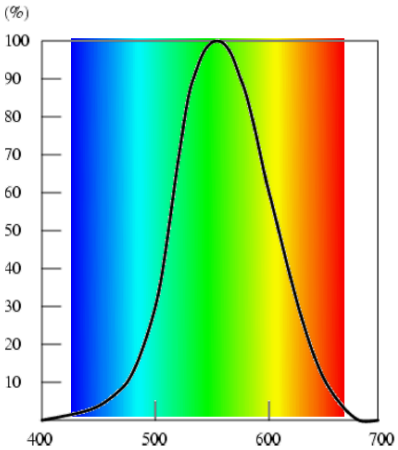
Surface Features

Lecture Notes

Colour

The perception of brightness

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



Link radiometry-photometry (Watt to lumen)

- **Photometry**: subjective impressions
  Radiometry: objective, physical measurements
  at 555 nm : 1lm = 1/683 W = 1.46 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

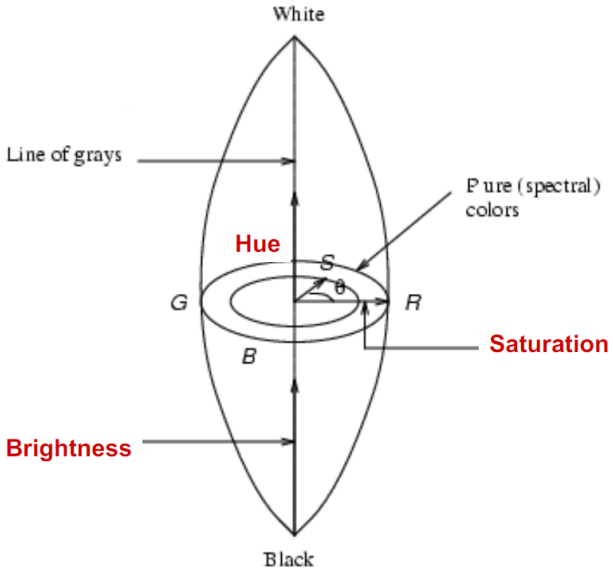
The study of colour…

Use:

- pleasing to the eye (visualisation of results)
- characterising colours (features e.g. for recognition)
- generating colours (displays, light for inspection)
- understanding human vision

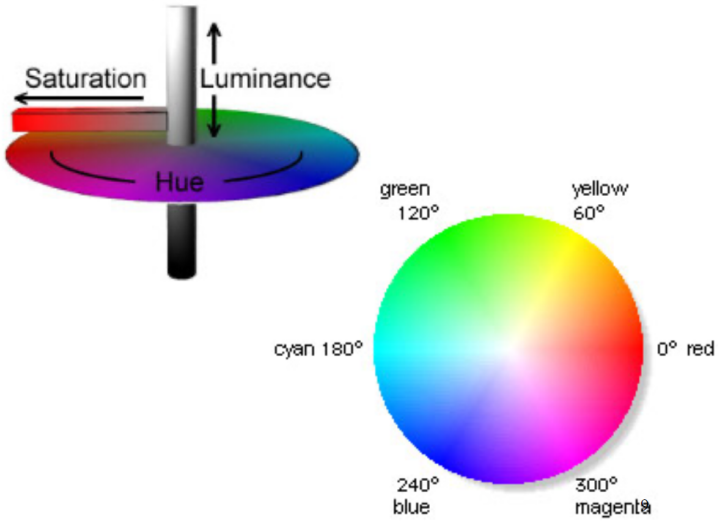
The perceptual attributes of light (humans)

1. brightness
2. hue
3. saturation



The C.I.E. color space

Human perception of light = 3-dimensional

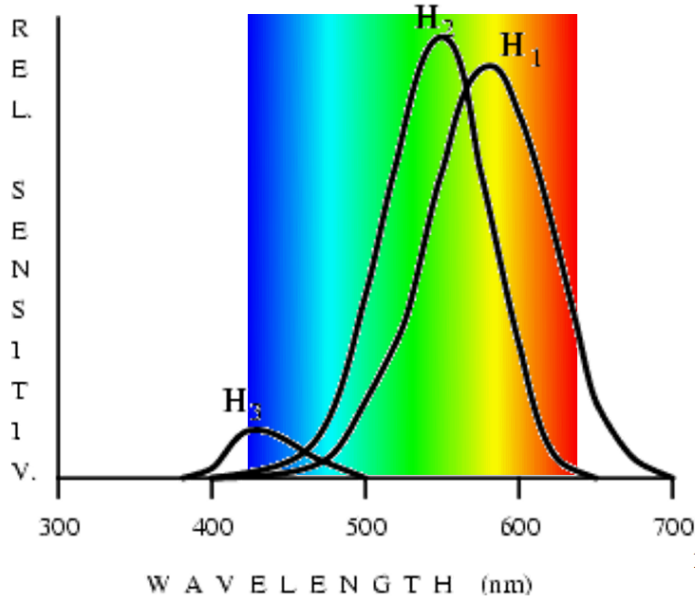


The history of colour

- Newton $\rightarrow$ spectrum
  Young $\rightarrow$ 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 λ is projected onto only 3 numbers R_i.

- 2 sources with equal R_i's ⇒ 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 $P_j(\lambda), \quad j = 1, 2, 3$
- 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 $L'm' = R \Rightarrow m' = L'^{-1} Lm$

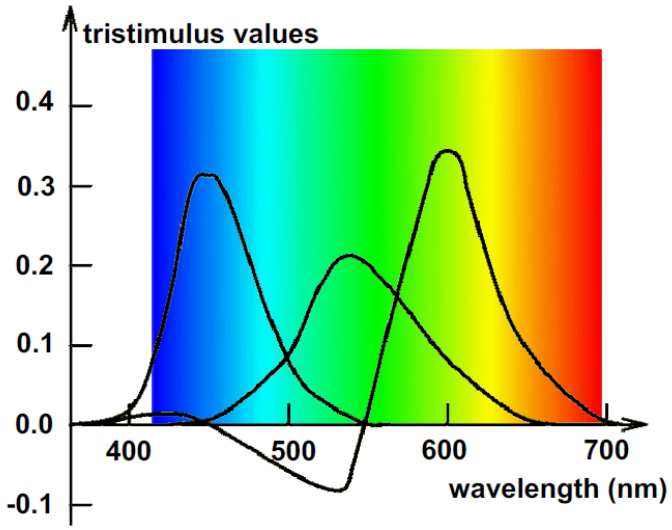
Tristimulus values

- "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

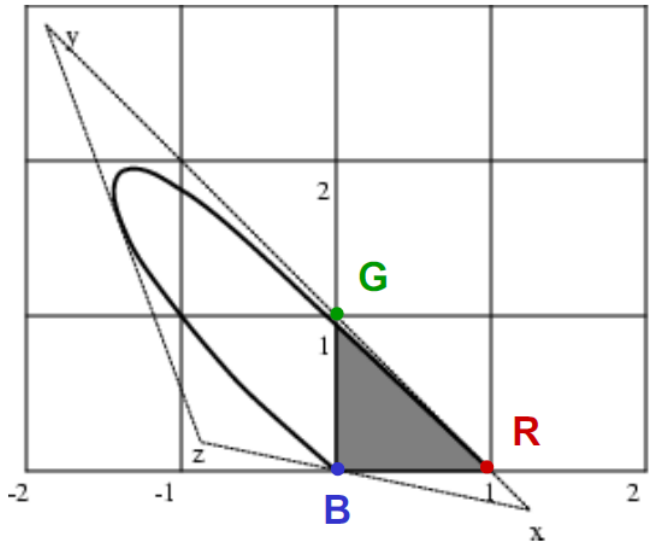
- negative values: colours that cannot be produced
- In such case: $mix(target, neg.primary) = mix(pos.primaries)$
- for **any** primary triple **some colours cannot be produced**
- for arbitrary source $C(\lambda)$
  $T_j(C) = \int C(\lambda)T_j(\lambda)d\lambda$

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

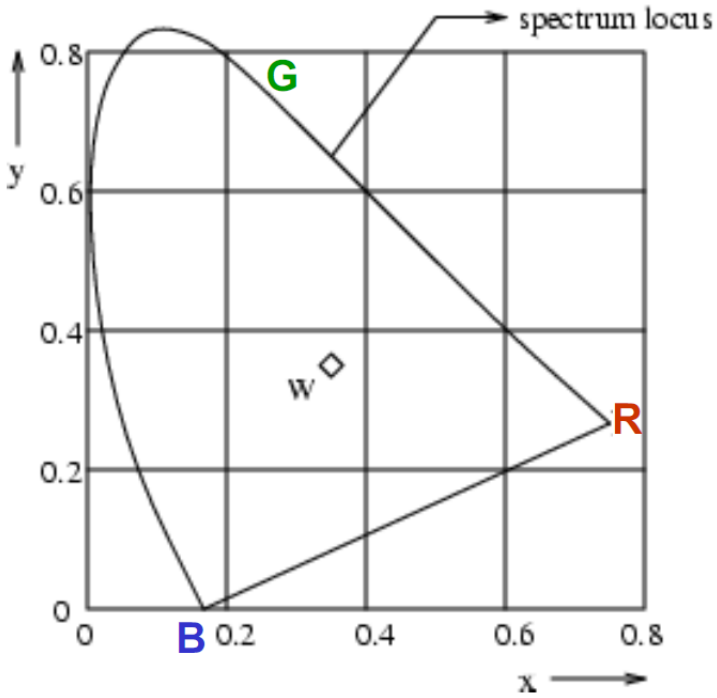
- chromaticity coordinates (r, g) for CIE primaries:
  $r = \frac{R}{R+G+B} \qquad g = \frac{G}{R+G+B}$
- The corresponding colour space:

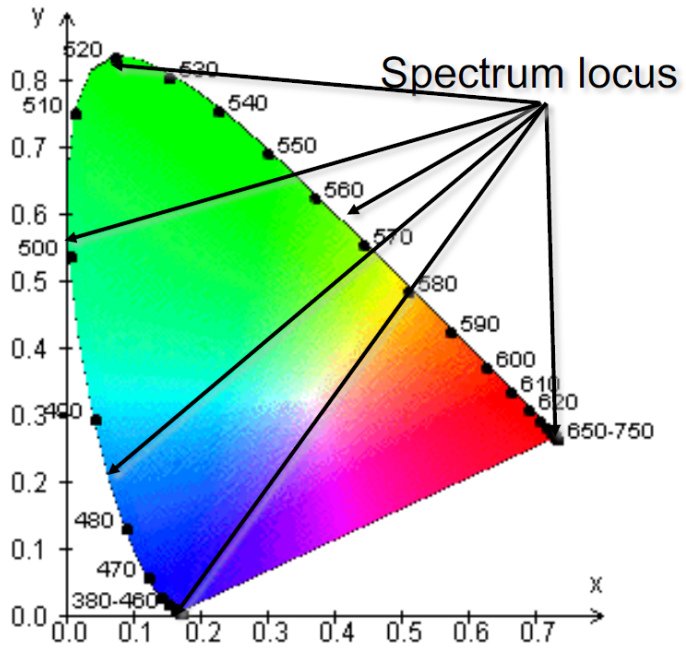


CIE x-y coordinates

- 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





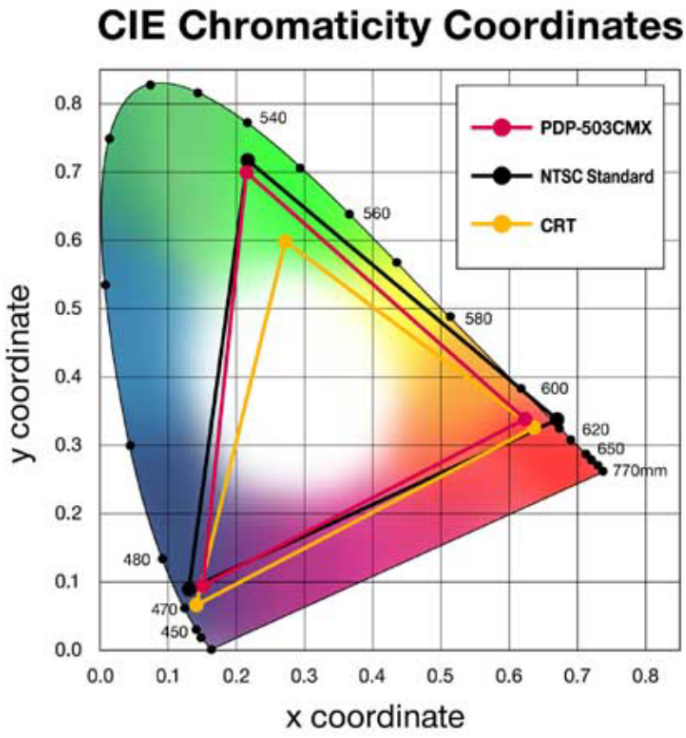
TV primaries

the EBU primaries have coordinates

$$R_r: \quad x = 0.64 \quad y = 0.33$$
$$G_r: \quad x = 0.29 \quad y = 0.60$$
$$B_r: \quad x = 0.15 \quad y = 0.06$$

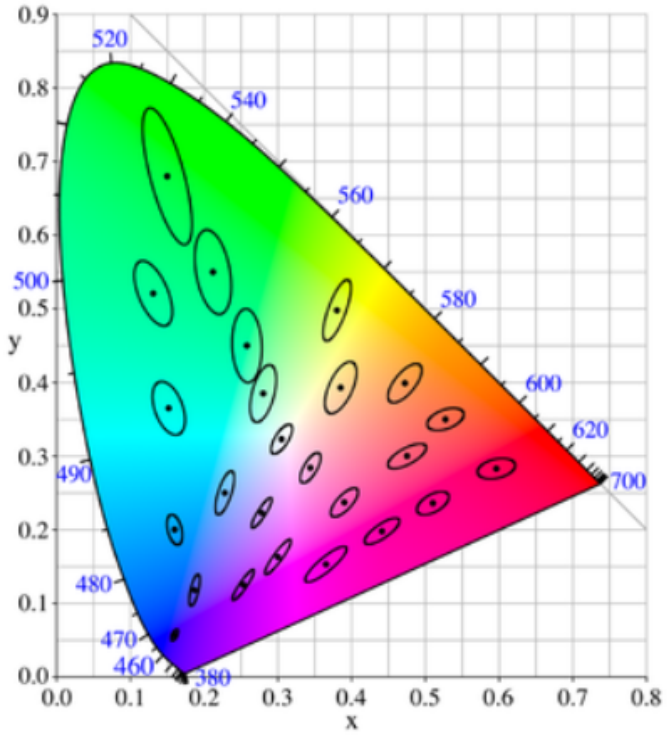
the NTSC primaries have coordinates

$$R_N: \quad x = 0.67 \quad y = 0.33$$
$$G_N: \quad x = 0.21 \quad y = 0.71$$
$$B_N: \quad x = 0.14 \quad y = 0.08$$



Notes

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



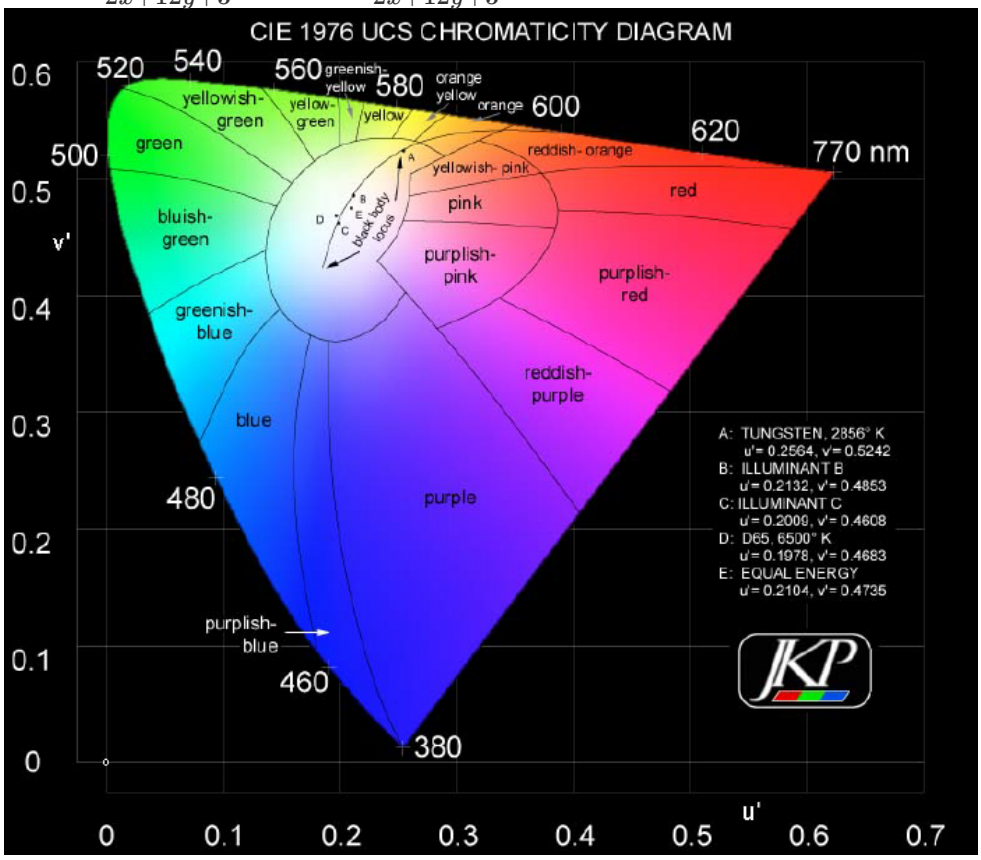
- pure spectrum colours (on the spectrum locus) are rare in nature

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



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

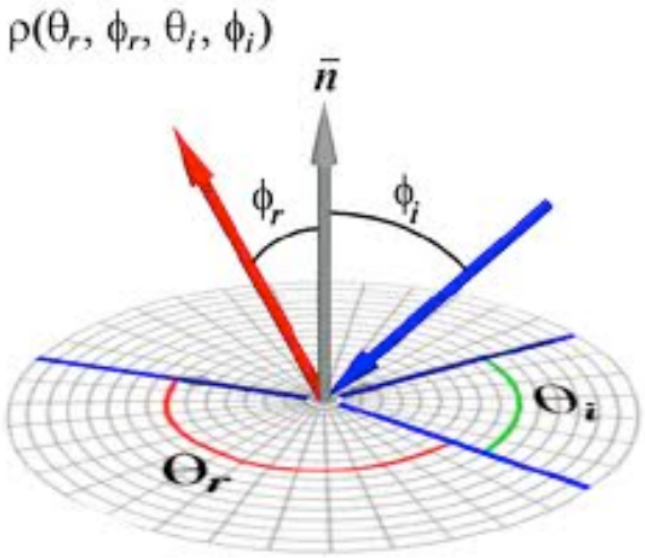
Illumination invariant colour features

- 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: $\theta_1 \leq arctan(\frac{u}{v}) < \theta_2$
- The Fourier Transform collects information globally over the entire image
  Not good for segmentation or inspection

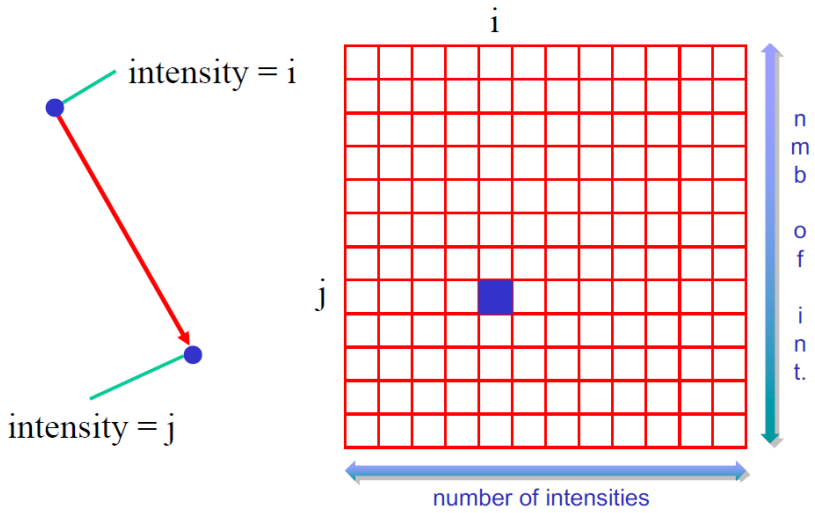
Histograms: principle

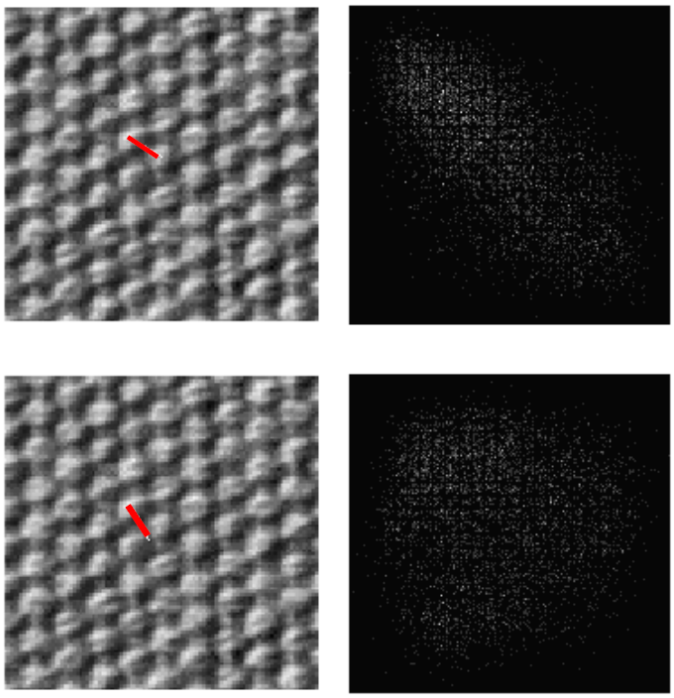
- Intensity probability distribution

- 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	$\sum_i \sum_j C(i,j) / (1+\lvert i-j \rvert)$
max. probability	$\max_{i,j} C(i,j)$

Laws filters

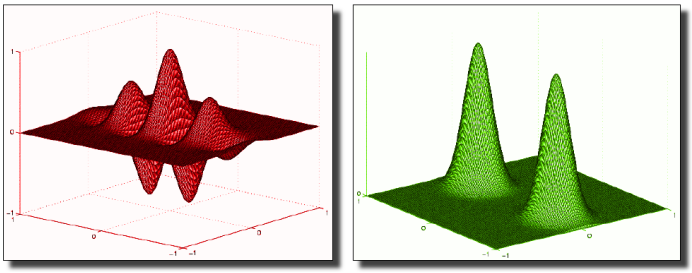
- This fixed filter set yields simple convolutions but has proven very effective in some cases

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
  $g(x,y) = exp(-\frac{x^2+y^2}{4\Delta_{x,y}^2})cos(2\pi u^* x + \varphi)$
- The filter's Fourier power spectrum
  $G(u,v) = \frac{1}{4\pi\Delta_{u,v}^2}(exp[-\frac{(u-u^*)^2+v^2}{4\Delta_{u,v}^2}] + exp[-\frac{(u+u^*)^2+v^2}{4\Delta_{u,v}^2}])$
- Good localisation in both domains:

Spatial domain Frequency domain

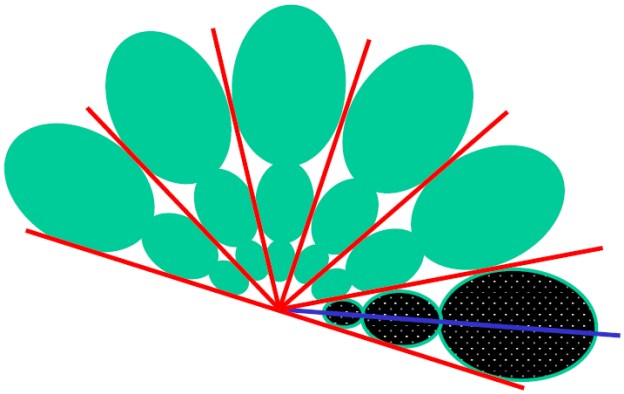


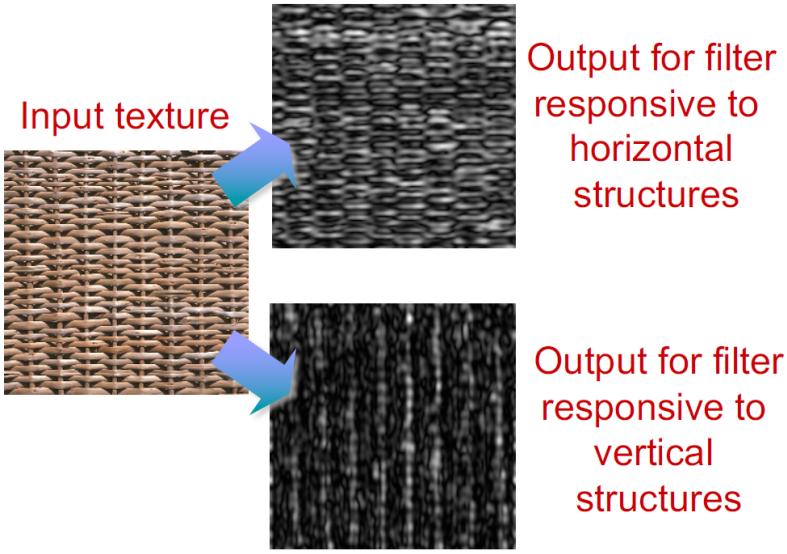
$$f = f(x,y) \Leftrightarrow F = F(u,v)$$

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

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

  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



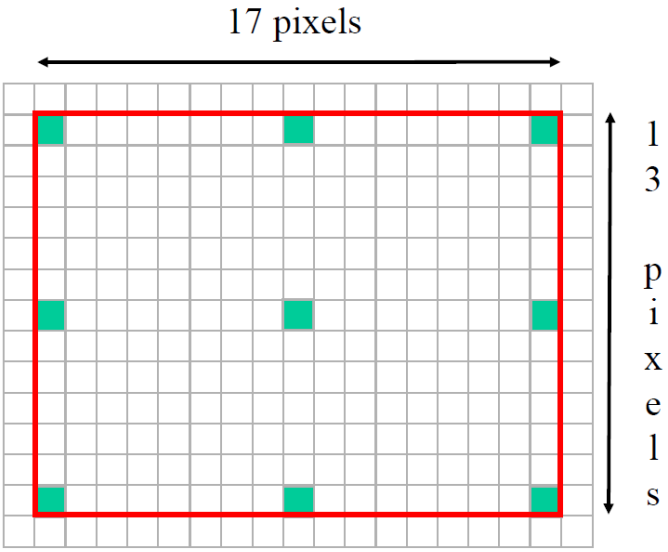


Eigenfilters (Ade, ETH)

- Filters adapted to the texture
  1. shift mask over training image
  2. collect intensity statistics
  3. PCA $\rightarrow$ eigenvectors $\rightarrow$ '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)



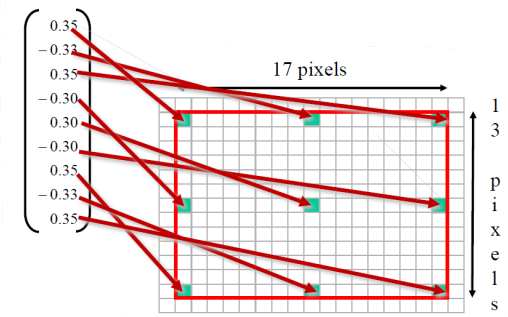
- Covariance matrix needed for PCA

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

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

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

- 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 $\int abs(|\bar{x}^{(1)} \bar{x}^{(2)}|^{1/3}) dt$

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.

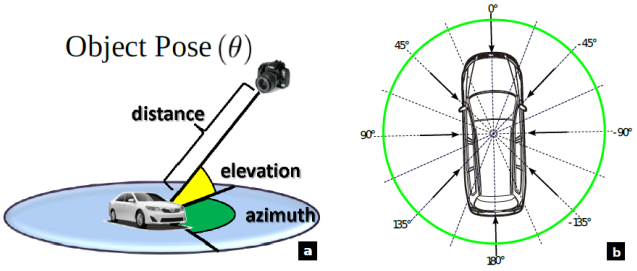
- PCA represents data in a lower-dimensional space keeping most of the variance
- It was seen to be powerful for similar patterns like faces, that exhibit a lot of redundancy
  $I = \mu + \sum_j^J \alpha_j U_j \qquad I : [\alpha_1, ..., \alpha_J]$
- Representation: PCA weights of the image – global image 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 azimut5h 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

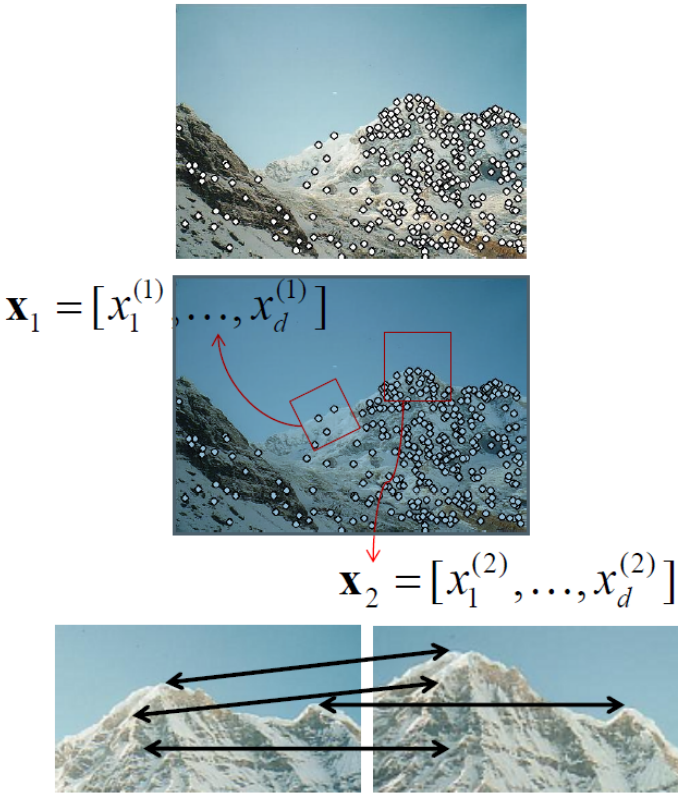
- Using the PCA representation identify the 3D face that yields the image when projected

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

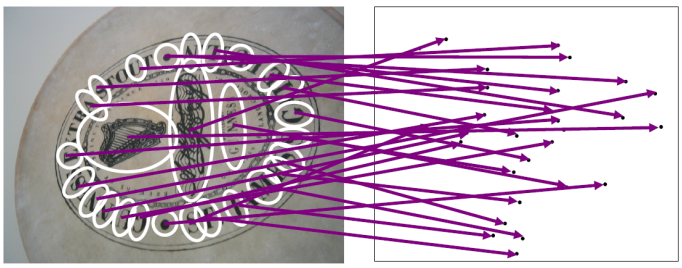
1. Too slow if naively done
   $\rightarrow$ Hierarchical vocabulary tree for speed-up
2. Will often fail when only based on descriptor matching

Indexing local features

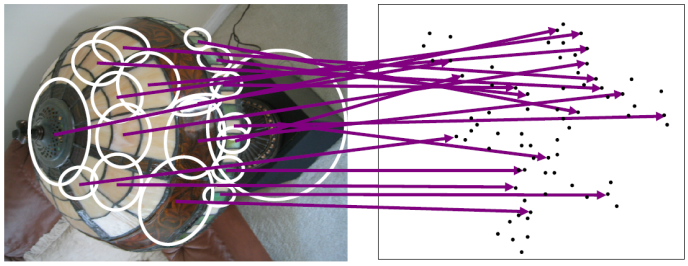
- With potentially thousands of interest pts + their descriptors per image, and hundreds to millions of images to search, how to efficiently find those relevant to a new test image?
- Quantize/cluster the descriptors into 'visual words'
  And match words hierarchically: vocabulary tree Use Inverted file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

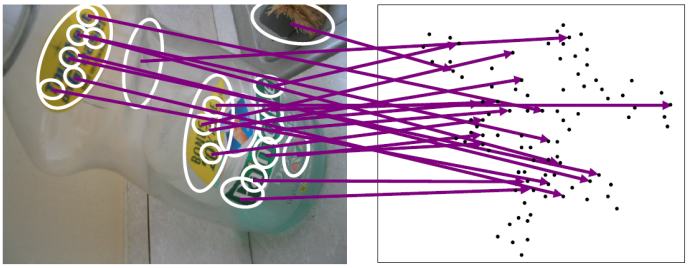
Visual words: main ideavisualwords

- Extract some local features from several images



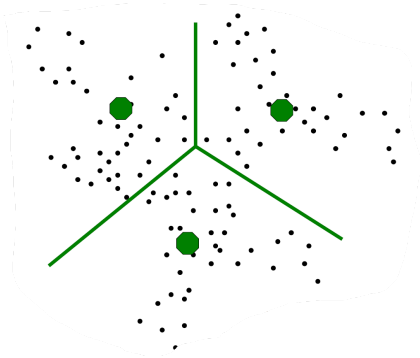
e.g., SIFT descriptor space: each point is 128-dimensional





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

- Vector quantize feature space = cluster the features



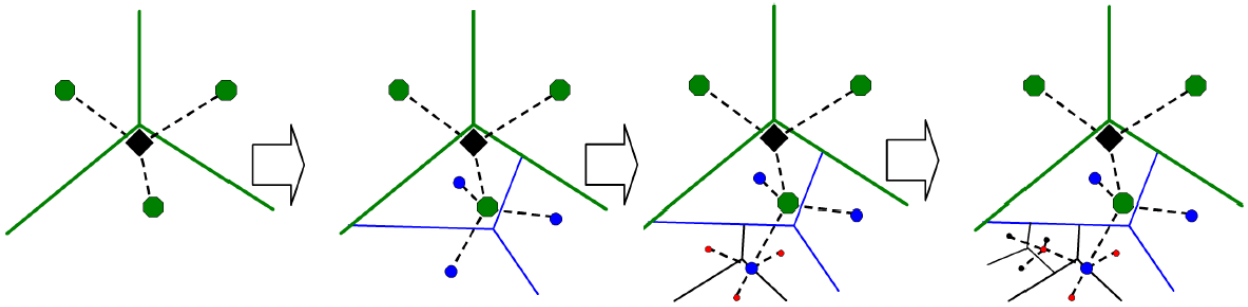
K-means clustering

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

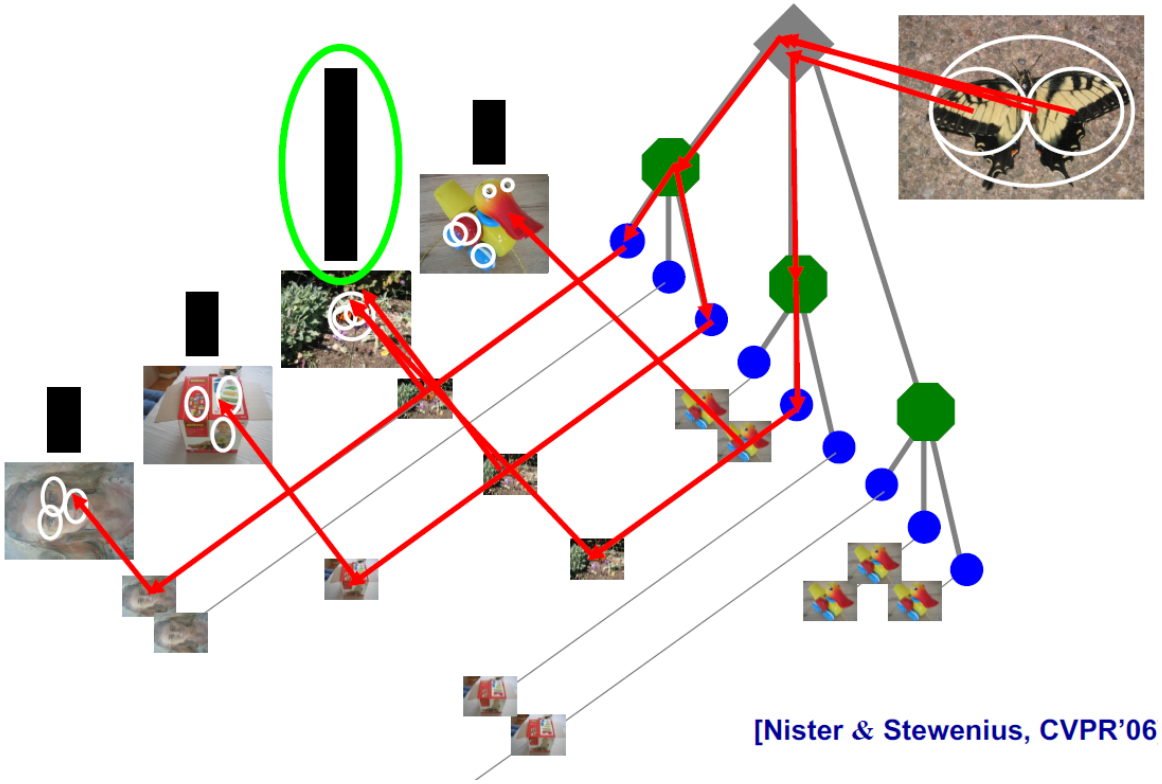
Indexing local features: inverted file index

- 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.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

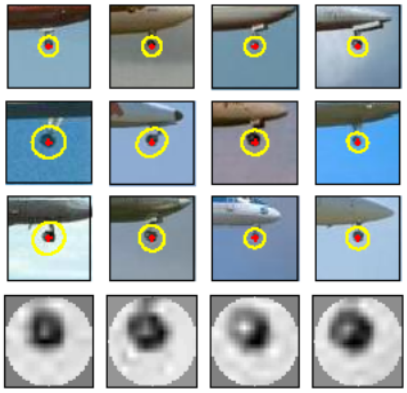
Intra-class and inter-class variation
The difference between classes can be as small as that between instances of the same class … yet the distinction needs to be made

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

Biounilding 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



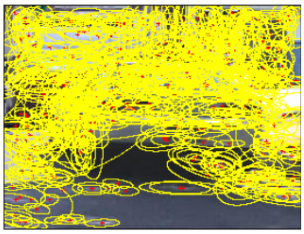
Sivic & Zisserman 2003; Csurka, Bray, Dance, & Fan 2004; many others.

Building Visual Vocabularies

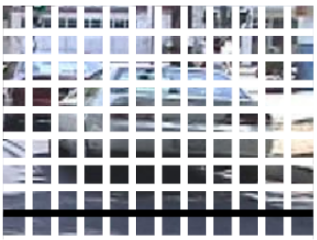
- Components:
  - Sampling strategy: extract features densely, or at interest points only, randomly, … ?
  - Clustering / quantization algorithm
- Questions / Issues:
  - Unsupervised vs. supervised
  - What corpus provides features (universal vocabulary)?
  - Vocabulary size, number of words

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]

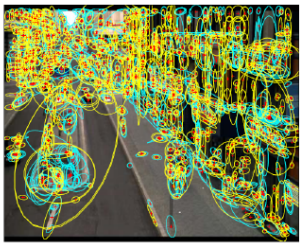


Sparse, at interest points



Dense, uniformly

Randomly



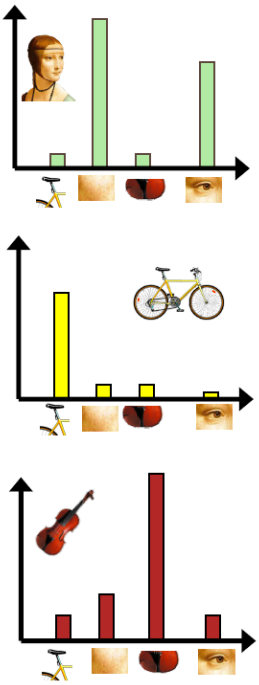
Multiple interest operators

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]

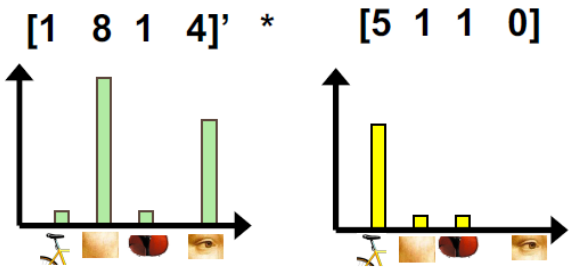
Bags of Visual Words

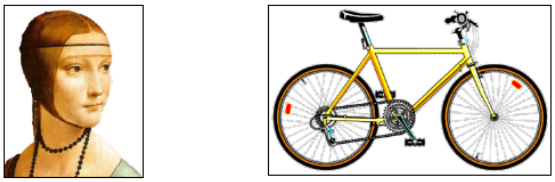
- Summarize entire image based on its distribution (histogram) of word occurrences.
- Analogous to bag of words representation commonly used for documents.
- Advantage of histogram is that irrespective of the nmb of local features, the size is always the same



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)



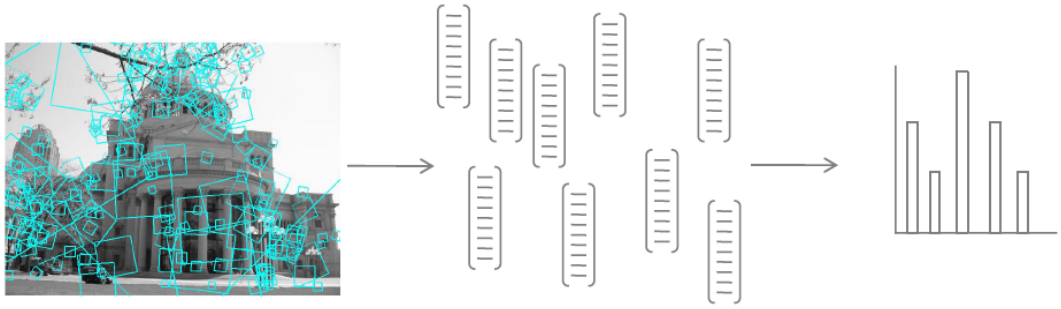


$\vec{d}_j$ (histograms in database) $\vec{q}$ (histogram query image)

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

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)

{face, flowers, building}

Image Classification (more sophisticated classifiers)

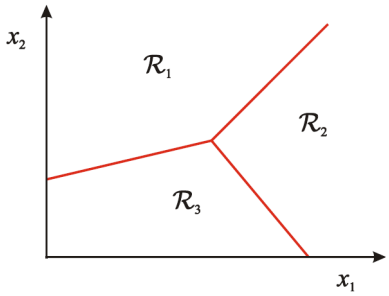
- Given bag-of-words of images from different classes, how do we learn a model for distinguishing them?

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

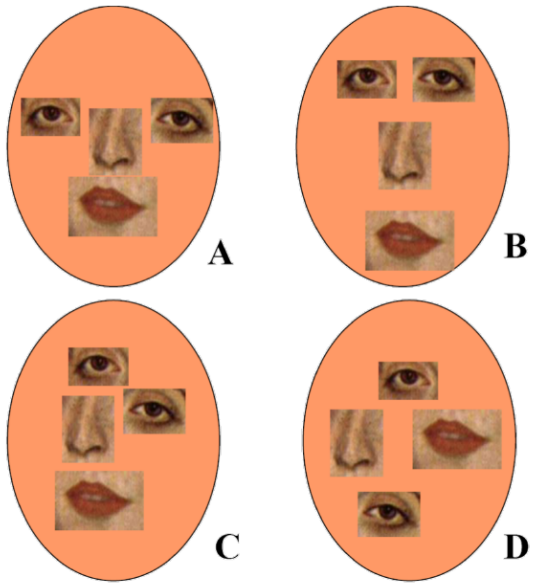
- Nearest neighbor
  Shakhnarovich, Viola, Darrell 2003 Berg, Berg, Malik 2005...
- Neural networks
  LeCun, Bottou, Bengio, Haffner 1998 Rowley, Baluja, Kanade 1998 …
- Support Vector Machines
  Guyon, Vapnik Heisele, Serre, Poggio, 2001,…
- Boosting
  Viola, Jones 2001, Torralba et al. 2004, Opelt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

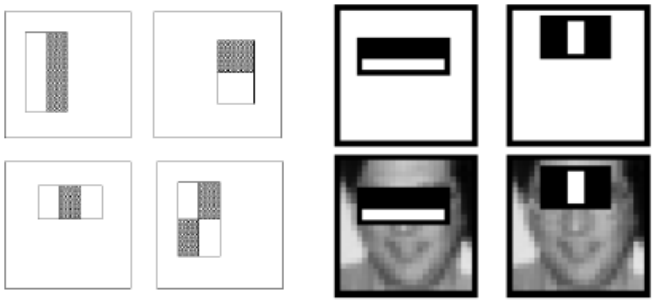
- One very popular example: Histogram of Oriented Gradients
  [Dalal & Triggs, CVPR 2005]

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 $\rightarrow$ 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 x 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?

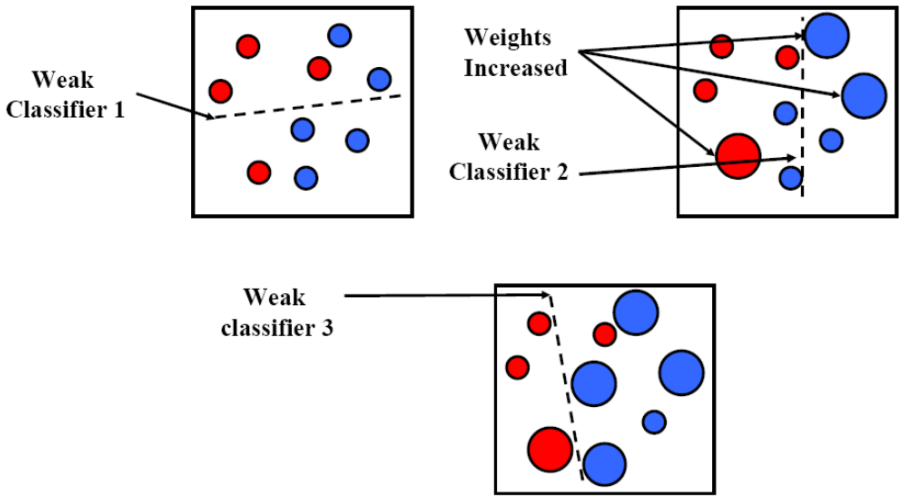
Sliding Windows: AdaBoost
Boosting

- 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

Y. Freund and R. Schapire, A short introduction to boosting, Journal of Japanese Society for Artificial Intelligence, 14(5):771-780, 1999

AdaBoost: Intuition

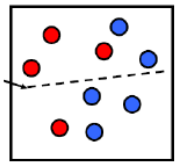
- Consider a 2D feature space with **positive and negative** examples.
- Each weak classifier splits the training examples with at least 50% accuracy.
- Examples misclassified by a previous weak learner are given more emphasis at future rounds.



- Final classifier is combination of the weak classifiers

- Given example images $(x_1, y_1), \ldots, (x_n, y_n)$ 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 $t = 1, \ldots, T$:
  1. 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.
  2. 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, $\epsilon_j = \sum_i w_i |h_j(x_i) - y_i|$.
  3. Choose the classifier, h_t, with the lowest error ϵ_t.
  4. Update the weights:
  $$w_{t+1,i} = w_{t,i} \beta_t^{1-e_i}$$
  where $e_i = 0$ if example x_i is classified correctly, $e_i = 1$ otherwise, and $\beta_t = \frac{\epsilon_t}{1-\epsilon_t}$.
- 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



For T rounds

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

Re-weight the examples:
incorrectly classified ⇒ more weight
Correctly classified ⇒ less weight

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

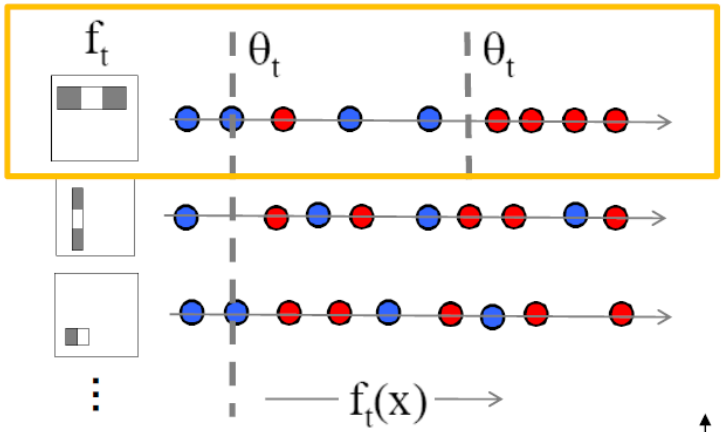
[Freund & Schapire 1995]

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.



Outputs of a possible rectangle feature on faces and non-faces.

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.

Slide credit: Kristen Grauman

[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

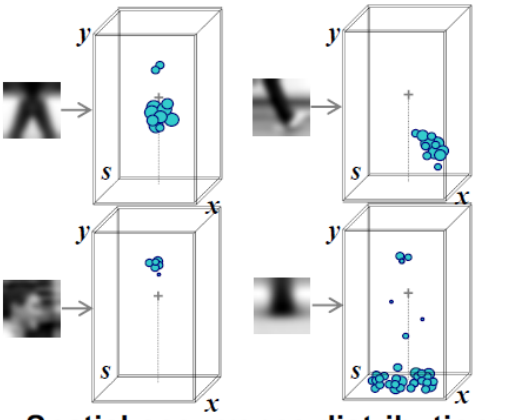
Implicit Shape Model: Basic Idea

- 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).

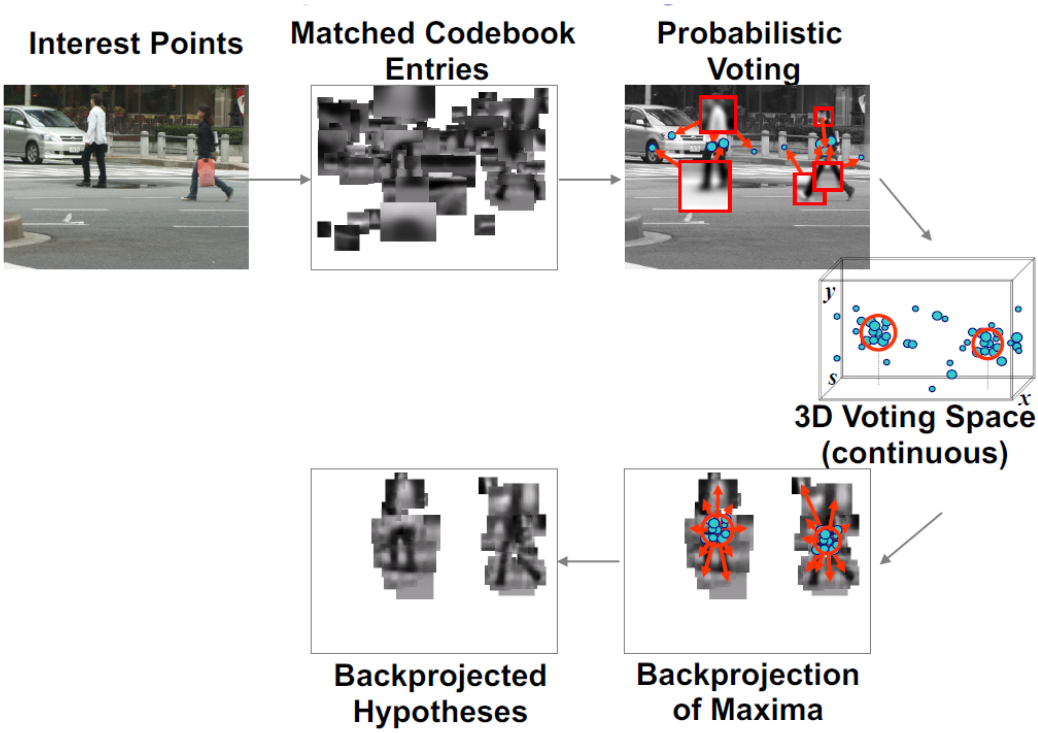
B. Leibe, A. Leonardis, and B. Schiele, Robust Object Detection with Interleaved Categorization and Segmentation, International Journal of Computer Vision, Vol. 77(1-3), 2008.

Implicit Shape Model - Representation

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



Spatial occurrence distributions



- 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

Motion Extraction

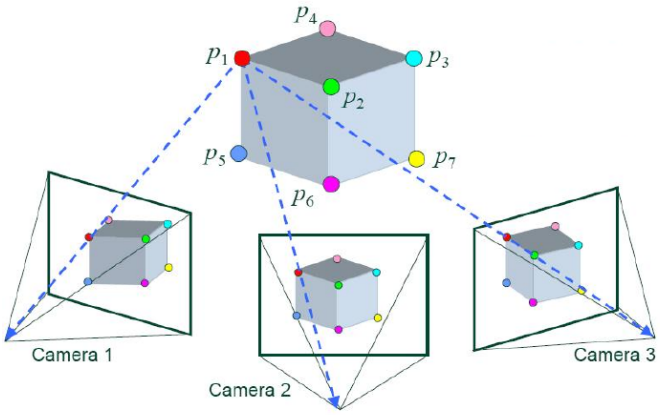
Lecture Notes

Motion is a basic cue

- Motion can be the only cue for segmentation
- Biologically favoured because of camouflage which set in motion a constant, evolutionary race
- Motion can be the only cue for segmentation
- Even impoverished motion data can elicit a strong percept

Some applications of motion extraction

- Change / shot cut detection
- Surveillance / traffic monitoring
- Autonomous driving
- Analyzing game dynamics in sports
- Motion capture / gesture analysis (HCI)
- Image stabilisation
- Motion compensation (e.g. medical robotics)
- Feature tracking for 3D reconstruction
  Tracked Points gives correspondences



- Etc.

optical flow

Definition of optical flow

- **OPTICAL FLOW = apparent motion of brightness patterns**
- Ideally, the optical flow is the projection of the threedimensional motion vectors on the image
- Such 2D motion vector is sought at every pixel of the image
  (note: a motion vector here is a 2D translation vector)

Caution required

Two examples where following brightness patterns is misleading:

1. Untextured, rotating sphere $\rightarrow O.F. = 0$
2. No motion, but changing lighting $\rightarrow O.F. \neq 0$

Qualitative formulation

- Suppose **a point of the scene** projects to a certain pixel of the current video frame. Our task is to figure out to which pixel in the next frame it moves…
- That question needs answering **for all pixels** of the current image.
- In order to find these corresponding pixels, we need to come up with a reasonable assumption on how we can detect them among the many.
- We assume these corresponding pixels have the **same intensities** as the pixels the scene points came from in the previous frame.
- That will only hold **approximately**...

Mathematical formulation

- Our mathematical representation of a video:
  $I(x, y, t) =$ brightness at (x, y) at time t
- **Optical flow constraint equation**:
  $\frac{dI}{dt} = \frac{\partial I}{\partial x}\frac{dx}{dt} + \frac{\partial I}{\partial y}\frac{dy}{dt} + \frac{\partial I}{\partial t} = 0$
  This equation states that if one were to track the image projections of a scene point through the video, it would not change its intensity. This tends to be **true over short lapses of time**.
  Note: $\frac{dI}{dt}$: Change of intensity when following a physical point through the images
  $\frac{\partial I}{\partial t}$: Change of intensity when looking at the same pixel (x,y) through the images
- denote $u = \frac{dx}{dt}, v = \frac{dy}{dt}, I_x = \frac{\partial I}{\partial x}, I_y = \frac{\partial I}{\partial y}, I_T = \frac{\partial I}{\partial t}$
  $I_x u + I_y v + I_t = 0$ (1 equation per pixel)
- Note that we can measure the 3 derivatives of I, but that u and v are unknown
  1 equation in 2 unknowns $\rightarrow$ the **'aperture problem'**

The aperture problem

- $I_x u + I_y v + I_t = 0 \Rightarrow (I_x\ I_y) \cdot (u\ v) = -I_t$
- Aperture problem : only the component **along the gradient** can be retrieved
  $\frac{I_t}{\sqrt{I_x^2 + I_y^2}}$

Remarks

1. The underdetermined nature could be solved using higher derivatives of intensity
2. for some intensity patterns, e.g. patches with a planar intensity profile, the aperture problem cannot be resolved anyway.

- For many images, large parts have planar intensity profiles… higher-order derivatives than 1st order are typically not used (also because they are noisy)

Horn & Schunck algorithm

- Breaking the spell via an additional smoothness constraint:
  $e_s = \int\int((u_x^2 + u_y^2) + (v_x^2 + v_y^2))dxdy$
  to be minimized besides the OF constraint equation term
  $e_c = \int\int(I_x u + I_y v + I_t)^2 dxdy$
- The integrals are over the image.
- minimize $e_s + \lambda e_c$ (also reduces influence of noise)

The calculus of variations

- look for functions that extremize **functionals**
  (a functional is a function that takes a vector as its input argument, and returns a scalar)
  like for our functional: $\int\int((u_x^2 + u_y^2) + (v_x^2 + v_y^2))dxdy + \lambda\int\int(I_x u + I_y v + I_t)^2 dxdy$
  what are the optimal u(x,y) and v(x,y)?
- $I = \int_{x_1}^{x_2} F(x, f, f')dx$ with $f = f(x), f' = \frac{df}{dx}$
  $f(x_1) = f_1$ and $f(x_2) = f_2$
- Suppose
  1. $f(x)$ is a solution
  2. $\eta(x)$ is a test function with $\eta(x_1) = 0$ and $\eta(x_2) = 0$

  We then consider
  $I = \int_{x_1}^{x_2} F(x, f + \varepsilon\eta, f' + \varepsilon\eta')dx \qquad (f \to f + \varepsilon\eta)$

  > Rationale: supposed f is the solution, then any deviation should result in a worse I; when applying classical optimization over the values of ε the optimum should be $\varepsilon = 0$
  > With this trick, we reformulate an optimization over a function into a classical optimization over a scalar… a problem we know how to solve

  for the optimum:
  $\frac{dI}{d\varepsilon}|_{\varepsilon=0} = 0$
  Around the optimum, the derivative should be zero.
  $\int_{x_1}^{x_2}(\eta(x)F_f + \eta'(x)F_{f'})dx = 0$
  Using integration by parts:
  $\int_{x_1}^{x_2}\frac{d}{dx}(gh)dx = \int_{x_1}^{x_2}(\frac{dg}{dx}h + \frac{dh}{dx}g)dx = [gh]_{x_1}^{x_2}$
  where $[gh]_{x_1}^{x_2} = g(x_2)h(x_2) - g(x_1)h(x_1)$
  $\to \int_{x_1}^{x_2}\frac{d}{dx}(\eta(x)F_{f'})dx$:
  $\int_{x_1}^{x_2}\eta'(x)F_{f'} + \eta(x)\frac{d}{dx}F_{f'}dx = [\eta(x)F_{f'}]_{x_1}^{x_2}$
  $\int_{x_1}^{x_2}\eta'(x)F_{f'}dx = [\eta(x)F_{f'}]_{x_1}^{x_2} - \int_{x_1}^{x_2}\eta(x)\frac{d}{dx}F_{f'}dx$

Therefore, $\int_{x_1}^{x_2} \eta(x)(F_f - \frac{d}{dx}F_{f'})dx = 0$
regardless of $\eta(x)$, then $F_f - \frac{d}{dx}F_{f'} = 0$ (**Euler-Lagrange equation**)

- Generalizations
  1. $I = \int_{x_1}^{x_2} F(x, f_1, f_2, .., f_1', f_2', ...)dx$
     Simultaneous Euler-Lagrange equations, i.e. one for u and one for v:
     $F_{f_i} - \frac{d}{dx}F_{f_i'} = 0$
     As we add ε_1, η_1 to f_1 and ε_2, η_2 to f_2, then repeat,
     once deriving w.r.t ε_1, once w.r.t ε_2, thus obtaining a system of 2 PDEs
  2. 2 independent variables x and y
     $I = \int\int_D F(x, y, f + \varepsilon\eta, f_x + \varepsilon\eta_x, f_y + \varepsilon\eta_y)dxdy$
     Hence $0 = \int\int_D (\eta F_f + \eta_x F_{f_x} + \eta_y F_{f_y})dxdy$ **(∗)**
     Now by Gauss's integral theorem $\int\int_D (\frac{\partial Q}{\partial x} + \frac{\partial P}{\partial y})dxdy = \int_{\partial D}(Qdy - Pdx)$
     such that $\int\int_D (\frac{\partial \eta F_{f_x}}{\partial x} + \frac{\partial \eta F_{f_y}}{\partial y})dxdy = \int_{\partial D}(\eta F_{f_x}dy - \eta F_{f_y}dx) = 0$
     $\int\int_D (\eta_x F_{f_x} + \eta_y F_{f_y})dxdy + \int\int_D (\eta \frac{\partial F_{f_x}}{\partial x} + \eta \frac{\partial F_{f_y}}{\partial y})dxdy = 0$
     Consequently, with(∗), $0 = \int\int_D \eta(F_f - \frac{\partial}{\partial x}F_{f_x} - \frac{\partial}{\partial y}F_{f_y})dxdy = 0$ $\rightarrow$is the **Euler-Lagrange equation**

Horn & Schunck

- The Euler-Lagrange equations:
  $F_u - \frac{\partial}{\partial x}F_{u_x} - \frac{\partial}{\partial y}F_{u_y} = 0$
  $F_v - \frac{\partial}{\partial x}F_{v_x} - \frac{\partial}{\partial y}F_{v_y} = 0$
- In our case, $F = (u_x^2 + u_y^2) + (v_x^2 + v_y^2) + \lambda(I_x u + I_y v + I_t)^2$
  so the Euler-Lagrange equations are
  $Deltau = \lambda(I_x u + I_y v + I_t)I_x$
  $Deltav = \lambda(I_x u + I_y v + I_t)I_y$
  where $\Delta = \frac{\partial^2}{\partial x^2} + \frac{\partial^2}{\partial y^2}$ is the Laplacian operator
- Remarks:
  1. Coupled PDEs solved using iterative methods and finite differences (iteration i)
     $\frac{\partial u}{\partial i} = \Delta u - \lambda(I_x u + I_y v + I_t)I_x$
     $\frac{\partial v}{\partial i} = \Delta v - \lambda(I_x u + I_y v + I_t)I_y$
  2. More than two frames allow for a better estimation of I_t
  3. Information spreads from edge- and corner-type patterns
  4. Errors at object boundaries (where the smoothness constraint is no longer valid)
  5. Example of **regularisation** (selection principle for the solution of ill-posed problems by imposing an extra generic constraint, like here smoothness)

Other approaches

1. **Model-based tracking (application-specific)**
    - active contours
    - analysis/synthesis schemes
    - tracking-by-detection
2. **Feature tracking (more generic)**
    - corner tracking
    - blob/contour tracking
    - intensity profile tracking
    - region tracking
    - condensation tracker (e.g. blob w. color hist.)

3D acquisition

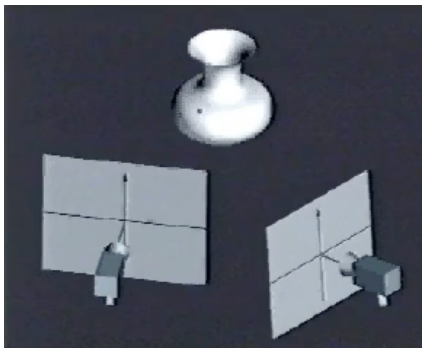
Lecture Notes

3D acquisition taxonomy

- 3D acquisition methods
    - passive
        - uni-directional
            - Shape-from
                - texture
                - contour
                - silhouettes
                - defocus
                - shading
        - multi-directional
            - Stereo
    - active
        - uni-directional
            - time-of-flight
        - multi-directional
            - Line scanning
            - Structured light
            - Photom. stereo

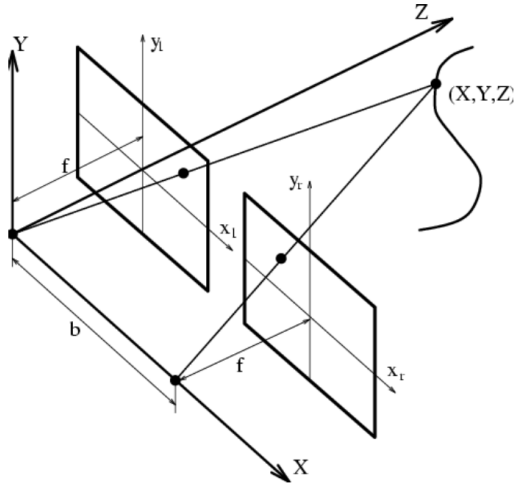
Stereo

- The underlying principle is **triangulation**:



(Passive) stereo

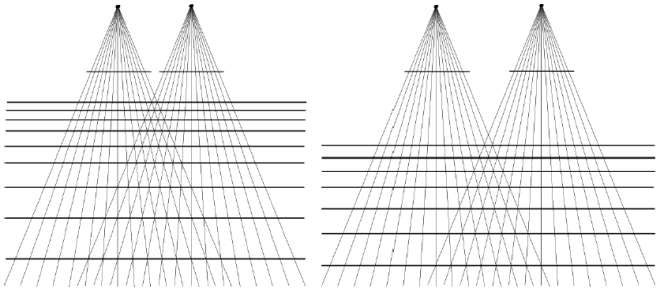
- Simple configuration:



A simple stereo setup

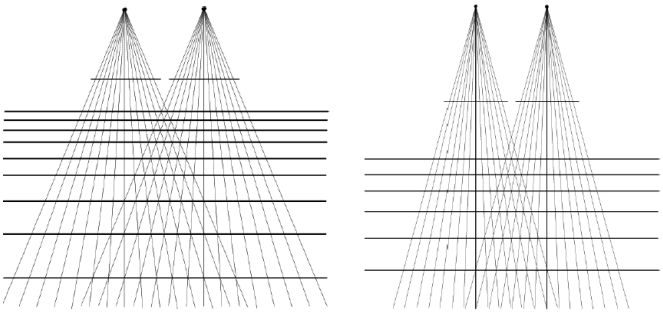
- identical cameras
- coplanar image planes
- aligned x-axes
- Reminder:
  the camera projection can be formulated as
  $\rho p = KR^t(P - C)$ for some non-zero $\rho \in \mathbb{R}$
  Here R is the identity (world coord system attached to left cam)
  $$\rho \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} = K \begin{pmatrix} X \\ Y \\ Z \end{pmatrix}, \rho' \begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = K \begin{pmatrix} X - b \\ Y \\ Z \end{pmatrix}, K = \begin{pmatrix} fk_x & 0 & 0 \\ 0 & fk_y & 0 \\ 0 & 0 & 1 \end{pmatrix}$$
  $$\begin{cases} x = \frac{fk_x X}{Z} \\ y = \frac{fk_y Y}{Z} \end{cases}, \begin{cases} x' = \frac{fk_x (X-b)}{Z} \\ y' = \frac{fk_y Y}{Z} \end{cases}$$ Note that $y = y'$
- The 3D coordinates of the point are
  $X = b\frac{x}{(x-x')}, Y = b\frac{k_x}{k_y}\frac{y}{(x-x')}, Z = bk_x\frac{f}{(x-x')}$
  Note that the same horizontal shift $x - x'$ yields the same depth
  $x - x'$ is the so-called **disparity**

- Stereo is imprecise for far away objects,but increasing b and/or f can increase depth resolution
  - Increasing b increases depth resolution



one has to strike a balance with visibility...

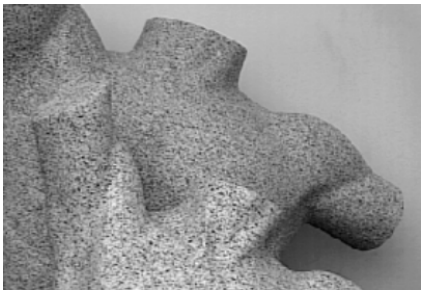
  - Increasing f increases depth resolution



Remarks

1. increasing b and/or f reduces simultaneous visibility
2. iso-disparity loci are depth planes, not so for other stereo configurations
3. human stereo vision only up to $\pm$ 10 m
4. the real problem is finding correspondences

- The HARD problem is finding the **correspondences**



Notice: no reconstruction for the untextured back wall...

Stereo, the general setup

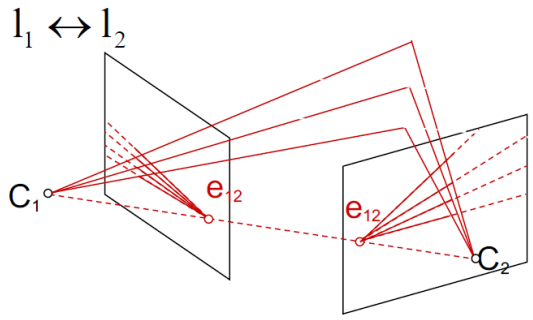
- We start by the relation between the two projections of a point, to ease correspondence search
- In the second image the point must be along the projection of the viewing ray for the first camera: We cast this constraint in mathematical expressions:

p and p' are the two images of P
$\mu p = K R^t (P - C)$
$\rho' p' = K' R'^t (P - C')$
w.r.t. world frame P is on the ray with equation
$P = C + \mu R K^{-1} p \qquad$ for some $\mu \in \mathbb{R}$
and thus, filling in the ray's equation

- $\rho' p' = \mu K' R'^t R K^{-1} p + K' R'^t (C - C')$
  the second term is the projection of the 1st camera's center, the so-called **epipole**
  $\rho'_e e' = K' R'^t (C - C')$
  the first term is the projection of the ray's point at infinity, the so-called **vanishing point**
  finally, adopting the simplifying notation $A = \frac{1}{\rho'_e} K' R'^t R K^{-1}$
  $\rho' p' = \rho'_e (\mu A p + e')$, A is the **infinity homography**
  this projected ray is called the **epipolar line** for p and runs through the points Ap and e'
  note that the epipole lies on all the epipolar lines
- $\rho' p' = \rho'_e (\mu A p + e')$ expresses that p' lies on the line l' through the epipole e' and the vanishing point Ap of the ray of sight of p
  the epipolar constraint (epipolar line)
  we can rewrite this constraint as $|p' e' A p| = p'^t (e' \times A p) = 0$
  can be written, given $[e']_\times = \begin{pmatrix} 0 & -e'_3 & e'_2 \\ e'_3 & 0 & -e'_1 \\ -e'_2 & e'_1 & 0 \end{pmatrix}$
  as $|p' e' A p| = p'^t [e']_\times A p$
  $F = [e']_\times A$ is the **fundamental matrix**, and has rank 2
- one point yields one equation $p'^t F p = 0$ that is linear in the entries of the fundamental matrix F
  so, we can actually obtain F without any prior knowledge about camera settings if we have sufficient pairs of corresponding points !!
- F can be computed **linearly** from 8 pairs of corresponding points, i.e. already from 8 'correspondences' (not 9, as this is a homogeneous system!)
- F being rank 2 yields an additional, but non-linear constraint. Thus, 7 correspondences suffice to **non-linearly** solve for F
- Remarks:
  Of course, in practice one wants to use extra correspondences if available, e.g. for obtaining a least-squares solution, based on the linear system, followed by a step to impose rank 2.
  Interest points (found with some detector and described by a descriptor) are good candidates to find the initial correspondences.

Epipolar correspondence

- Epipolar lines are in mutual correspondence



- allows to separate matching problem

Example: epipolar geometry

Exploiting epipolar geometry

Separate 2D correspondence search problem to 1D search problem by using two view geometry

Correspondence problem: constraints

Reducing the search space:

1. Points on the epipolar line
2. Min. and max. depth ⇒ line segment
3. Preservation of order
4. Smoothness of the disparity field

Correspondence problem : methods

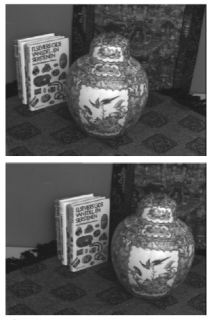
1. **correlation**
   - deformations…
   - small window ⇒ noise!
   - large window ⇒ bad localisation
2. **feature-based**
   - mainly edges and corners
   - sparse depth image
3. **regularisation methods**

Stereo, the general setup

- 3D reconstruction, for calibrated setup
  $P = C + \mu R K^{-1} p \qquad P = C' + \mu R' K'^{-1} p'$
  Yields 6 equations in 5 unknowns X, Y, Z and μ, μ'
  However, due to noise and errors, the rays may not intersect!
  $\rightarrow$ e.g. use the middle where the rays come closest
- Notice that this 3D reconstruction requires the stereo system to be calibrated
- From correspondences only some information about the 3D structure is still available, cf. next slides

Uncalibrated reconstruction

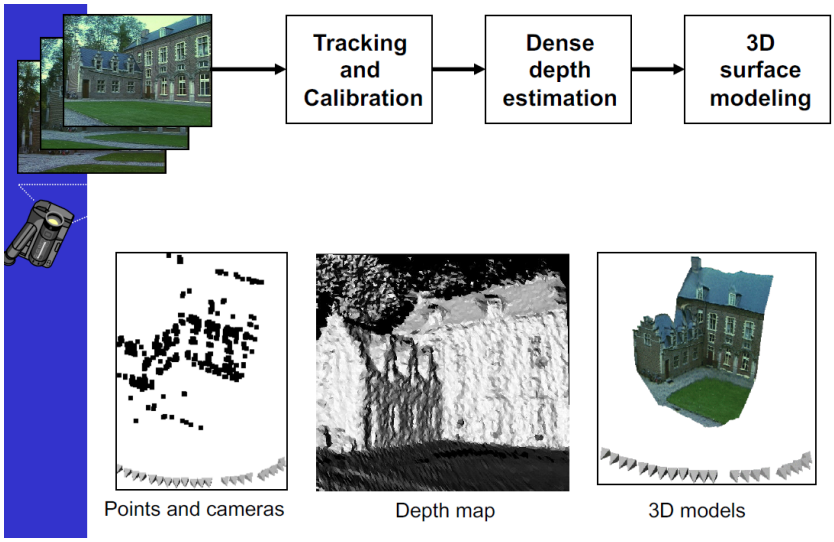
From 2 views...



If the camera translates...

- An affine reconstruction can be made
  A projective reconstruction is always possible
  (if no pure rotation)
- From 3 general views, taken with the same camera parameters...

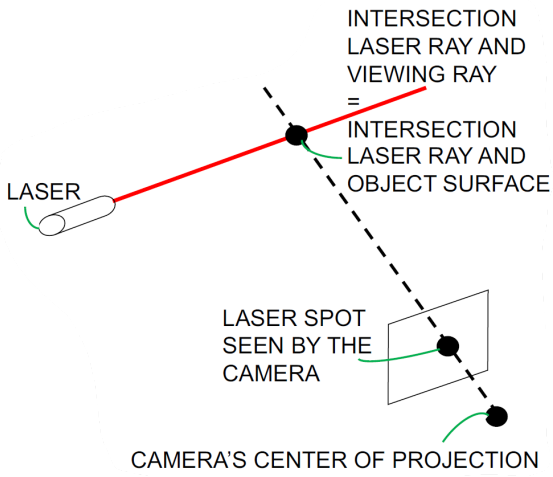
- A metric reconstruction is possible



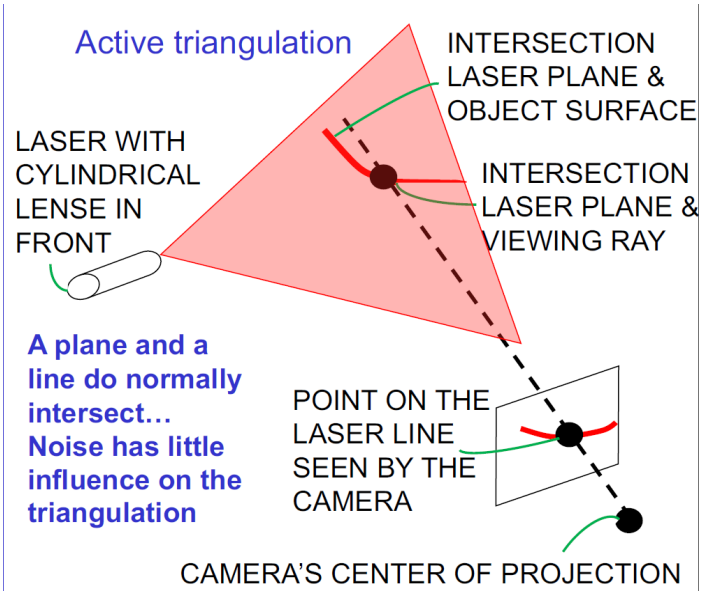
Active triangulation

- Idea: simplify the correspondence problem of stereo by replacing one of the two cameras by a projector…
- e.g. a laser -> the bright laser point is immediately visible in the remaining camera
- Then triangulate the laser line and the point's camera ray
- But in order to get more than one point in 3D, the laser point has to be moved over the object surface

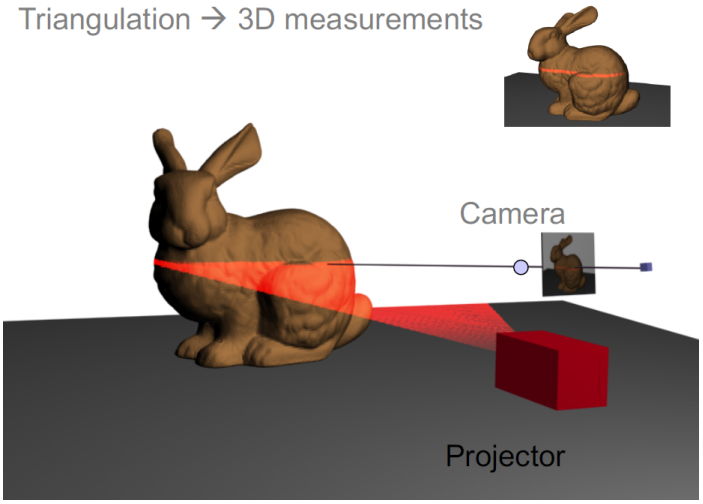
- Per point an image then needs to be taken, after which the laser can project the next point



- NO INTERSECTION IF SOME **ERRORS** IN THE LINE EQ.S!
- Two lines do normally not intersect... **Noise** disrupts triangulation



- Triangulation → 3D measurements



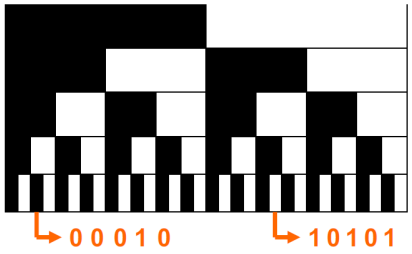
Structured light

- patterns of a special shape are projected onto the scene

- deformations of the patterns yield information on the shape
- Focus is on combining a good resolution with a minimum number of pattern projections

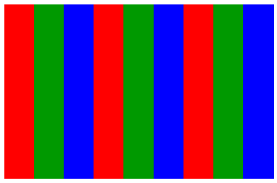
Serial binary patterns

- A sequence of patterns with increasingly fine subdivisions
  Yields 2^n identifiable lines for only n patterns



Reducing the nmb of projections: colour

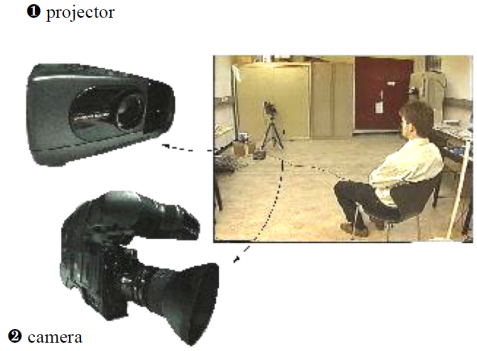
- Binary patterns
  Yields identifiable lines for 2^n only n patterns
- Using colours, e.g. 3,
  Yields 3^n identifiable lines for only n patterns



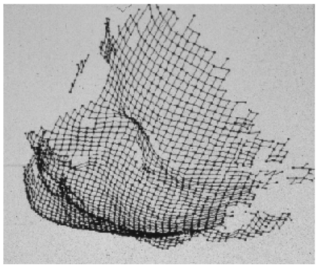
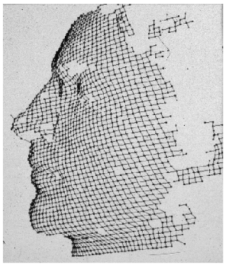
Interference from object colours...

One-shot implementation

- The setup is extremely simple now:



- A checkerboard pattern carrying a column code

- An application in agriculture
- Eyetronics' ShapeCam
- Shape + texture often needed
  - Higher resolution
  - Texture is also extracted
- Recent, commercial example: Kinect
  - Kinect 3D camera, affordable and compact solution by Microsoft.
  - Projects a 2D point pattern in the NIR, to make it invisible to the human eye
  - Kinect as one-shot, low-cost scanner
    Excerpt from the dense NIR dot pattern

4D: Face animation - input

4D: Face animation – replay + effects

4D: Facial motion capture

motion capture for League of Extraordinary Gentlemen

Facial motion capture

With 2 patterns…!

Compare the reflectance for

- a pattern with linearly changing intensity
- a pattern with homogeneous intensity

Phase shift

$$I_r = A + Rcos(\phi - \theta)$$
$$I_g = A + Rcos(\phi)$$
$$I_b = A + Rcos(\phi + \theta)$$

1. detect phase from 3 subsequently projected cosine patterns, shifted over 120 degrees
2. unwrap the phases / additional stereo
3. texture is obtained by summing the 3 images / color camera w. slower integration

$$A = \frac{I_r + I_g + I_b}{3}$$
$$\phi = arctan(tan(\frac{\theta}{2})\frac{I_r - I_b}{2I_g - I_r - I_b})$$

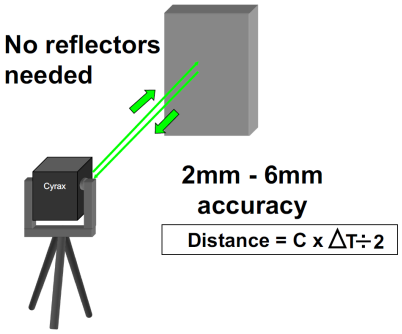
4D acquisition

Motion retargetting, from 3D phase shift scans

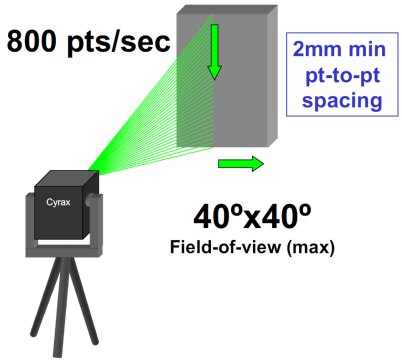
Time-of-flight

- measurement of the time a modulated light signal needs to travel before returning to the sensor
- this time is proportional to the distance
- waves :
  1. **radar** *low freq. electromagnetic*
  2. **sonar** *acoustic waves*
  3. **optical** *radar optical waves*
- working principles :
  1. pulsed
  2. phase shifts
- Example 1: Cyrax

  Accurate, detailed, fast measuring
  - Pulsed laser



  - Laser sweeps over surface



  - Up to 100m range (50m rec)
  - Cyrax is also a visualization tool

    Cyrax detects the **intensity** of each reflected laser pulse and colors it
  - Step 1: Target the structure

    Step 2: Scan the structure

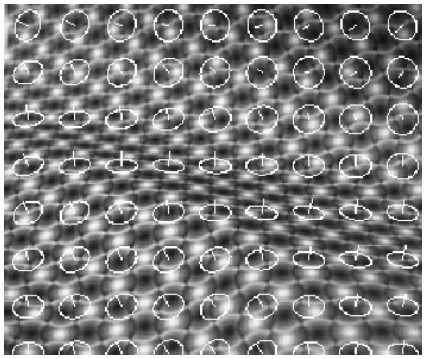
    Step 3: Color the points

    Step 4: Model fitting in-the-field

    - Project: As-built of Chevron hydrocarbon plant
      Cost Benefits, Greater detail & no errors, Higher accuracy, Fewer construction errors, 6 week schedule savings
- Example 2: Riegl

Shape-from-texture

- assumes a slanted and tilted surface to have a homogeneous texture
- inhomogeneity is regarded as the result of projection
- e.g. anisotropy in the statistics of edge orientations $\rightarrow$ orientations deprojecting to maximally isotropic texture

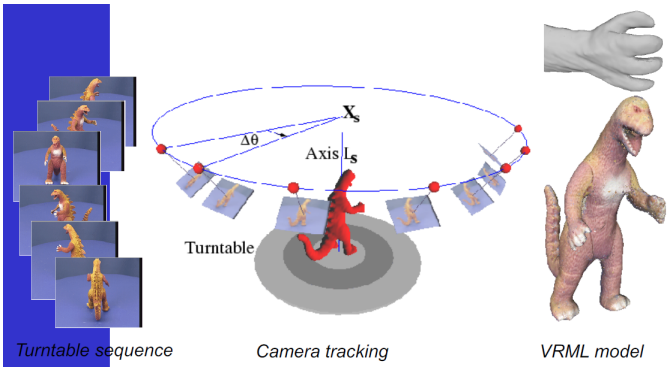


Shape-from-contour

- makes assumptions about contour shape
- E.g. the maximization of area over perimeter squared (compactness) $\Downarrow$
  ellipse $\rightarrow$ circle
- E.g. assumption of symmetry $\Downarrow$
  Symmetric contours $\rightarrow$ surface of revolution

Shape-from-silhouettes - uncalibrated

- tracking of turntable rotation
    - volumetric modeling from silhouettes
    - triangular textured surface mesh



Shape-from-shading

- Uses directional lighting, often with known direction
- local intensity is brought into correspondence with orientation via **reflectance maps**
- orientation of an isolated patch cannot be derived uniquely
- extra assumptions on surface smoothness and known normals at the rim

Photometric stereo

- constraint propagation eliminated by using light from different directions
- simultaneously when the light sources are given different colours

Tracking

Lecture Notes

What's Tracking

- "follow the movements of somebody/something" – Oxford Dictionary
- Something
  - Point
  - Region
  - Template
  - Part
  - Segmentation
- Somebody
  - An object with known identity
  - Pose
  - A person, a vehicle, a face, etc.

Tracking Applications

- Autonomous Driving
- Image Editing
- Safety Monitoring
- Sports
- AR/VR
- Space Management
- You can easily name more!

Tracking Problems

- Shape of the tracking targets

    - Point
    - Box
    - Segmentation
- Prior knowledge of the target
    - Arbitrary targets
    - Known properties
    - Known semantic categories

Two Scenarios

- Scenario 1
    - Track without a concrete example
    - Track a known category such as corners or vehicles
    - Primarily used in autonomous systems
- Scenario 2
    - Track with an example or a specific object
    - Commonly used in image editing
    - Also called interactive tracking sometimes

Challenges

- Temporal information analysis
    - How to use the multiple frames
    - How to represent the tracking task
- Large amount of data
    - Especially expensive to run deep learning models
- Incorporating prior information
    - Locating the known targets may become the accuracy bottleneck

Excitement

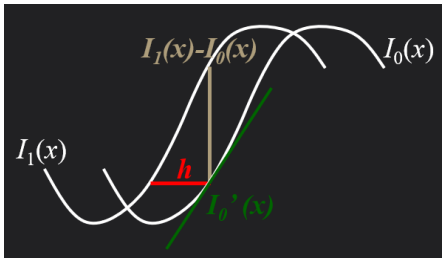
- It is intellectually challenging
- A basic building block for real-world applications
- Video sources are everywhere!

Track a Point

- find the point with the same color
  $E(h) = [(I_0(x+h) - I_1(x)]^2$
  Optimize displacement for this energy so the two pixels have the same color
  $E(h) \approx [I_0(x) + hI_0'(x) - I_1(x)]^2$

$$\frac{\partial E}{\partial h} = 2I_0'(x)[I_0(x) + hI_0'(x) - I_1(x)]$$
$$\frac{\partial E}{\partial h} = 0 \rightarrow h \approx \frac{I_1(x) - I_0(x)}{I_0'(x)}$$

Intuition



Problem 1 Zero Gradient

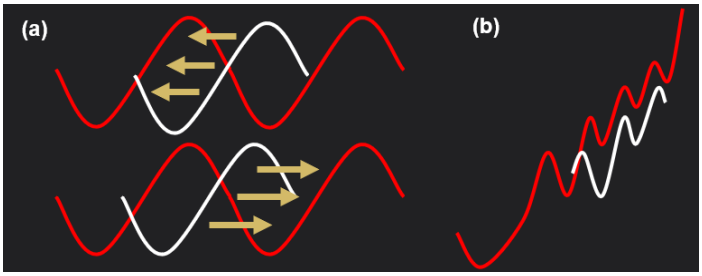
- What if the image gradient is 0?
  a.k.a. The image region is flat!

Problem 1 "Aperture problem"

- For tracking to be well defined, nonzero gradients in all possible directions are needed
- If no gradient along one direction, we cannot determine relative motion in that axis

Problem 2 Local Minima

- Motion to closest minimum has to be assumed
- Indirect result: Frame-rate should be faster than motion "of half-wavelength" (**Nyquist rate**)
- Nonconvex regions may indicate multiple solutions



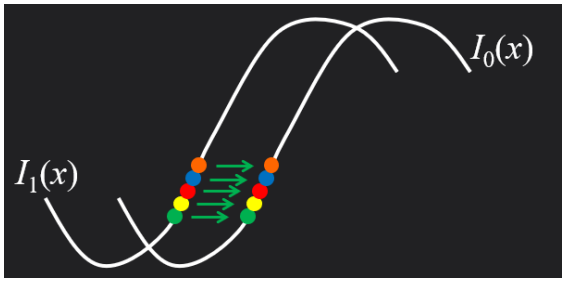
Recall: Optical Flow in Motion Estimation

- Optical Flow recovers (smooth) motion everywhere
- Least-squares regularization: Horn-Schunk makes smooth spatial change assumption
- Assuming displacements are small and using Taylor expansion
  $I_x u + I_y v + I_t = 0$
  $I_x = \frac{\partial I}{\partial x}, I_y = \frac{\partial I}{\partial y}, I_t = \frac{\partial I}{\partial t}$

1 equation in 2 unknowns

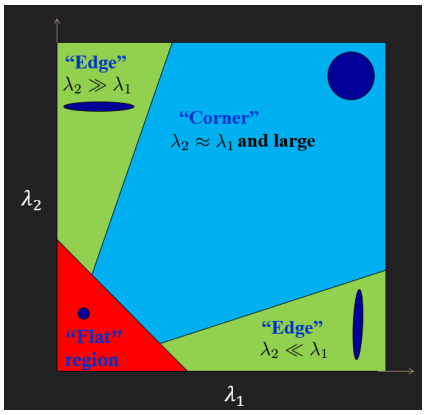
Treating Aperture Problem in Tracking

- Get additional info to constrain motion:
  - Optical Flow: Smoothly regularize in space
  - Tracking: Assume single motion for a region
- Spatial coherence constraint: “A pixel’s neighbors all move together”



Least Squares Problem

- $$\begin{pmatrix} I_x(\mathbf{p}_1) & I_y(\mathbf{p}_1) \\ I_x(\mathbf{p}_2) & I_y(\mathbf{p}_2) \\ \vdots & \vdots \\ I_x(\mathbf{p}_n) & I_y(\mathbf{p}_n) \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = - \begin{pmatrix} I_t(\mathbf{p}_1) \\ I_t(\mathbf{p}_2) \\ \vdots \\ I_t(\mathbf{p}_n) \end{pmatrix}$$
- Over determined System of Equations $A\mathbf{d} = \mathbf{b}$
  Pseudo Inverse $(A^T A)\mathbf{d} = A^T \mathbf{b}$
- $$\begin{pmatrix} \sum I_x I_x & \sum I_x I_y \\ \sum I_x I_y & \sum I_y I_y \end{pmatrix} \begin{pmatrix} u \\ v \end{pmatrix} = - \begin{pmatrix} \sum I_x I_t \\ \sum I_y I_t \end{pmatrix}$$
- (u,v) can only be found, if this is solvable
  - 2x2 image structure matrix is invertible
  - with no small eigenvalue
- This matrix and the requirement sound familiar – have we seen these before?
- Recall Harris corner detector!
- Thus, good image features (with large structural eigenvalues) are also good for tracking (with which we can find motion)



Track a Bigger Area

Template Tracking

- Keep a template image to compare with each frame
- This is typically applied for small patches, e.g. 5x5
- Why not run it for the entire object (for a larger window)
- What if the template takes some transformation
  - Translation
  - Rotation
  - Scaling
  - Projective

Lucas-Kanade Template Tracker

- We can easily generalize the motion model to other parametric models!
  $E(u,v) = \sum_{x,y}[I(x+u, y+v) - T(x,y)]^2$
  $\rightarrow E(p) = \sum_{x,y}[I(W(x;p)) - T(x,y)]^2$
- At an iterative step, assuming we know the "optimal" warp $W(x;p)$,
  $\sum_{\mathbf{x}}[I(\mathbf{x}+\Delta\mathbf{x}) - T(\mathbf{x})]^2$
  $\rightarrow \sum_{\mathbf{x}}[I(W(\mathbf{x};\mathbf{p}+\Delta\mathbf{p})) - T(\mathbf{x})]^2$
  $\rightarrow \sum_{\mathbf{x}}[I(W(\mathbf{x};\mathbf{p})) + \nabla I\frac{\partial W}{\partial \mathbf{p}}\Delta\mathbf{p} - T(\mathbf{x})]^2$
  Derivative of Δp: $\rightarrow \sum_{\mathbf{x}}[\nabla I\frac{\partial W}{\partial \mathbf{p}}]^T[I(W(\mathbf{x};\mathbf{p})) + \nabla I\frac{\partial W}{\partial \mathbf{p}}\Delta\mathbf{p} - T(\mathbf{x})]^2$
  set the derivative to 0: $\rightarrow \Delta\mathbf{p} = H^{-1}\sum_{\mathbf{x}}[\nabla I\frac{\partial W}{\partial \mathbf{p}}]^T[T(\mathbf{x}) - I(W(\mathbf{x};\mathbf{p}))]^2$
  where $H = \sum_{\mathbf{x}}[\nabla I\frac{\partial W}{\partial \mathbf{p}}]^T[\nabla I\frac{\partial W}{\partial \mathbf{p}}]$
- Iterative update

  Warp I to obtain $I(W([x\ y];P))$

  Compute the error image $T(x) - I(W([x\ y];P))$

  Warp the gradient ∇I with $W([x\ y];P)$

  Evaluate $\frac{\partial W}{\partial P}$ at $([x\ y];P)$ (Jacobian)

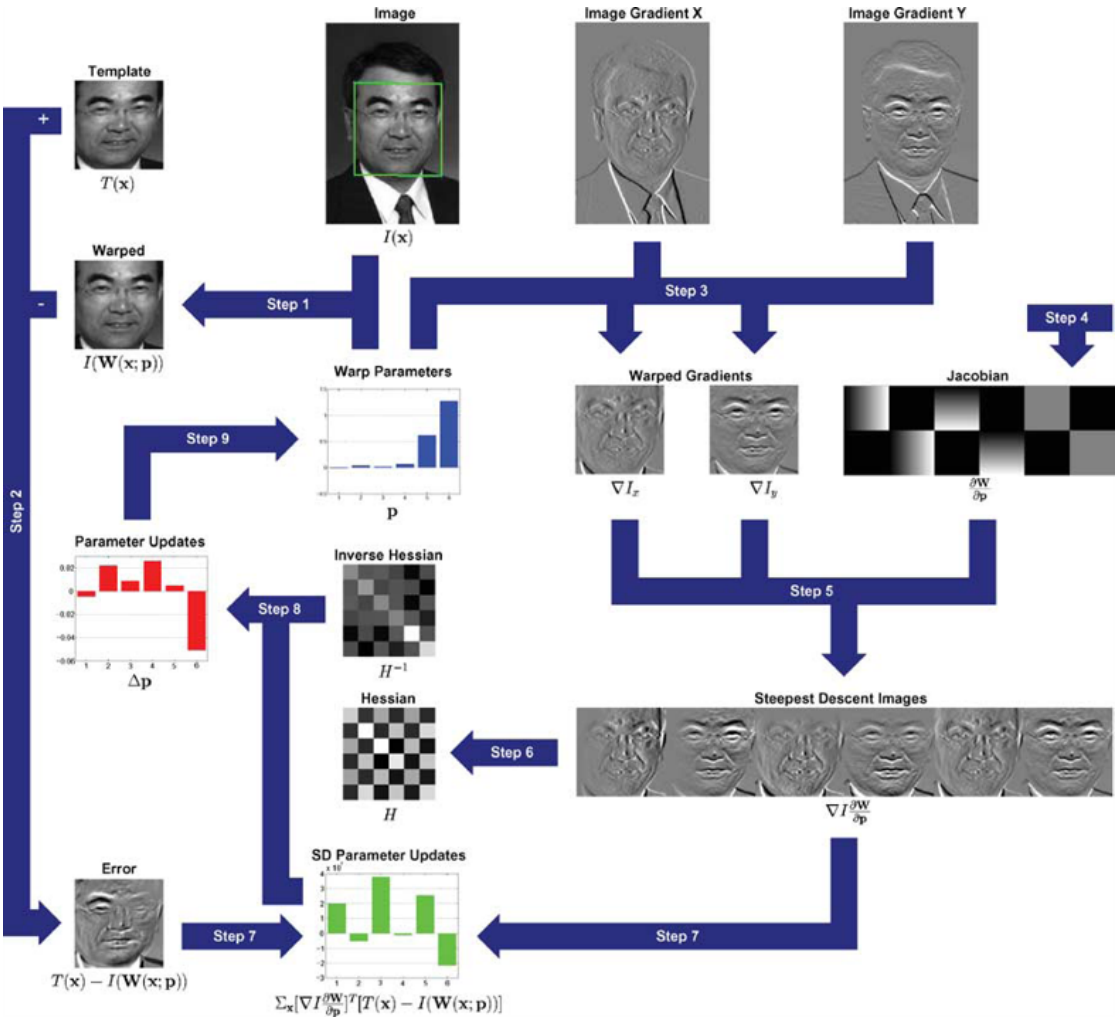
  Compute steepest descent images $\nabla I\frac{\partial W}{\partial P}$

  Compute Hessian matrix $\sum(\nabla I\frac{\partial W}{\partial P})^T(\nabla I\frac{\partial W}{\partial P})$

  Compute $\sum(\nabla I\frac{\partial W}{\partial P})^T(T(x,y) - I(W([x,y];P)))$

  Compute ΔP

  Update $P \longleftarrow P + \Delta P$
- Baker & Matthews, IJCV'04: Lucas-Kanade 20 Years On: A Unifying Framework



- Good
  - It is a beautiful framework
  - It can handle different parameter space
  - It can converge fast in a high-frame rate video
- Bad
  - Not robust to image noise or large displacement
  - Some transformations are impossible to parameterize

What if we don’t have a parameterized warping?

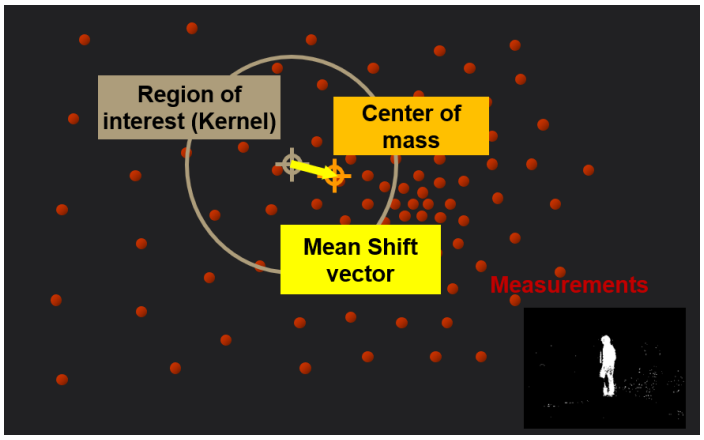
Mean Shift Method

M* aximize similarity between tracked and target regions through evolution towards higher density in a parameter space

- It can handle non-parameterized distribution through sampling
- A mean (center) location is iteratively updated by moving it to the centroid of pixels within a chosen radius

Mode Seeking by Mean Shift

- Only Samples of the Distribution



- Typically this search only takes a few iterations
- Initialize multiple means and pick the location where many converges

Choice of Feature Space

- Color pixels
- Grayscale
- Gradients

Tracking by Detection

Tracking by Features

Template Detection

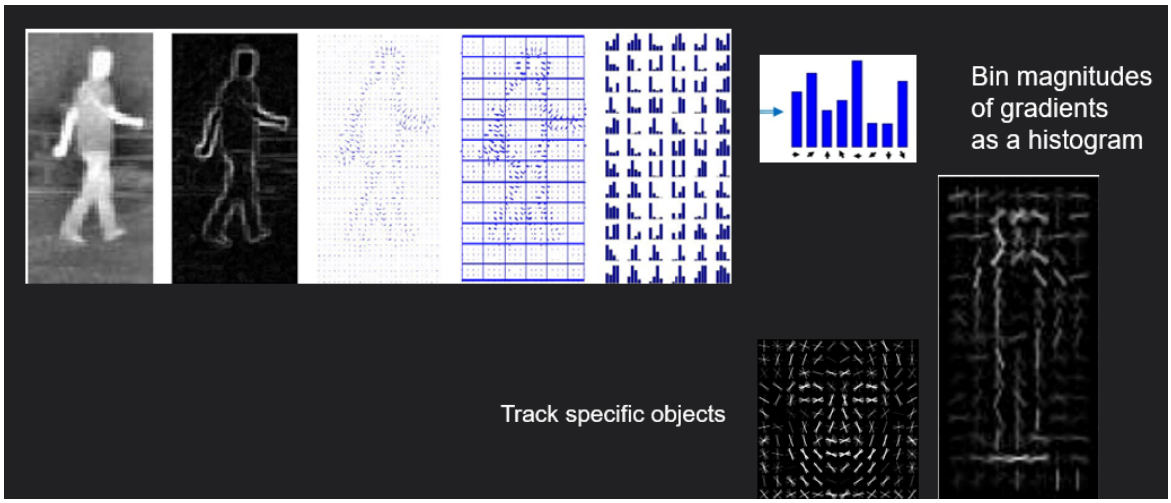
Detect Key Points

Invariant to scale, rotation, or perspective

Build Feature Descriptors

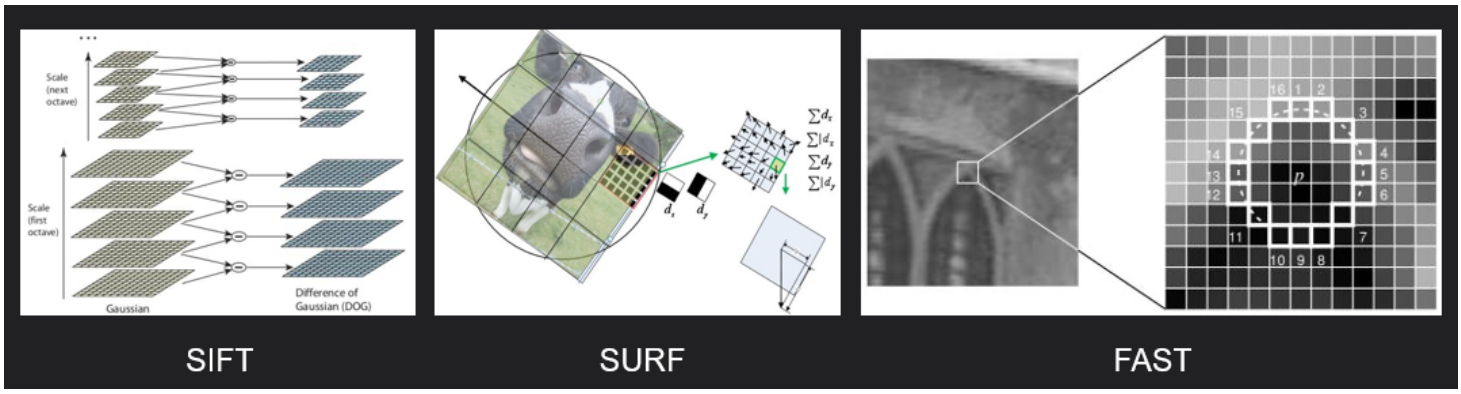
Example: Histogram of Oriented Gradients

HOG is a (rotation invariant) feature descriptor

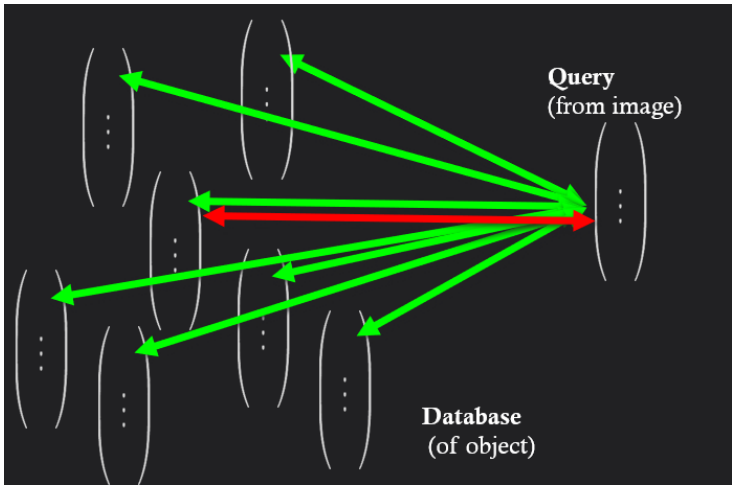


- Object shapes defined by edges, thus HOG over entire objects can be descriptive

Other Features



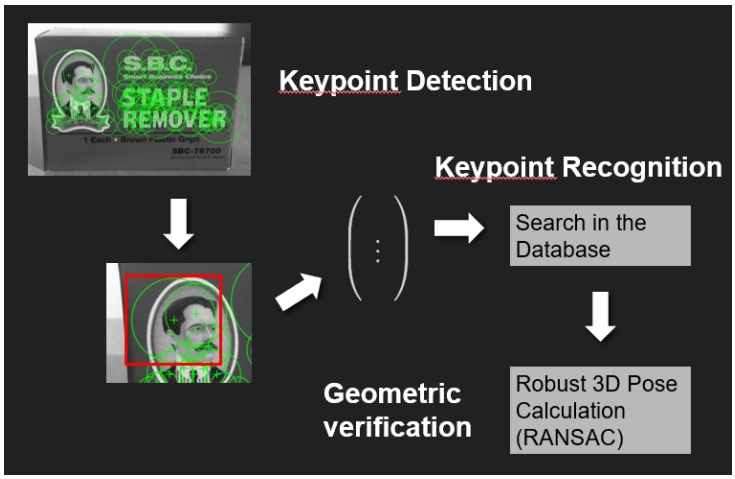
Match Keypoint Descriptors



- This can be accelerated in multiple ways
  - Tree structure (KD-tree)
  - Approximate nearest neighbors
  - Hashing
  - Parallel implementation

Outlier Elimination

Summary



Track Objects of a Known Category
Track by Detection

- Detect object(s) independently in each frame
- Associate detections over time into tracks

Multiple Objects
Detecting Objects

- Usually turn to supervised learning
- Modern detectors based on ConvNets can be both accurate and fast

Associating Objects

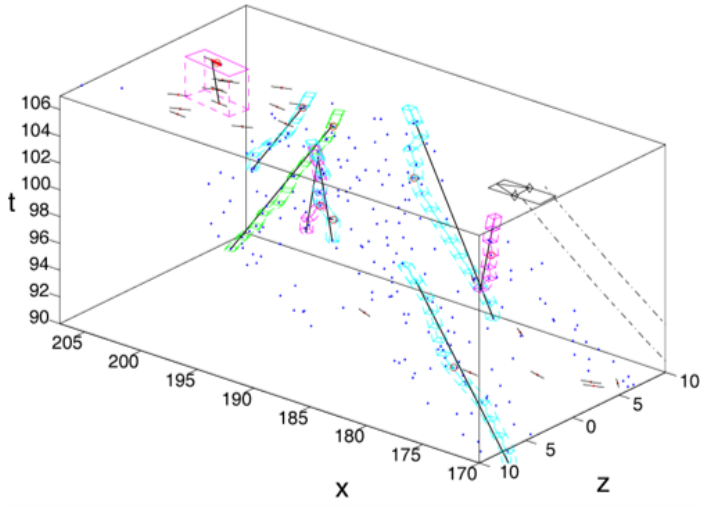
- There are primarily two ways of associating objects
    1. Learn an appearance model to measure the similarity directly
    2. Predict the motion of the objects in the first image
- Object association is not resolved. It is receiving a lot of attention recently because of the increased popularity of video analysis. Recent years, both methods see tremendous progress. We will talk more about those trends in the deep learning part.

Discriminative learning

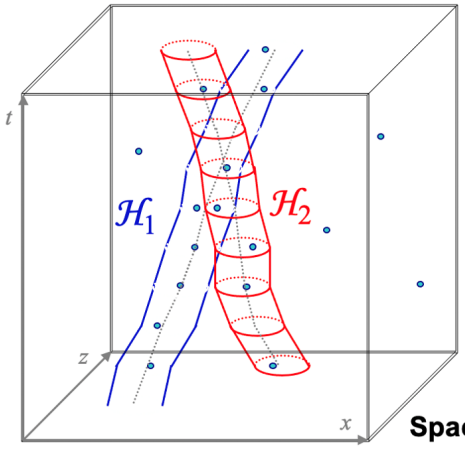
- KNN – K-nearest neighbors
- Logistic regression (of binary class)
- Decision trees - random forests
- SVM
- ...
- Everything is deep learning now

Space-Time Analysis

- Collect detections in space-time volume



Trajectory Estimation



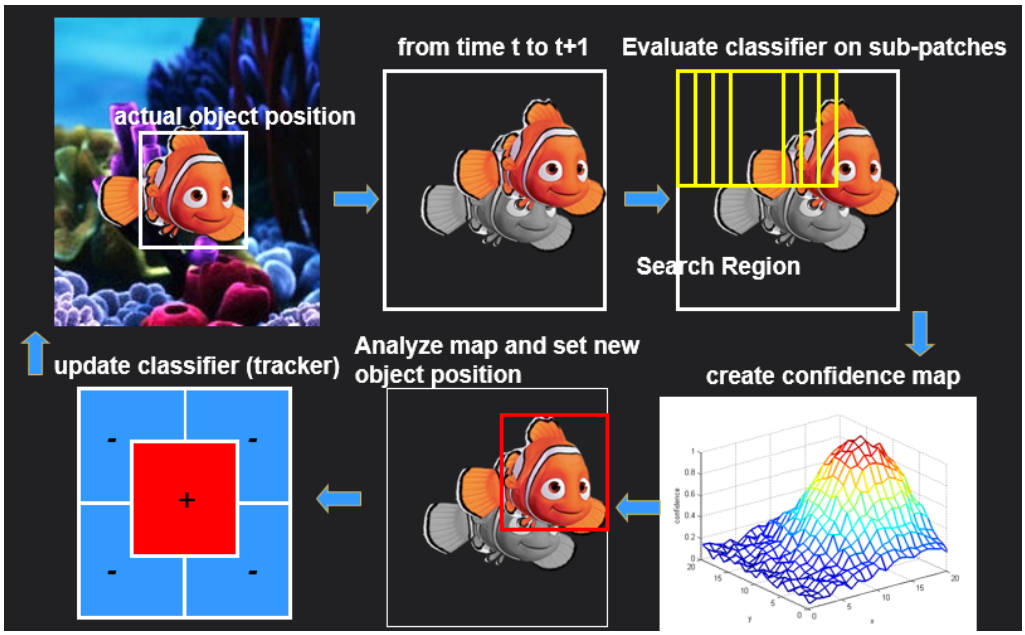
Space Time Volume

Online Learning

Online Learning of Appearance Model

Learning current object appearance vs. local background.

Tracking Loop



To Avoid Drift

- Only thing we are sure about the object is its initial model (e.g. appearance in first frame)
- We can “anchor” / correct our model with this information, in order to help avoid drift

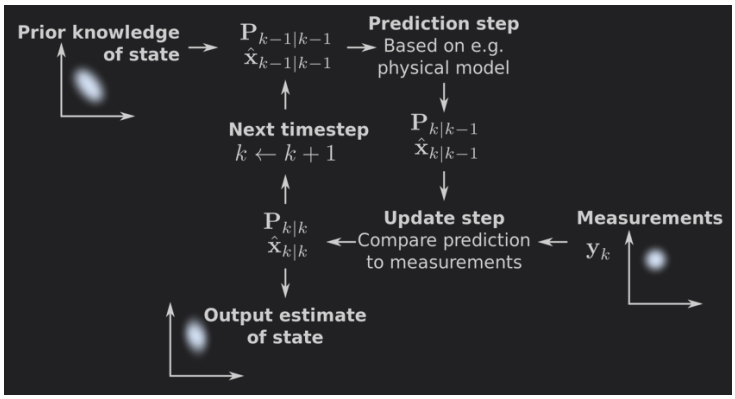
Location Information

A simple constant-velocity heuristics can go a long way, especially when the camera is stationary.

More Complicated Motion

- Temporal Filtering/Predictions
  - To predict location
  - To reduce noise
  - To disambiguate multiple objects

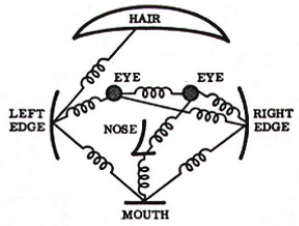
Kalman Filter



Tracking Articulated Objects

Articulated Tracking: Part-Based Models

- Intuitive model of an object
- Model has two components
  1. parts (2D image fragments)
  2. structure (configuration of parts)
- Dates back to Fischler & Elschlager 1973

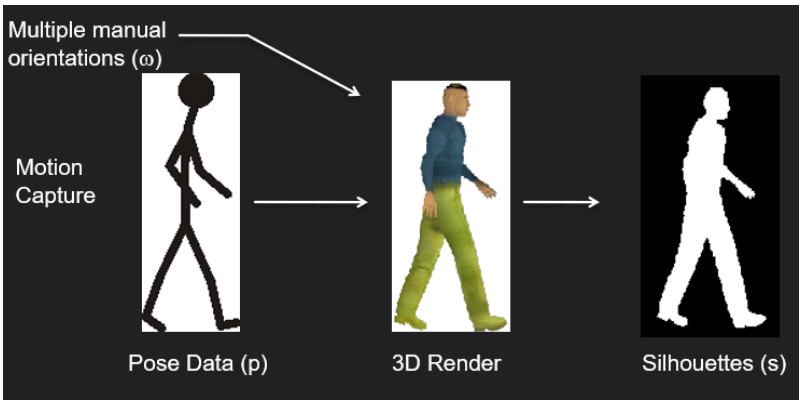


Temporal Information

- What temporal info can we use for tracking?
- What variation would we expect in population?

Articulation Space

- Tracking Articulated Motion as High-Dimensional Inference
  - Walking cycles have one main (periodic) DOF
  - Regressors to learn this (latent) space, and its variation (Gaussian Process regression, PCA, etc)
  - (Pose, Silhouette) training data can be obtained by 3D rendering



Articulation Tracking