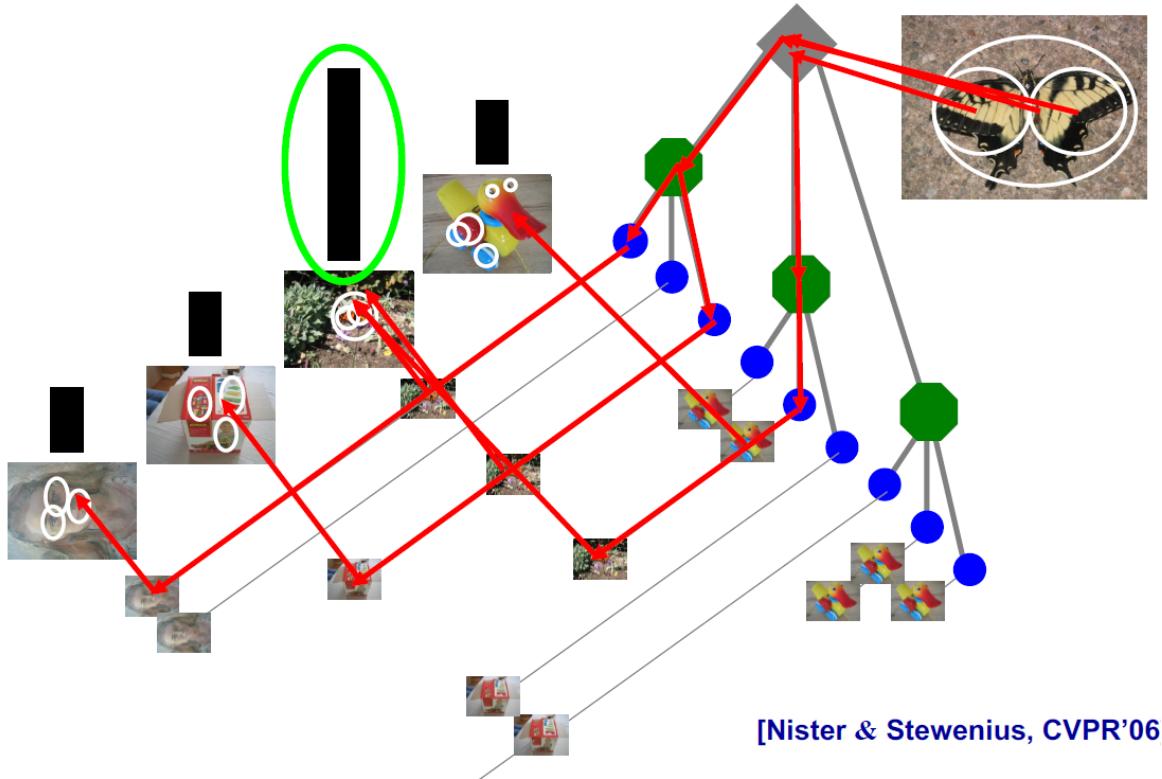
## Indexing local features: inverted file index

- For text documents, an efficient way to find all **pages** on which a **word** occurs is to use an index...
- We want to find all **images** in which a **visual word** occurs.

## Inverted file index

- Database images are loaded into the index, mapping words to image numbers
- New query image is mapped to indices of database images that share a word.

## Retrieval with vocabulary tree + inverted file index



## Performance

- Evaluated on large databases Indexing with up to 1M images
- Online recognition for database of 50,000 CD covers  
Retrieval in ~1s
- Best with very large visual vocabularies
- NOTE: object class recognition typically done with smaller vocabularies

## Supporting the matching step

- Ex. cleaning matches based on RANSAC-Ep.G geom.

### RANSAC - intermezzo

- Matching can start from interest points and their descriptors, but such matching is rather fragile.
- Typically, several 'matches' are wrong, so-called outliers, and one needs to add a test on the configuration of the matches in order to remove the outliers and keep the correct inliers.
- Epipolar geometry and projective matching are often used tests, using RANSAC to withstand unavoidable mismatches.
- **RANdom SAmple Consensus**

## RANSAC

- The RANSAC test on **epipolar geometry** assumes that there is a fundamental matrix that matches agree with, and

- The RANSAC test on **projectivities** assumes that there is a projectivity that maps points in the first image onto the matching points in the second
- Such tests allow for the elimination of many outliers
- but these tests make strong assumptions about the scene:
  - Epipolar geometry: rigidity of the scene (i.e. objects in the scene do not move with respect to each other)
  - Projectivity: the scene is not only rigid, but also (largely) planar  
Nonetheless such tests help !
- RANSAC assumes the data consists of "inliers", i.e. correct matches, and "outliers", i.e. incorrect matches
- From a set of match candidates,
  1. randomly select the minimal nmb of matches to formulate an initial test hypothesis (e.g. 7 for epipolar geometry or 4 for a projectivity; this nmb better be small since the selected tuple must not contain any outlier match for it to work)
  2. check how consistent other matches are with this hypothesis, i.e. in how far it is supported
  3. use all supporting matches to refine the hypothesis and discard the rest

Finally, RANSAC selects the hypothesis with maximal support after a fixed number of trials or after sufficient support was reached
- How often should we draw?... Suppose  
 $n$  - minimum number of data required to fit the model  
 $k$  - nmb of iterations / trials performed by the algorithm  
 $t$  – threshold to determine when a match fits a model  
 $d$  - nmb of 'inliers' needed for a model to be OK  
 $t$  and  $d$  are typically chosen beforehand. The nmb of iterations  $k$  can then be calculated. Let  $p$  be the probability that RANSAC only selects inliers for the  $n$  data units generating a valid test at least once, i.e. the probability that the algorithm gets a good output. When **w is the proportion of inliers (estimated)**,  
 $1 - p = (1 - w^n)^k$  is the probability that NO good hypothesis is selected

## Object Categorization

### Challenges: Robustness

- illumination
- object pose
- clutter
- occlusions
- intra-class variation
- viewpoint

On top of factors affecting specific object recognition, the added complexity of intra-class variation

## Intra-class and inter-class variation

The difference between classes can be as small as that between instances of the same class ... yet the distinction needs to be made

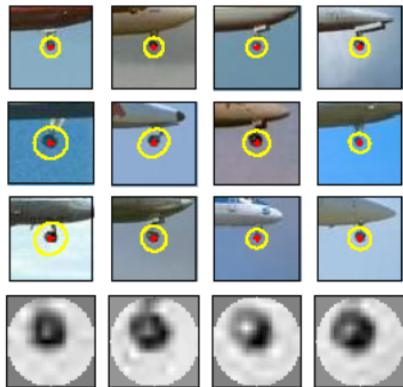
## Two main tasks: Classification and Detection

- classification: is there a car in this image ? A binary answer is enough
- detection: where is the car ? Need localization

## Building blocks: Local features and Visual Words

### Why visual words?

- Useful for describing object appearance:
- Appearances of objects vary a lot within a category
- But appearance of local parts varies less!
- Ideally: a part = a visual word, but that is not always so



Sivic & Zisserman 2003; Csurka, Bray, Dance, & Fan 2004; many others.

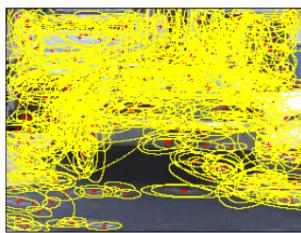
## Building Visual Vocabularies

- Components:
  - Sampling strategy: extract features densely, or at interest points only, randomly, ... ?
  - Clustering / quantization algorithm
- Questions / Issues:
  - Unsupervised vs. supervised
  - What corpus provides features (universal vocabulary)?
  - Vocabulary size, number of words

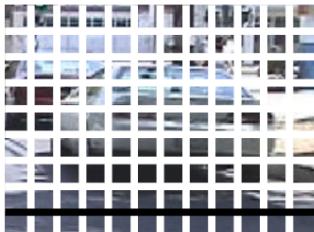
## Sampling Strategies

- To find specific, textured objects, sparse sampling from interest points often more reliable.
- Multiple complementary interest operators offer more image coverage.
- For object categorization, dense sampling offers better coverage.

- [See Nowak, Jurie & Triggs, ECCV 2006]



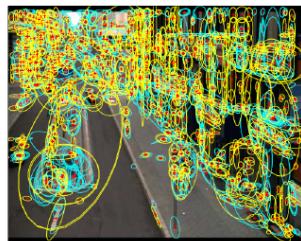
Sparse, at  
interest points



Dense, uniformly



Randomly



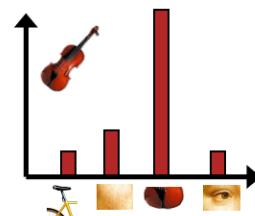
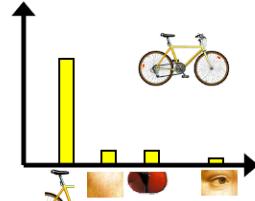
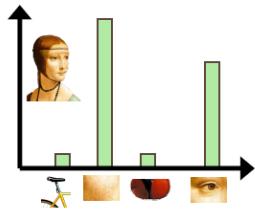
Multiple interest  
operators

## Clustering / Quantization Methods

- Typically specific object recognition works better with larger vocabularies, object class recognition with smaller ones (words cover more variability in that case)
- k-means (typical choice), agglomerative clustering, meanshift,...
- Hierarchical clustering: allows faster insertion / word assignment while still allowing large vocabularies
- Vocabulary tree [Nister & Stewenius, CVPR 2006]

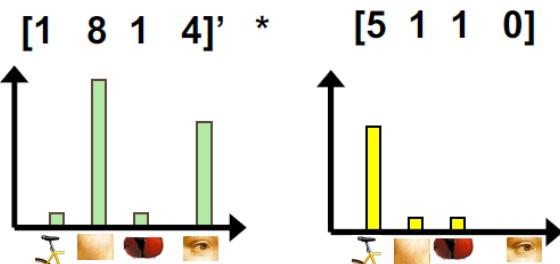
## Bags of Visual Words

- Summarize entire image based on its distribution (histogram) of word occurrences.
- Analogous to bag of words representation commonly used for documents.
- Advantage of histogram is that irrespective of the nmb of local features, the size is always the same



## Comparing Bags of Words

- Any histogram comparison measure can be used here.  
E.g. normalized scalar product between histograms  
(or with more sophisticated classifiers mentioned shortly)


 $\vec{d}_j$ 

(histograms in database)

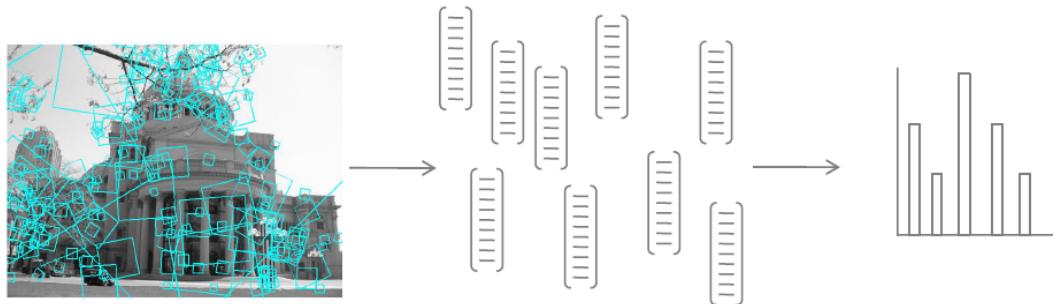

 $\vec{q}$ 

(histogram query image)

$$sim(\vec{d}_j, \vec{q}) = \frac{\vec{d}_j \cdot \vec{q}}{|\vec{d}_j| |\vec{q}|}$$

## Learning/Recognition with BoW Histograms

- Bag of words enable to describe the unordered feature set with a single vector of fixed dimensionality

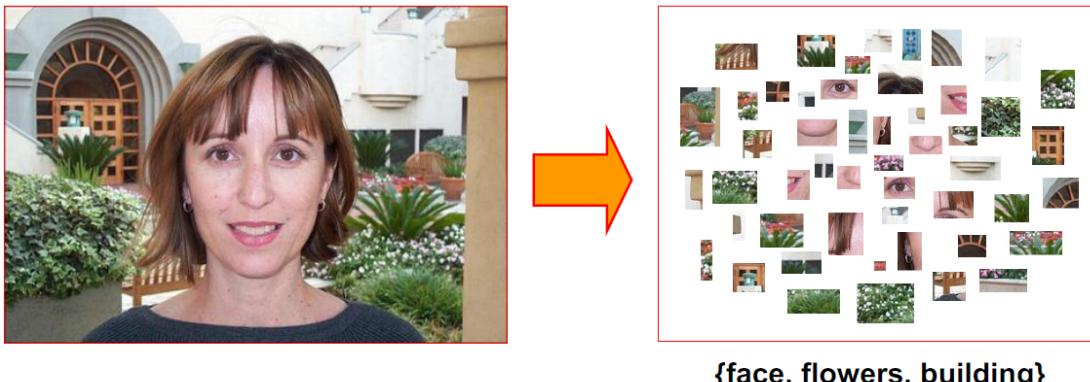


- Provides easy way to use distribution of feature types with various learning algorithms requiring vector input.

## BoW for Object Categorization

- Works pretty well for whole-image classification

Csurka et al. (2004), Willamowski et al. (2005), Grauman & Darrell (2005), Sivic et al. (2003, 2005)



## Image Classification (more sophisticated classifiers)

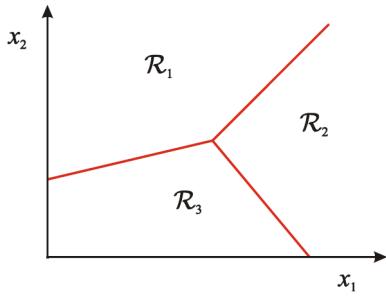
- Given bag-of-words of images from different classes, how do we learn a model for distinguishing them?

## Discriminative Methods

- Learn a decision rule (classifier) assigning bag-of-words representations of images to different classes

## Classification

- Assign input vector to one of two or more classes
- Any decision rule divides input space into decision regions separated by decision boundaries



## Classifier Choices

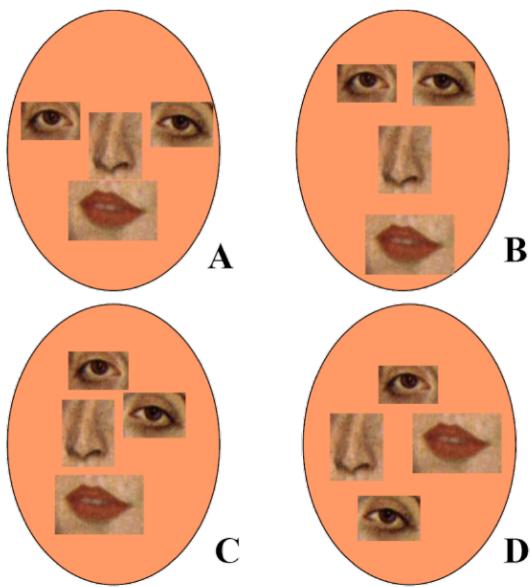
- Nearest neighbor  
Shakhnarovich, Viola, Darrell 2003 Berg, Berg, Malik 2005...
- Neural networks  
LeCun, Bottou, Bengio, Haffner 1998 Rowley, Baluja, Kanade 1998 ...
- Support Vector Machines  
Guyon, Vapnik Heisele, Serre, Poggio, 2001,...
- Boosting  
Viola, Jones 2001, Torralba et al. 2004, Opelt et al. 2006,...

## BoW for Image Classification

- Good performance for classification (object present/absent)
- Better than some more elaborate part-based models with spatial constraints...
- What could be possible reasons why?

## Properties of BoW Representations

- The bag of words removes spatial layout.
- This is both a strength and a weakness.
- Why a strength? A => B
- Why a weakness? A => C,D



## BoW Representation: Spatial Information

- A bag of words is an orderless representation: discarding spatial relationships between features
- Options to bring in some spatial info:
  - Visual “phrases” : frequently co-occurring words
  - Semi-local features : describe configuration, neighborhood
  - Let position be part of each feature descriptor
  - **Count bags of words only within sub-grids of an image**

## Spatial Pyramid Representation

Representation in-between orderless BoW and structural description



## Spatial Pyramid description matching

- Can capture scene categories well → texture-like patterns  
but with some variability in the positions of all the local pieces.

- Sensitive to global shifts of the view

## **Discussion: Bag-of-Words**

- Pros:
  - Flexible to geometry / deformations / viewpoint
  - Compact summary of image content
  - Provides vector representation for sets
  - Empirically good recognition results in practice
- Cons:
  - Basic model ignores geometry – can verify afterwards, or
  - embed within feature descriptors
  - Background and foreground mixed when bag covers whole image
  - Interest points or sampling: no guarantee to capture object-level parts
  - Optimal vocabulary formation remains unclear

## **Detection: Sliding-window approaches**

### **Detection via Classification: Main Idea**

- Basic component: a binary classifier  
If object may be in a cluttered scene, slide a window around looking for it.  
a brute-force approach with many local decisions.
- Fleshing out this pipeline a bit more, we need to:
  1. Obtain training data
  2. Define features
  3. Train a classifier

## **Feature extraction: Global Appearance**

- Simple holistic descriptions of image content
  - Grayscale / color histogram
  - Vector of pixel intensities
- Pixel-based representations sensitive to small shifts
- Color or grayscale-based appearance description can be sensitive to illumination and intra-class appearance variation

## **Gradient-based Representations**

- Consider edges and (oriented) intensity gradients
- Summarize local distribution of gradients with histogram
  - Locally orderless: offers invariance to small shifts and rotations
  - Contrast-normalization: try to correct for variable illumination

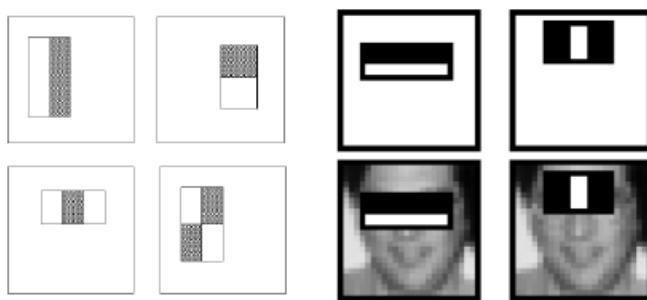
- One very popular example: Histogram of Oriented Gradients  
[Dalal & Triggs, CVPR 2005]

## **HoG pedestrian detector**

- Finally, a decision needs to be made for each window...

## **Sliding Windows: Rectangular Integral Image Filters**

- Feature output is difference between adjacent regions



- Efficiently computable with integral image: any sum can be computed in constant time  
Value at  $(x, y)$  is sum of pixels above and to the left of  $(x, y)$
- Avoid scaling images → scale features directly for same cost

## **Large Library of Filters**

- Considering all possible filter parameters: position, scale, and type:  
180,000+ possible features associated with each  $24 \times 24$  window
- Obviously in a classifier you would need to select the informative ones and to form the classifier.  
Not all of them will be informative!

## **Classifier Construction**

- How to compute a decision for each subwindow?

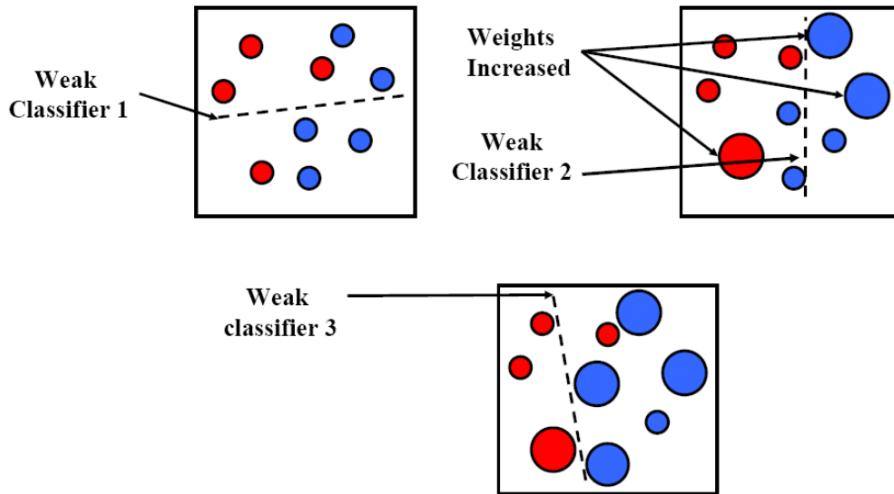
## **Sliding Windows: AdaBoost**

### **Boosting**

- Build a strong classifier by combining many “weak classifiers”, which need only be better than chance
- Sequential learning process: at each iteration, add a weak classifier
- Flexible to choice of weak learner
  - including fast simple classifiers that alone may be inaccurate
- We’ll look at Freund & Schapire’s AdaBoost algorithm
  - Easy to implement
  - Base learning algorithm for Viola-Jones face detector

## AdaBoost: Intuition

- Consider a 2D feature space with **positive and negative** examples.
- Each weak classifier splits the training examples with at least 50% accuracy.
- Examples misclassified by a previous weak learner are given more emphasis at future rounds.



- Final classifier is combination of the weak classifiers

- Given example images  $(x_1, y_1), \dots, (x_n, y_n)$  where  $y_i = 0, 1$  for negative and positive examples respectively.
- Initialize weights  $w_{1,i} = \frac{1}{2m}, \frac{1}{2l}$  for  $y_i = 0, 1$  respectively, where  $m$  and  $l$  are the number of negatives and positives respectively.
- For  $t = 1, \dots, T$ :

- Normalize the weights,

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^n w_{t,j}}$$

so that  $w_t$  is a probability distribution.

- For each feature,  $j$ , train a classifier  $h_j$  which is restricted to using a single feature. The error is evaluated with respect to  $w_t$ ,  $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$ .
- Choose the classifier,  $h_t$ , with the lowest error  $\epsilon_t$ .
- Update the weights:

$$w_{t+1,i} = w_{t,i} \beta_t^{1-\epsilon_t}$$

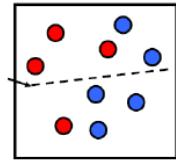
where  $\epsilon_t = 0$  if example  $x_i$  is classified correctly,  $\epsilon_t = 1$  otherwise, and  $\beta_t = \frac{1}{\epsilon_t}$ .

- The final strong classifier is:

$$h(x) = \begin{cases} 1 & \sum_{t=1}^T \alpha_t h_t(x) \geq \frac{1}{2} \sum_{t=1}^T \alpha_t \\ 0 & \text{otherwise} \end{cases}$$

where  $\alpha_t = \log \frac{1}{\beta_t}$

**Start with uniform weights on training examples**



$\{x_1, \dots, x_n\}$

**For T rounds**

**Evaluate weighted error for each feature, pick best.**

**Re-weight the examples: incorrectly classified  $\Rightarrow$  more weight  
Correctly classified  $\Rightarrow$  less weight**

**Final classifier is combination of the weak ones, weighted according to the error they had.**

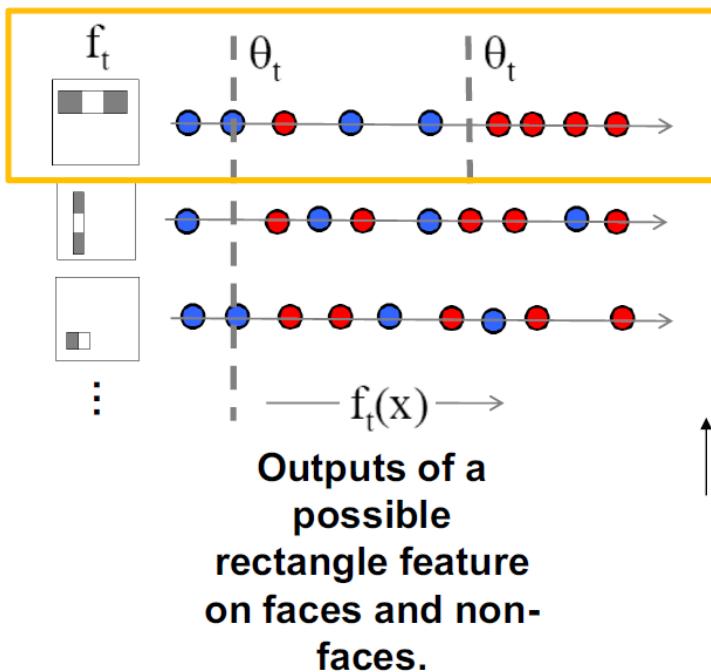
[Freund & Schapire 1995]

## Example: Face Detection

- For frontal faces global appearance models + a sliding window detection approach fit well:
    - Regular 2D structure
    - Center of face almost shaped like a “patch”/window
  - Now we'll take AdaBoost and see how the Viola-Jones face detector works
  - Large Library of Filters
- Use AdaBoost both to select the informative features and to form the classifier

## AdaBoost for Feature+Classifier Selection

- Want to select the single rectangle feature and threshold that best separates **positive** (faces) and (non-faces) training examples, in terms of weighted error.



Slide credit: Kristen Grauman

**Resulting weak classifier:**

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

For next round, reweight the examples according to errors, choose another filter/threshold combo.

[Viola & Jones, CVPR 2001]

## AdaBoost for Efficient Feature Selection

- Image features = weak classifiers
- For each round of boosting:
  - Evaluate each rectangle filter on each example
  - Sort examples by filter values
  - Select best threshold for each filter (min error)
  - Sorted list can be quickly scanned for the optimal threshold
  - Select best filter/threshold combination
  - Weight on this feature is a simple function of error rate
  - Reweight examples

P. Viola, M. Jones, Robust Real-Time Face Detection, IJCV, Vol. 57(2), 2004. (first version appeared at CVPR 2001)

## Efficiency Considerations

- Even if the filters are fast to compute, each new image has a lot of possible windows to search.
- How to make the detection more efficient?
  - cascade of classifiers of increasing complexity
  - conservative classifiers: try to keep in all positives, letting passing some false positives to the next stages as price to pay
  - later classifiers are more expensive (more complex) having to check the fewer surviving patterns
  - this cascaded scheme isn't discussed in detail

## Viola-Jones Face Detector: Summary

- Train with 5K positives, 350M negatives
- Real-time detector using 38 layer cascade
- 6061 features in final layer
- [Implementation available in OpenCV: [website](#)

## Viola-Jones Face Detector: Results

- Speed
  - 384 by 288 pixel images detected at 15 fps on a conventional 700 MHz Intel Pentium III in 2001
  - Training time was weeks

## Discussion: Sliding-Windows

- Pros
  - Simple detection protocol to implement
  - Good feature choices critical, but part of the process
  - Past successes for certain classes
  - Good detectors available (Viola&Jones, HOG, etc.)
- Cons/Limitations
  - High computational complexity
    - More than 100,000 locations in an images no exception
    - This puts tight constraints on the classifiers we can use.
    - If training binary detectors independently for different classes, this means cost increases linearly with number of classes.
    - One can try to focus on promising locations only, e.g. by first running a generic object detector ('objectness scores')
  - With so many windows, false positive rate better be low
  - Typically need fully supervised training data (= bounding-boxes)

## Limitations (continued)

- Not all objects are “box” shaped
- Non-rigid, deformable objects not captured well with representations assuming a fixed 2D structure and/or must assume fixed viewpoint
- Objects with less-regular textures not captured well with holistic appearance-based descriptions
- When considering windows in isolation, context is lost
- In practice, often entails large training set with manually cropped windows (expensive)

- Requiring good match to a global appearance description can lead to sensitivity to partial occlusions

## Generalized Hough Transform: Implicit Shape Model (ISM)

### Implicit Shape Model (ISM)

- Basic ideas
  - Learn an appearance codebook
  - Learn a star-topology structural model
    - Features are considered independent given obj. center
- Algorithm: probabilistic Generalized Hough Transform

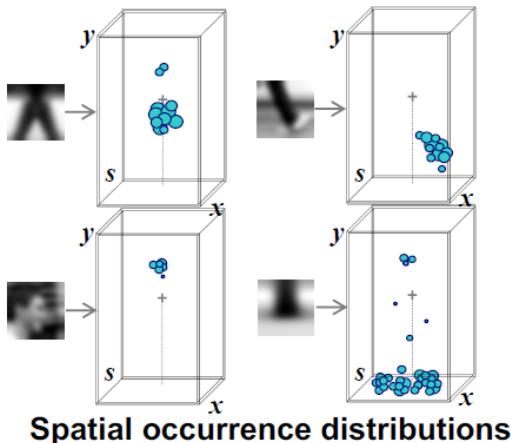
### Implicit Shape Model: Basic Idea

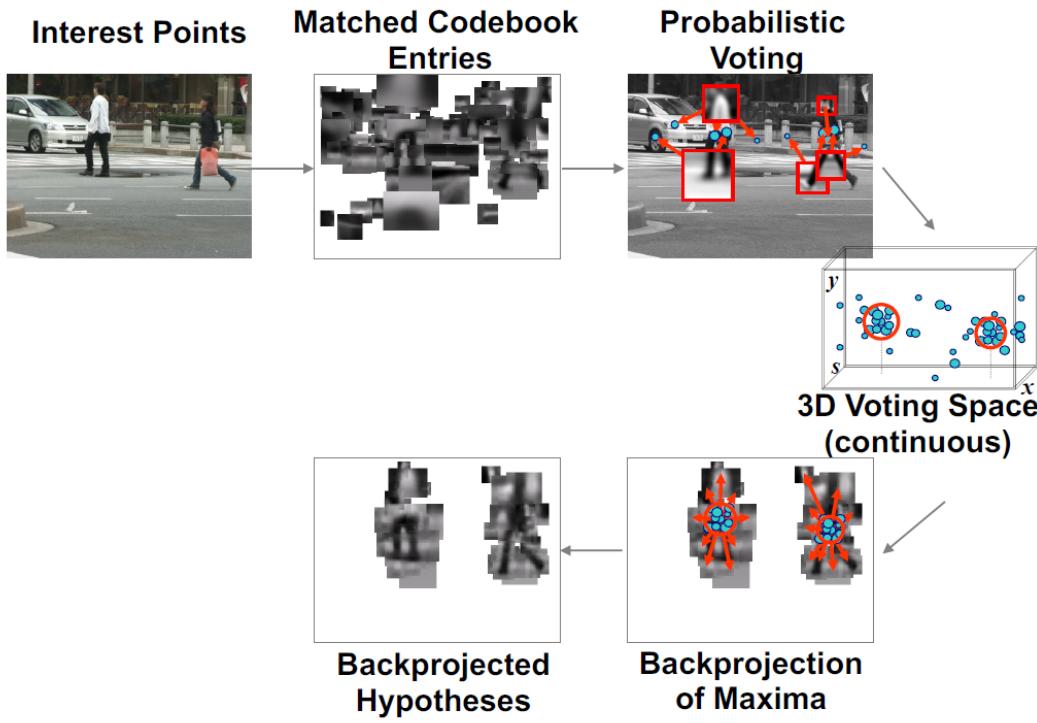
- Visual vocabulary is used to index votes for object position [a visual word = a "part"].
- Objects are detected as consistent configurations of the observed parts (visual words).

B. Leibe, A. Leonardis, and B. Schiele, Robust Object Detection with Interleaved Categorization and Segmentation, International Journal of Computer Vision, Vol. 77(1-3), 2008.

### Implicit Shape Model - Representation

- Learn appearance codebook
  - Extract local features at interest points
  - Agglomerative clustering → codebook
- Learn spatial distributions
  - Match codebook to training images
  - Record matching positions on object





- [Linux binaries available](#)

Including datasets & several pre-trained detectors

## Discussion: Implicit Shape Model

- Pros:
  - Works well for many different object categories
    - Both rigid and articulated objects
  - Flexible geometric model
    - Can recombine parts seen on different training examples
  - Learning from relatively few (50-100) training examples
  - Optimized for detection, good localization properties
- Cons:
  - Needs supervised training data
    - Object bounding boxes for detection
    - Segmentations for top-down segmentation
  - Only weak geometric constraints
    - Result segmentations may contain superfluous body parts.
  - Purely representative model
    - No discriminative learning