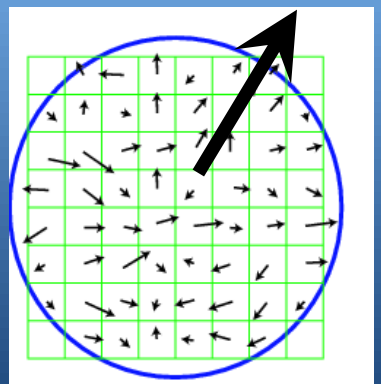
计算机视觉

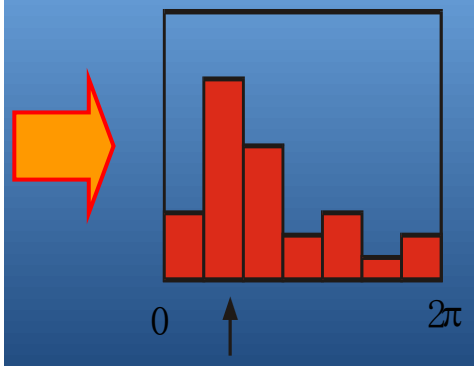
Computer Vision

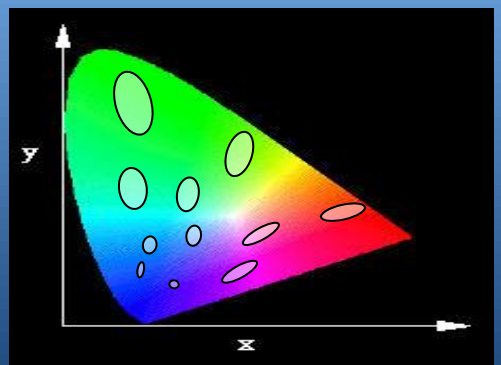
Lecture 6: Local Image Features 2

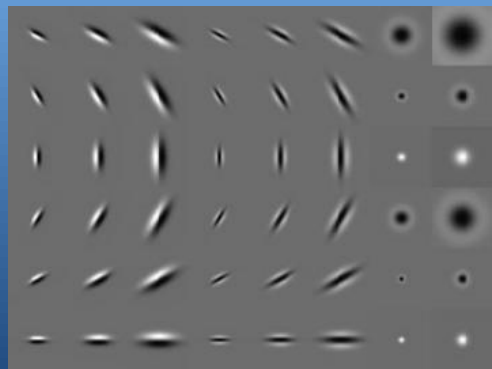
张 超

信息科学技术学院 智能科学系









Local Image Features

- Properties of detectors
  - Edge detectors
  - Corners
  - Blobs
  - Scale invariant detection
- Properties of descriptors
  - HOG
  - SIFT
  - Color
  - Texture
  - Shape context

Local Image Features

- Properties of detectors
  - Edge detectors
  - Corners
  - Blobs
  - Scale invariant detection
- Properties of descriptors
  - HOG
  - SIFT
  - Color
  - Texture
  - Shape context

HOG： Histograms of Oriented Gradients

Recall: Edges

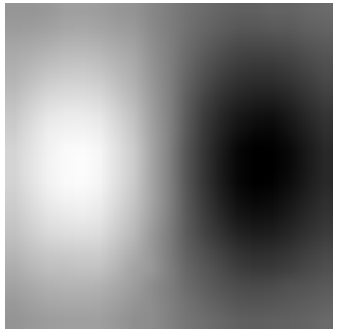
- Idea:
  - points where image value change very sharply are important
    - changes in surface reflectance
    - shadow boundaries
    - outlines
- Finding Edges:
  - Estimate gradient magnitude using appropriate smoothing
  - Mark points where gradient magnitude is
    - Locally biggest and
    - big

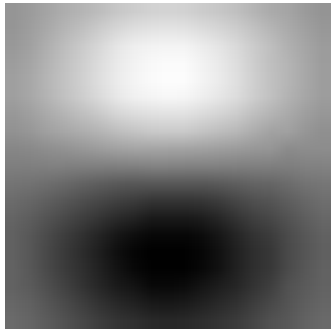
Recall: Smoothed gradients

- Fact: These two are the same
  - Smooth, then differentiate
  - Filter with derivative of Gaussian

$$\frac{\partial (G_\sigma * *I)}{\partial x} = (\frac{\partial G_\sigma}{\partial x}) * *I$$

- Exploit:
  - Filter image with derivative of Gaussian filters to get smoothed gradient





Edge Maps Depend on Shading

- If the image is brighter (resp. darker)
  - because the camera gain is higher (resp. lower)
  - because there is more (resp. less) light
  - because the pixel values got multiplied by a constant

  Then the gradient magnitude is bigger (resp. smaller)

- So scaling image brightness changes the edge map
  - because some magnitudes will go above (resp. below) the test threshold

- Edge maps differ for brighter/darker copies of a picture

Orientations - I

- Gradient magnitude is affected by illumination changes
  – but gradient direction isn't

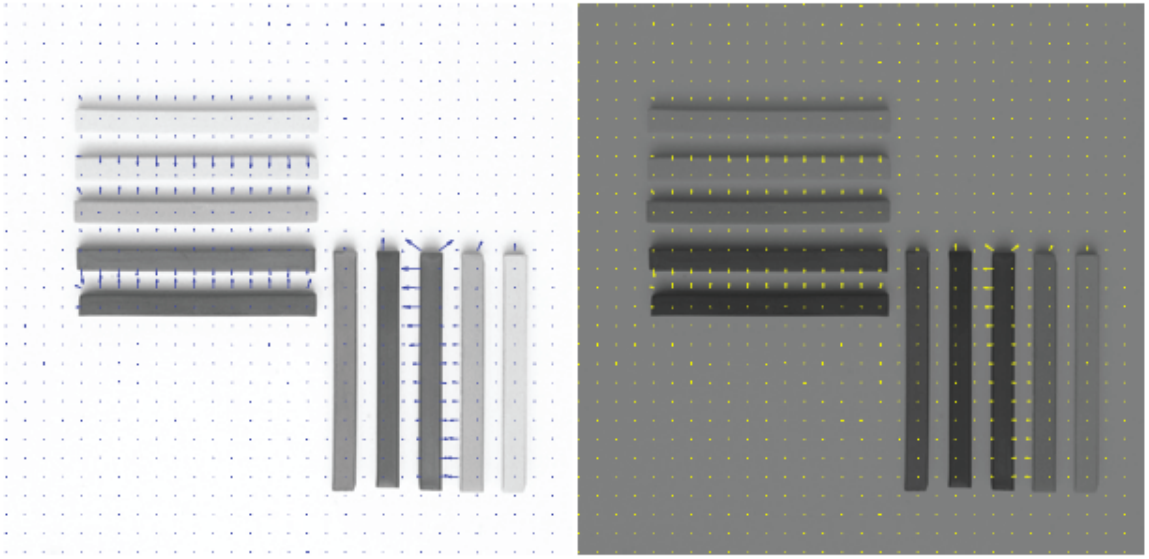


FIGURE 5.7: The magnitude of the image gradient changes when one increases or decreases the intensity. The orientation of the image gradient does not change; we have plotted every 10th orientation arrow, to make the figure easier to read. Note how the directions of the gradient arrows are fixed, whereas the size changes. *Philip Gatward © Dorling Kindersley, used with permission.*

Orientations - II

- Notice larger gradients are “better”
  - we know the orientation better; associated image points “more interesting”

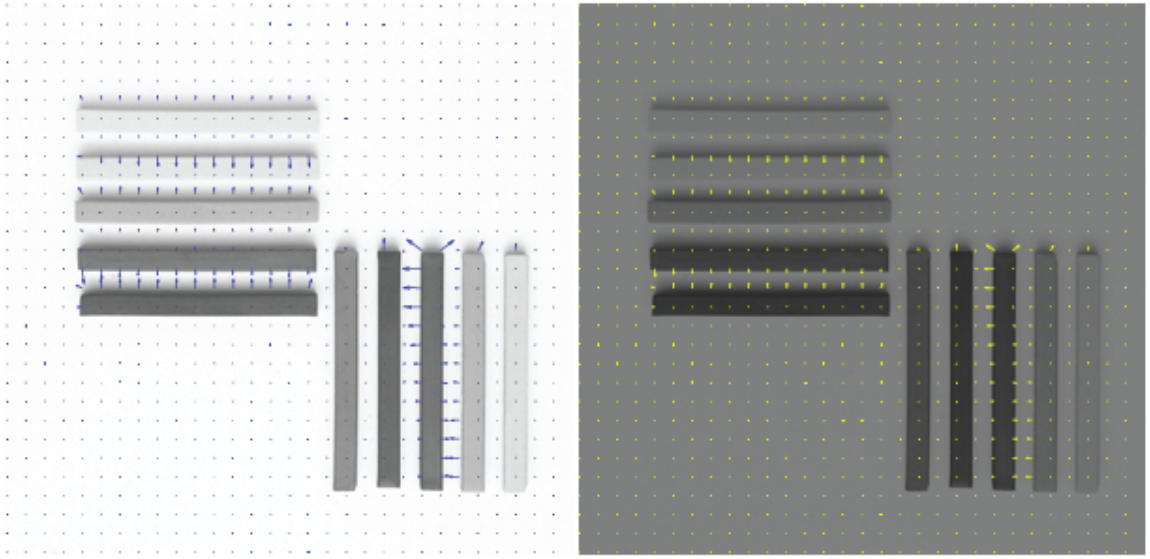


FIGURE 5.7: The magnitude of the image gradient changes when one increases or decreases the intensity. The orientation of the image gradient does not change; we have plotted every 10th orientation arrow, to make the figure easier to read. Note how the directions of the gradient arrows are fixed, whereas the size changes. *Philip Gatward © Dorling Kindersley, used with permission.*

Orientations at different scales

Rose plot

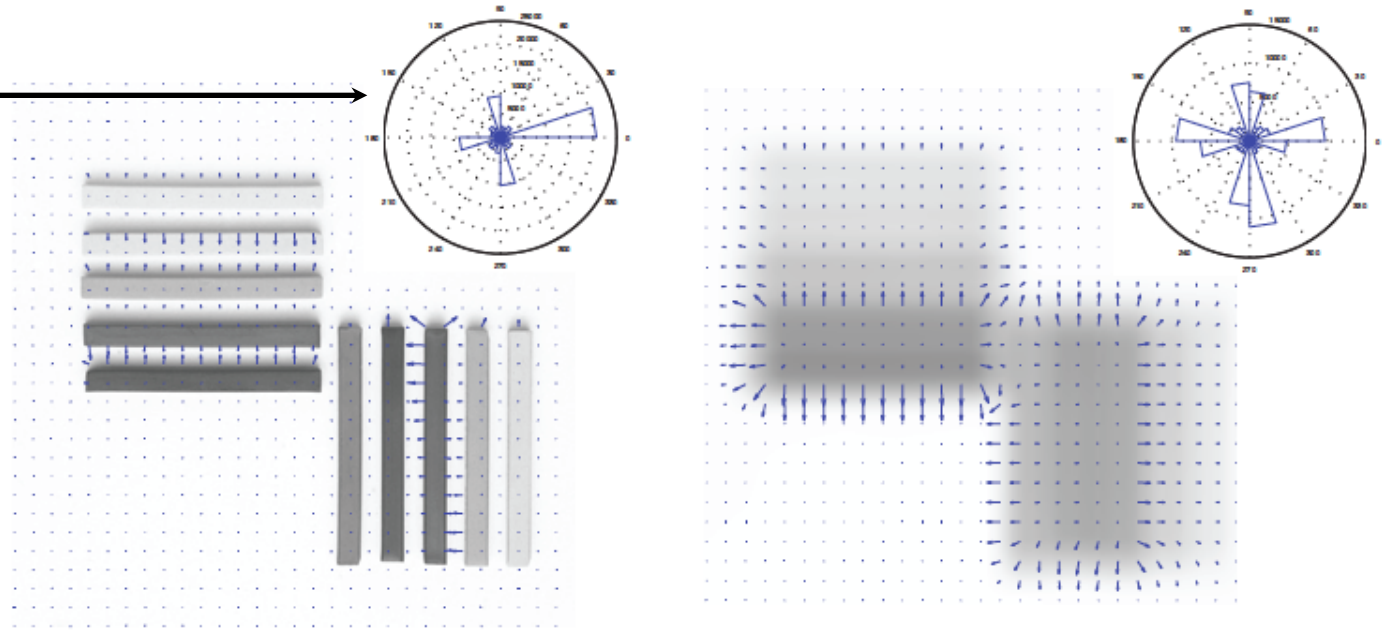


FIGURE 5.8: The scale at which one takes the gradient affects the orientation field. We show the overall trend of the orientation field by plotting a rose plot, where the size of a wedge represents the relative frequency of that range of orientations. **Left** shows an image of artists pastels at a fairly fine scale; here the edges are sharp, and so only a small set of orientations occurs. In the heavily smoothed version on the **right**, all edges are blurred and corners become smooth and blobby; as a result, more orientations appear in the rose plot. *Philip Gatward © Dorling Kindersley, used with permission.*

Orientation Histograms Vary

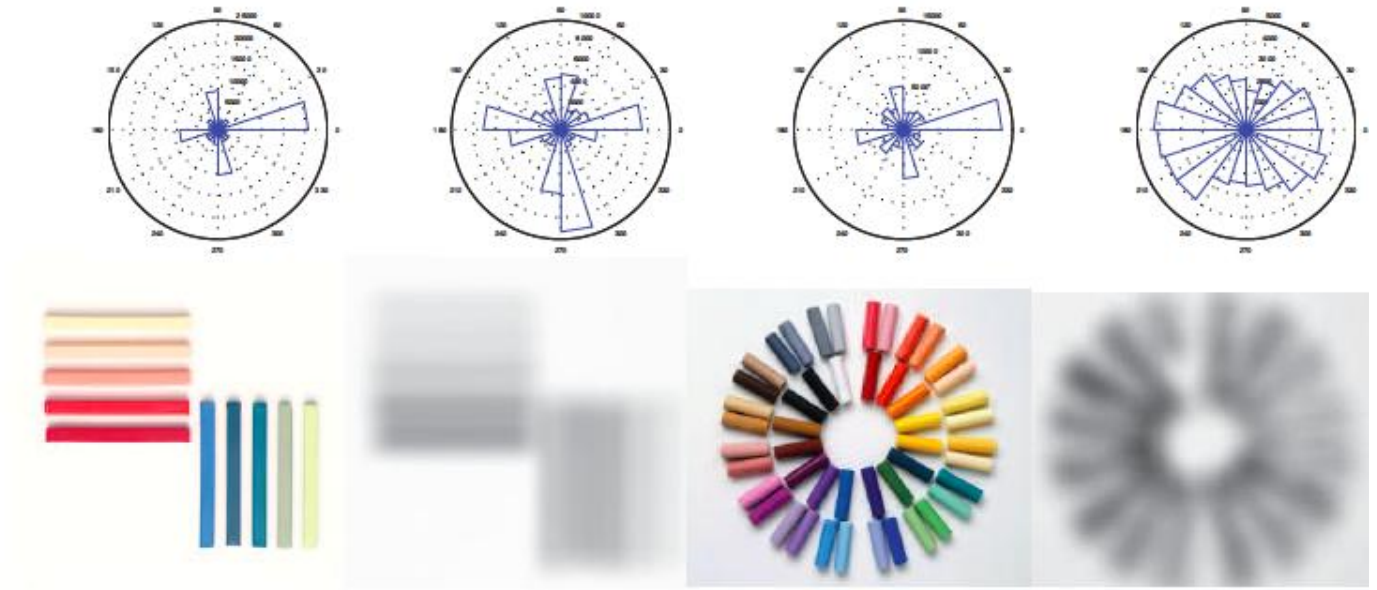


FIGURE 5.9: Different patterns have quite different orientation histograms. The **left** shows rose plots and images for a picture of artists pastels at two different scales; the **right** shows rose plots and images for a set of pastels arranged into a circular pattern. Notice how the pattern of orientations at a particular scale, and also the changes across scales, are quite different for these two very different patterns. *Philip Gatward © Dorling Kindersley, used with permission.*

Building Orientation Representations

- We would like to represent a pattern in an image patch
  - to detect things in images
  - to match points in one image to corresponding points in another image
- Necessary properties
  - we have to know which patch to describe
  - think of this as knowing the center and size of an image window
- Desirable features
  - representation doesn't change much if the center is slightly wrong
  - representation doesn't change much if the size is slightly wrong
  - representation is distinctive
  - representation doesn't change much if the patch gets brighter/darker
  - large gradients are more important than small gradients

Histograms of Oriented Gradients

- Necessary properties
  - we have to know which patch to describe
  - think of this as knowing the center and size of an image window

For the moment, assume window is known

- Desirable features
  - representation doesn't change much if the center is slightly wrong
  - representation doesn't change much if the size is slightly wrong
  - representation is distinctive
  - representation doesn't change much if the patch gets brighter/darker
  - large gradients are more important than small gradients

Use histograms

Use orientations

Break window into boxes, describe each separately

Weight orientation histogram entries

Histograms of Oriented Gradients

- Strategy:
  - break patch up into blocks
  - construct histogram representing gradient orientations in that block
    - which won't change much if the patch moves slightly
    - entries weighted by magnitude

- Variants
  - histogram of angles
  - histogram of gradient vectors, length normalized by block averages

HOG features

Given a grid cell $\mathcal{G}$ for patch with center $\boldsymbol{c} = (x_c, y_c)$ and radius r

Create an orientation histogram
For each point $\boldsymbol{p}$ in an $m \times m$ subgrid spanning $\mathcal{G}$
  Compute a gradient estimate $\nabla \mathcal{I}\,|_{\boldsymbol{p}}$ estimate at $\boldsymbol{p}$
    as a weighted average of $\nabla \mathcal{I}$, using bilinear weights centered at $\boldsymbol{p}$.
  Add a vote with weight $\| \nabla \mathcal{I} \| \frac{1}{r\sqrt{2\pi}} \exp\left(-\frac{\|\boldsymbol{p}-\boldsymbol{c}\|^2}{r^2} \right)$
    to the orientation histogram cell for the orientation of $\nabla \mathcal{I}$.

Algorithm 5.5: Computing a Weighted q Element Histogram for a SIFT Feature.

HOG features

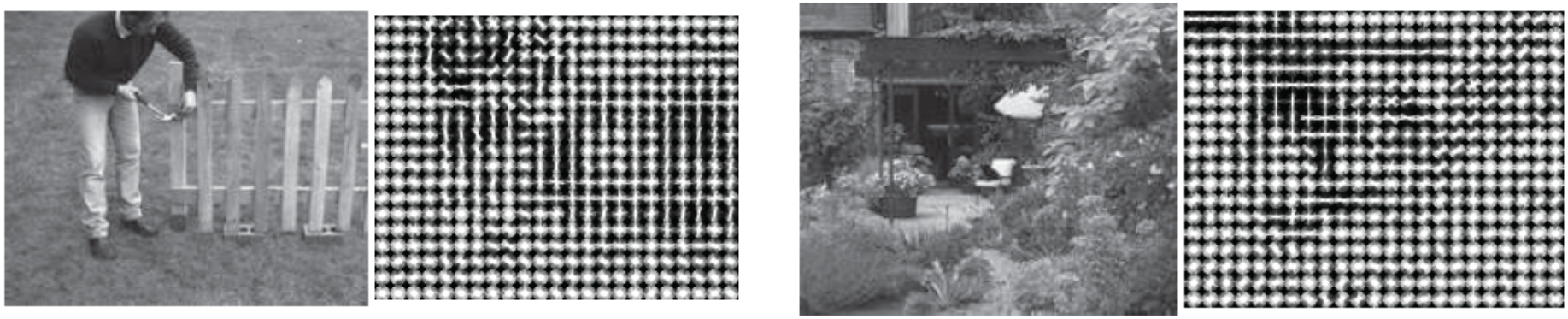
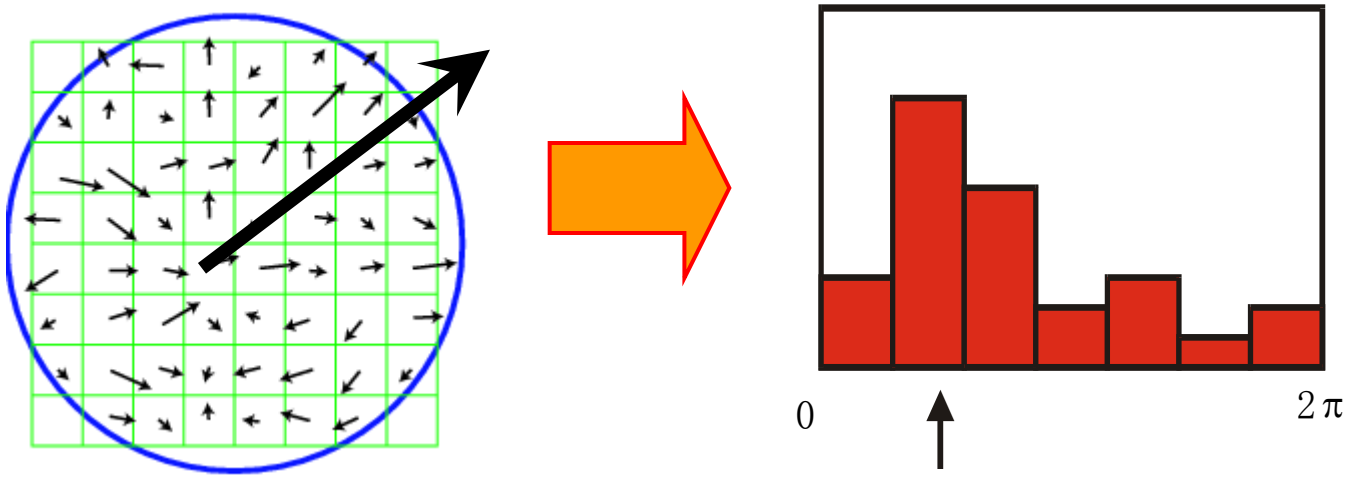


FIGURE 5.15: The HOG features for each the two images shown here have been visualized by a version of the rose diagram of Figures 5.7–5.9. Here each of the cells in which the histogram is taken is plotted with a little rose in it; the direction plotted is at right angles to the gradient, so you should visualize the overlaid line segments as edge directions. Notice that in the textured regions the edge directions are fairly uniformly distributed, but strong contours (the gardener, the fence on the **left**; the vertical edges of the french windows on the **right**) are very clear. This figure was plotted using the toolbox of Dollár and Rabaud. *Left:* © *Dorling Kindersley, used with permission. Right: Geoff Brightling* © *Dorling Kindersley, used with permission.*

Eliminating rotation ambiguity

- To assign a unique orientation to circular image windows:
  - Create histogram of local gradient directions in the patch
  - Assign canonical orientation at peak of smoothed histogram



HOG - Crucial Points

- Gradient orientations are not affected by intensity
- Orientations with larger magnitude are more important
- Describe an image window of known location, size
  - Histograms reduce the effect of poor estimate of location, size
  - Break window into subwindows
    - for each, compute an orientation histogram, weighting orientations by magnitude
  - Numerous variants available

Local Image Features

- Properties of detectors
  - Edge detectors
  - Corners
  - Blobs
  - Scale invariant detection
- Properties of descriptors
  - HOG
  - SIFT
  - Color
  - Texture
  - Shape context

SIFT： Scale-Invariant Feature Transform

Scale-Invariant Feature Transform

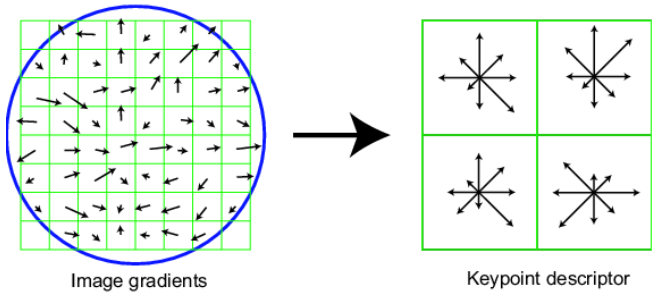
- Generates image features, “keypoints”
  - invariant to image scaling and rotation
  - partially invariant to change in illumination and 3D camera viewpoint
  - many can be extracted from typical images
  - highly distinctive

SIFT [Lowe]

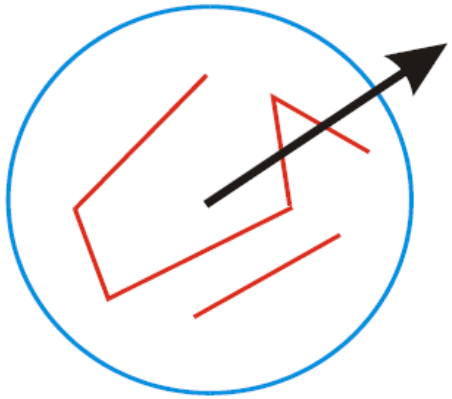
SIFT Algorithm Stages

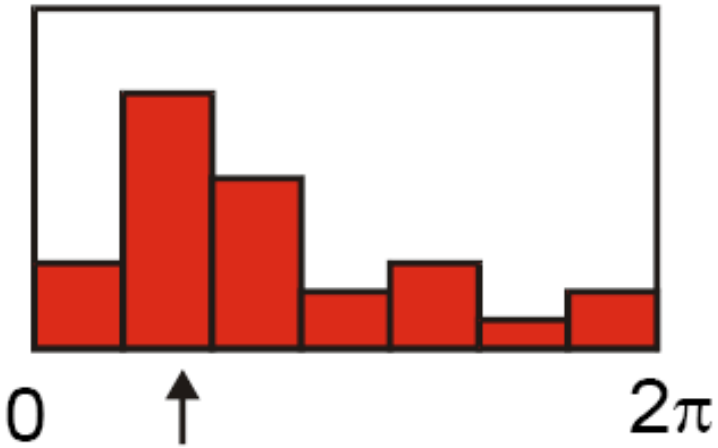
- **Scale-space Extrema Detection**
  - Uses difference-of-Gaussian function
- **Keypoint Localization**
  - Sub-pixel location and scale fit to a model
- **Orientation assignment**
  - 1 or more for each keypoint
- **Keypoint descriptor**
  - Created from local image gradients

Orientation Assignment



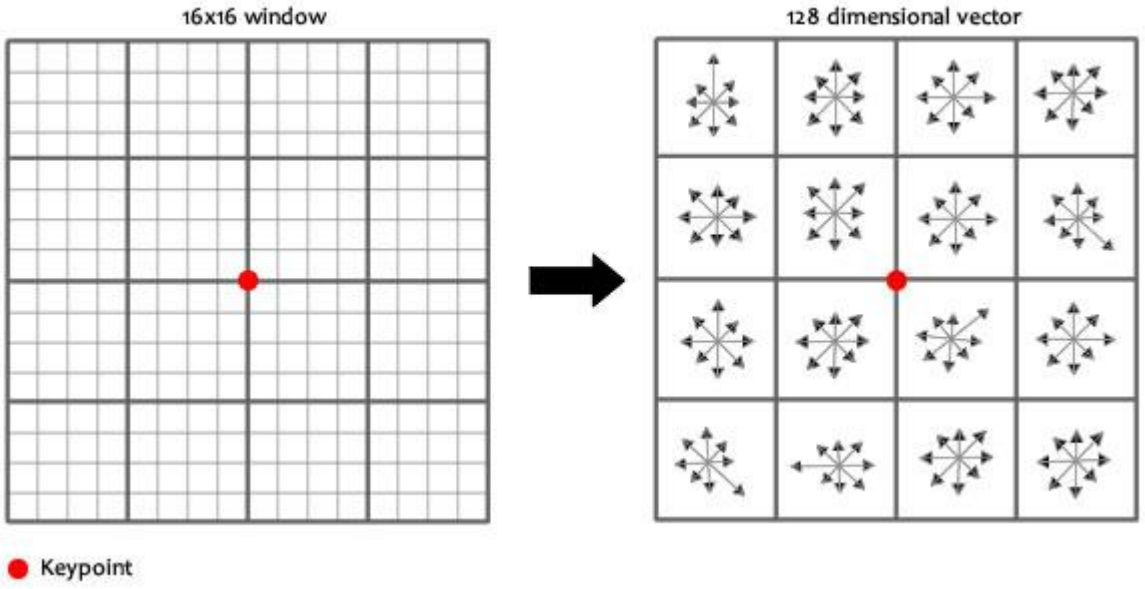
- Create histogram of local gradient directions computed at selected scale
- Assign canonical orientation at peak of smoothed histogram
- Each keypoint specifies stable 2D coordinates (x, y, scale, orientation)





SIFT vector formation

- Thresholded image gradients are sampled over 16x16 array of locations in scale space
- Create array of orientation histograms
- 8 orientations x 4x4 histogram array = 128 dimensions



SIFT Review

- Generates image features, “keypoints”
  - invariant to image scaling and rotation
  - partially invariant to change in illumination and 3D camera viewpoint
  - many can be extracted from typical images
- Each “keypoint” has an associated **descriptor** that is
  - Relative to keypoint orientation and scale
  - Is robust to small affine transformations.

Local Image Features

- Properties of detectors
  - Edge detectors
  - Corners
  - Blobs
  - Scale invariant detection
- Properties of descriptors
  - HOG
  - SIFT
  - Color
  - Texture
  - Shape context

Color

What is color?

- Color is a psychological property of our visual experiences when we look at objects and lights,
  not a physical property of those objects or lights
  (S. Palmer, *Vision Science: Photons to Phenomenology*)

- Color is the result of interaction between physical light in the environment and our visual system

Overview of Color

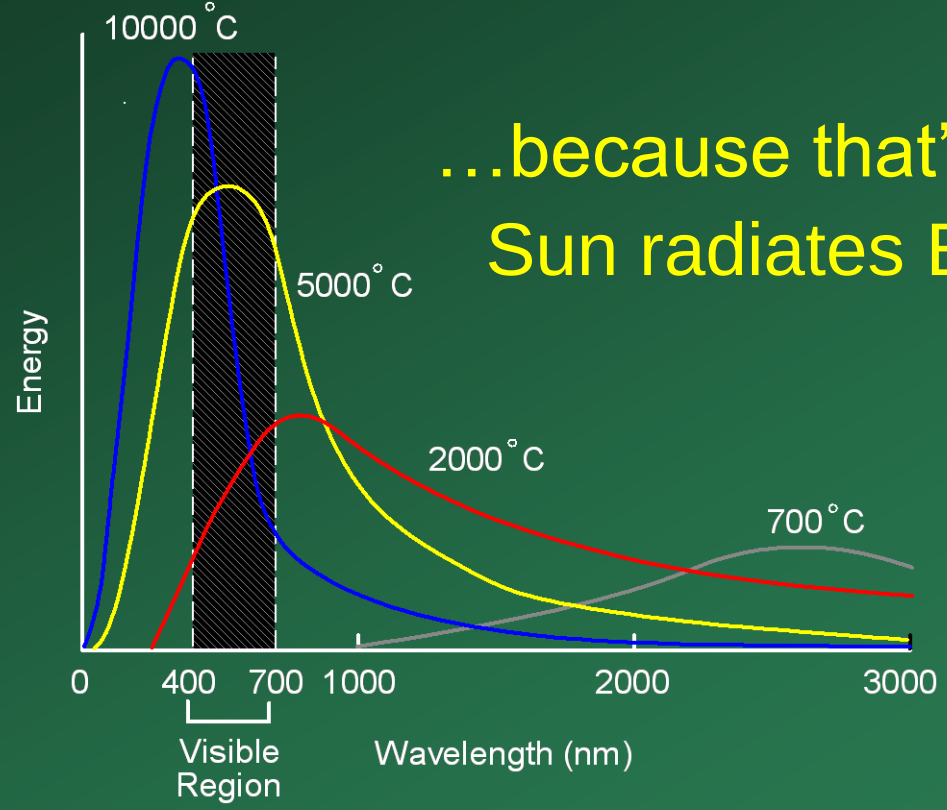
- Physics of color
- Human encoding of color
- Color spaces
- Inference from Color

Visible Light

Plank's law for Blackbody radiation

Surface of the sun: ~5800K

Why do we see light of these wavelengths?



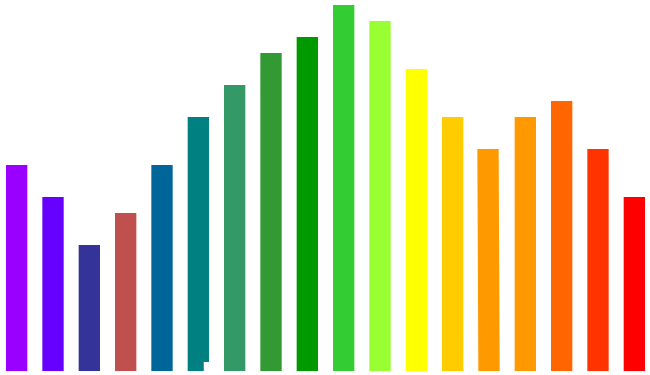
…because that's where the Sun radiates EM energy

Stephen E. Palmer, 2002

The Physics of Light

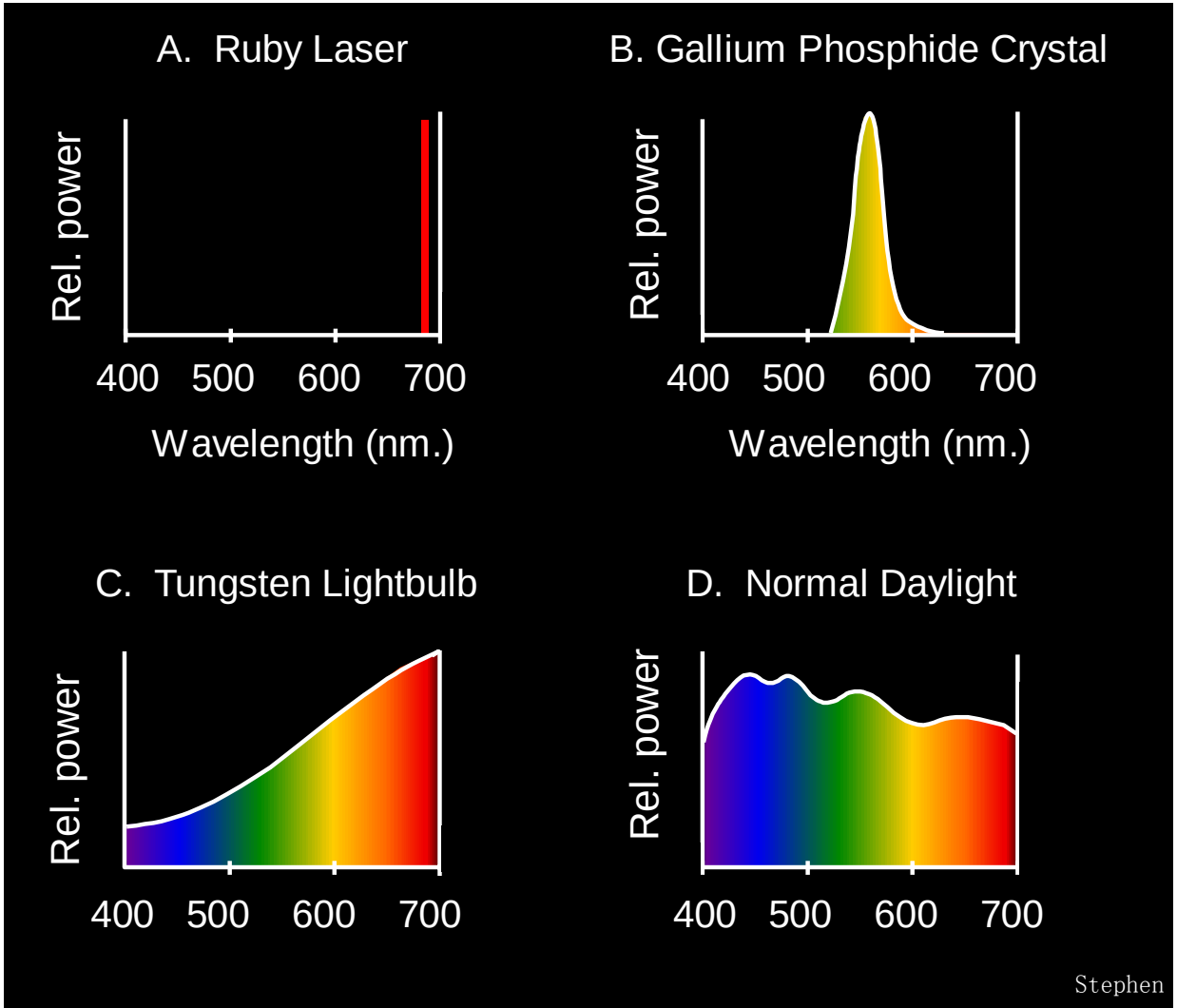
Any source of light can be completely described physically by its spectrum: the amount of energy emitted (per time unit) at each wavelength 400 - 700 nm.

Relative spectral power



The Physics of Light

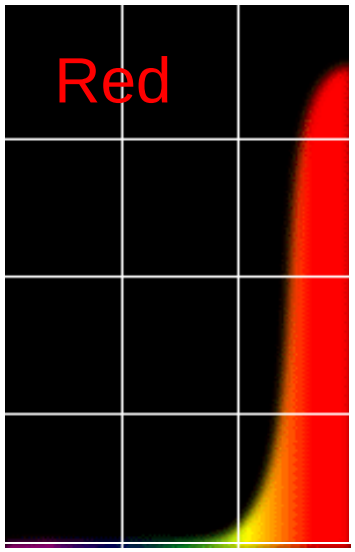
Some examples of the spectra of light sources

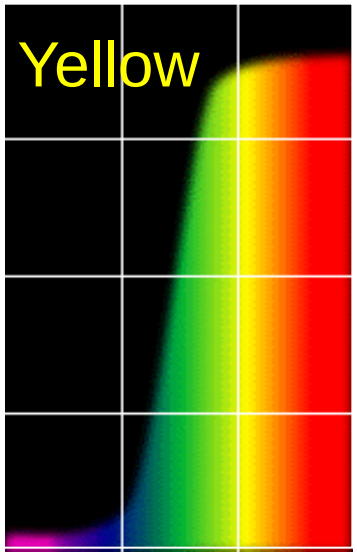


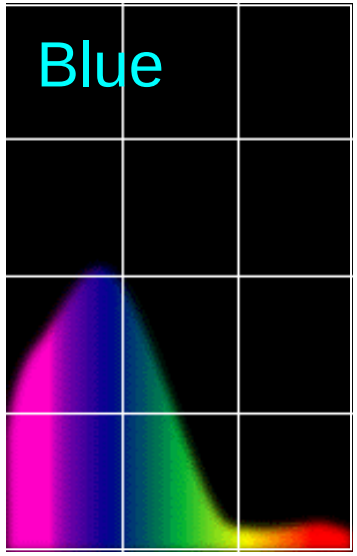
The Physics of Light

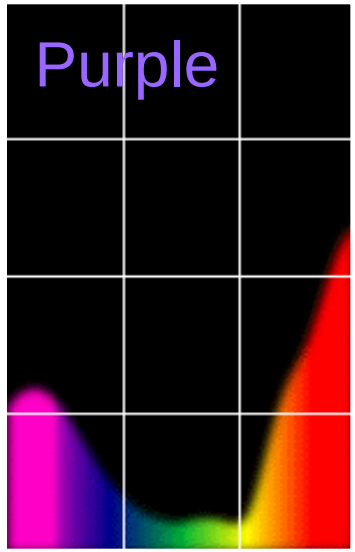
Some examples of the reflectance spectra of surfaces



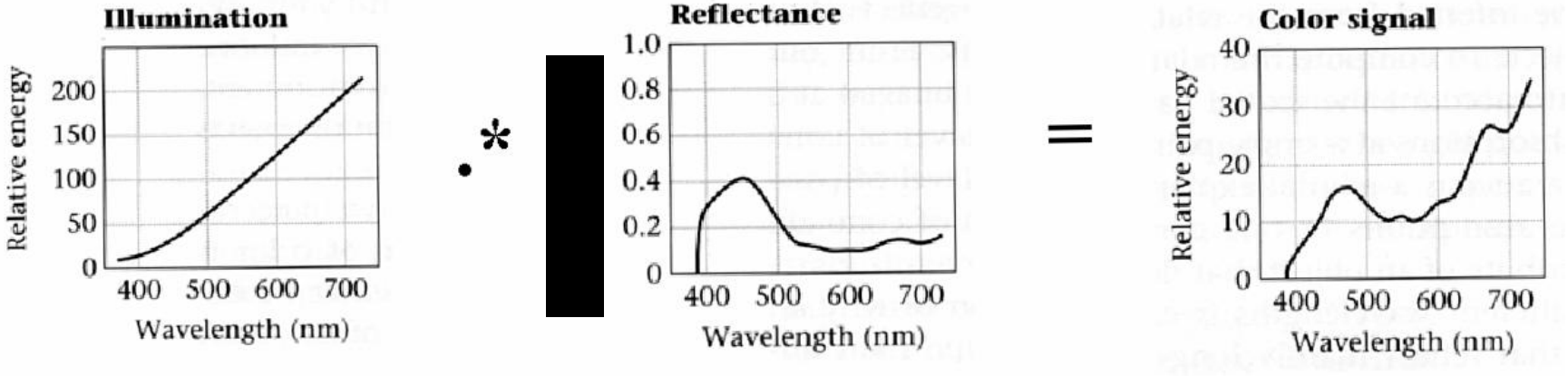






Interaction of light and surfaces

- Reflected color is the result of interaction of light source spectrum with surface reflectance
- Spectral radiometry
  - All definitions and units are now "per unit wavelength"
  - All terms are now "spectral"



From Foundation of Vision by Brian Wandell, Sinauer Associates, 1995

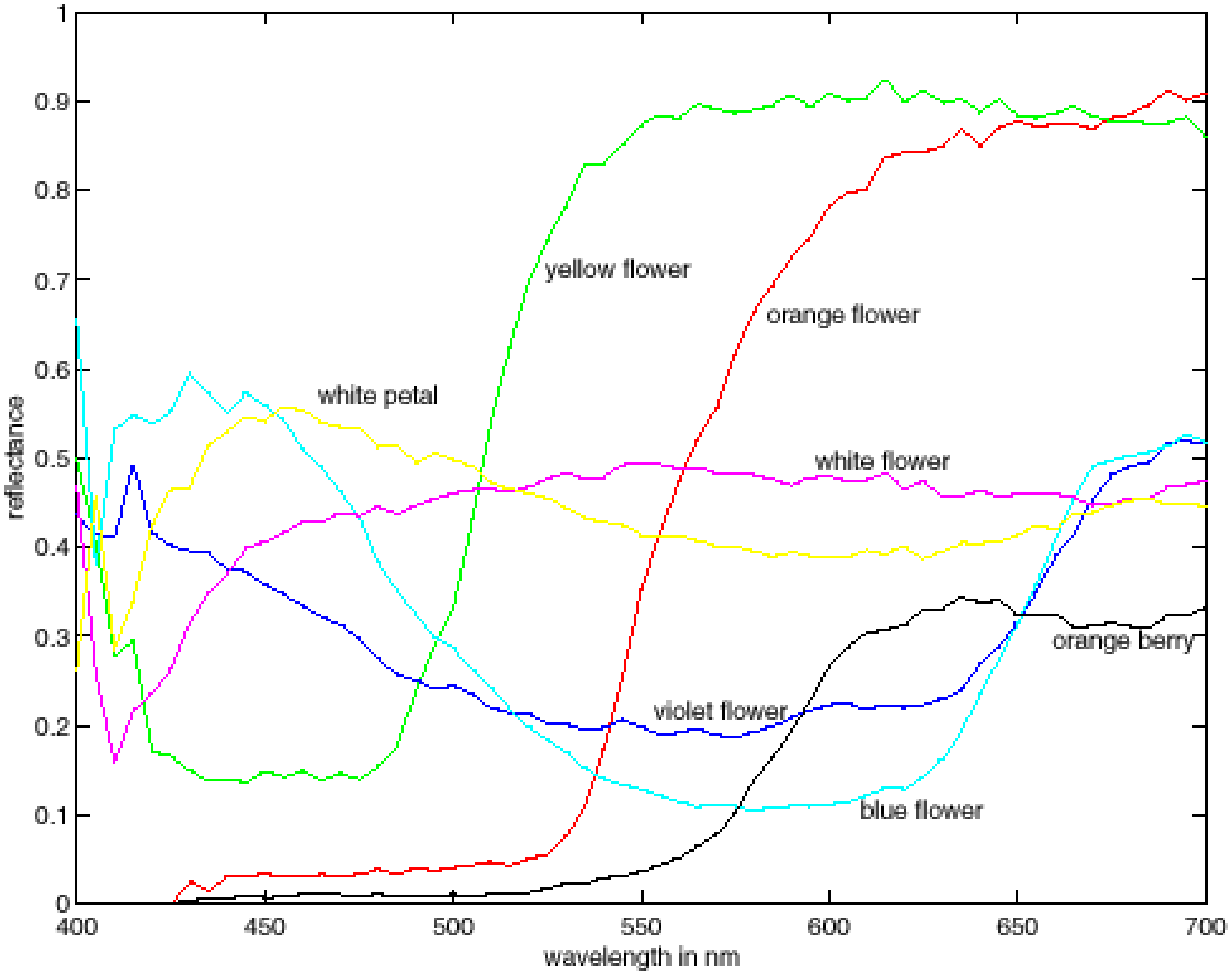
Interaction of light and surfaces

- What is the observed color of any surface under monochromatic light?

Olafur Eliasson, *Room for one color*

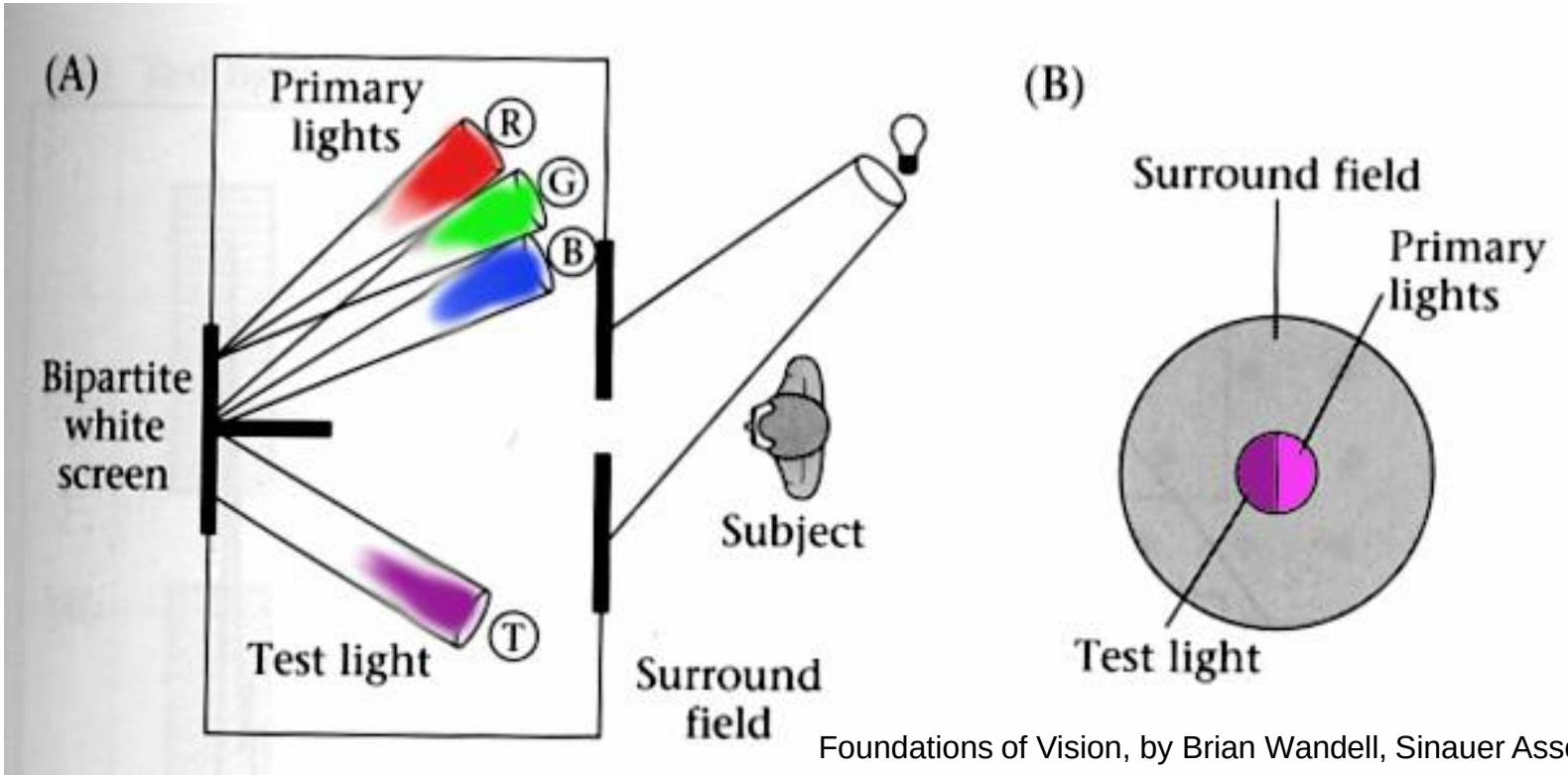
Slide by S. Lazebnik

Spectra of some real-world surfaces



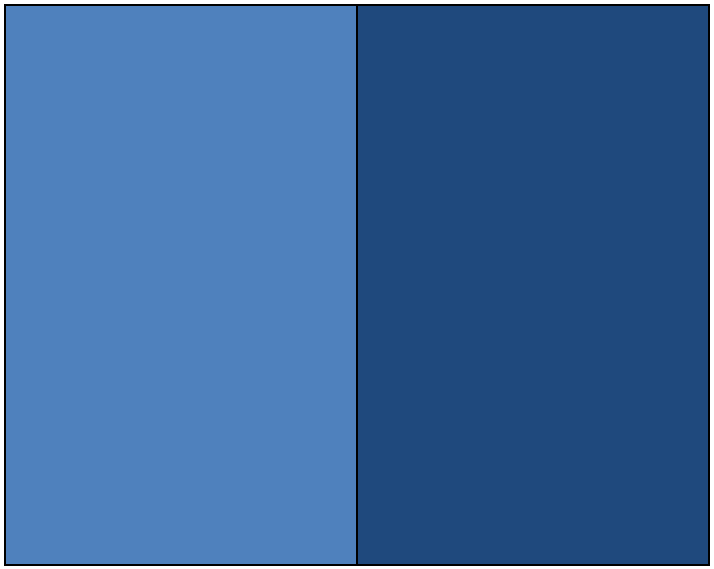
Standardizing color experience

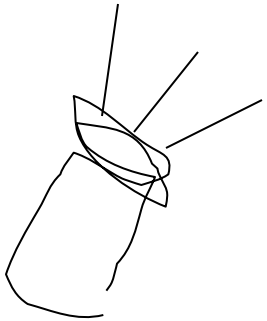
- We would like to understand which spectra produce the same color sensation in people under similar viewing conditions
- Color matching experiments



Foundations of Vision, by Brian Wandell, Sinauer Assoc., 1995

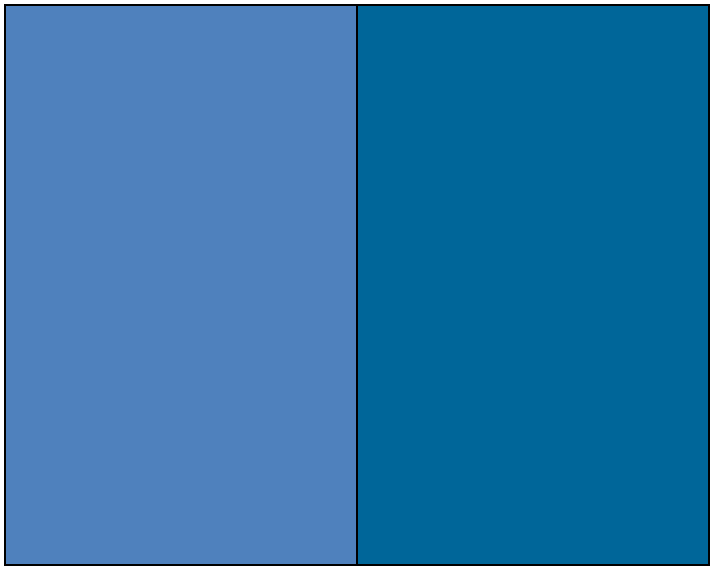
Color matching experiment 1

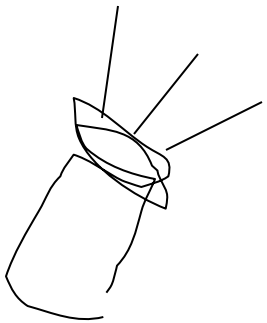


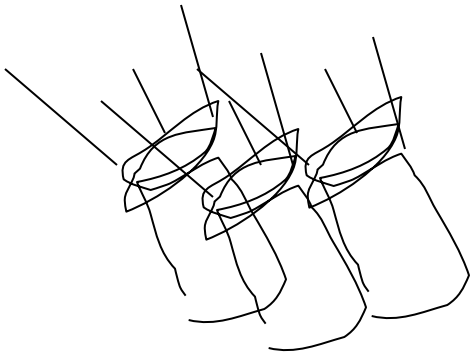


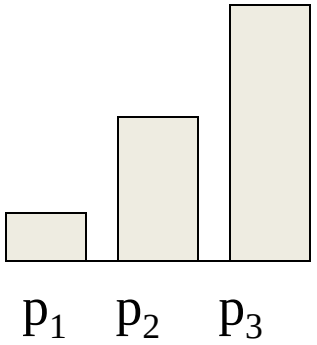
Source: W. Freeman

Color matching experiment 1



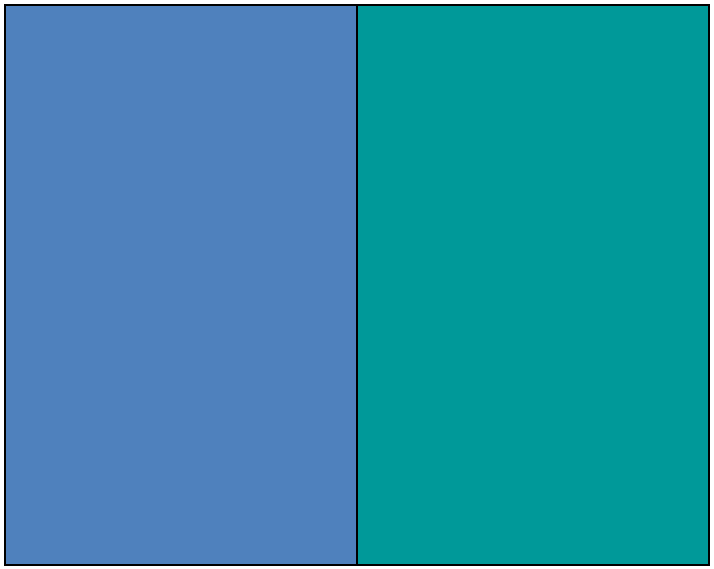


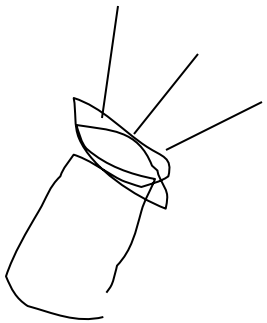


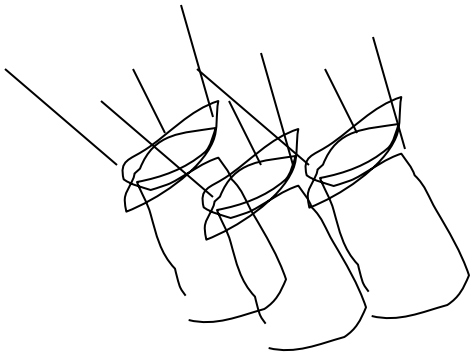


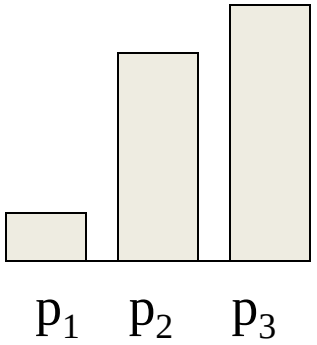
Source: W. Freeman

Color matching experiment 1



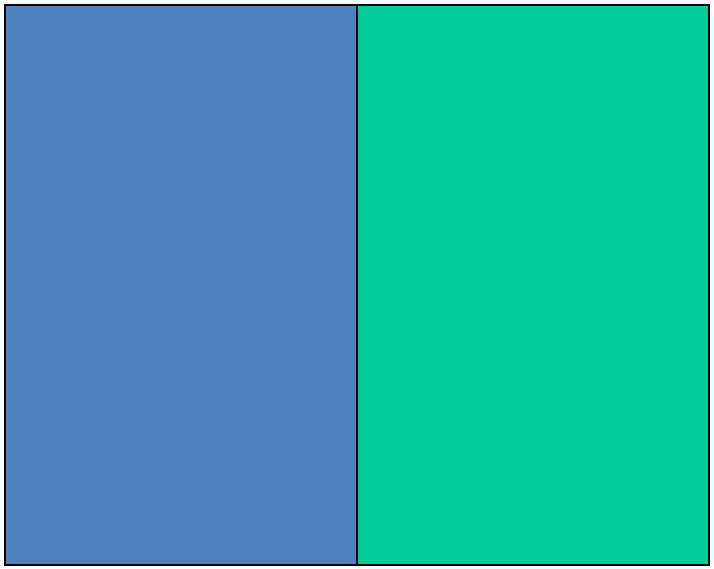




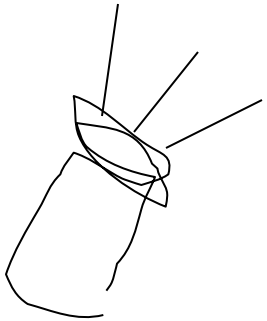


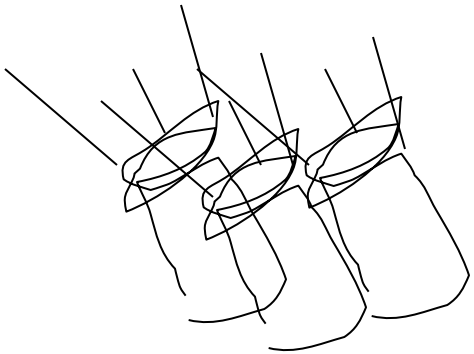
Source: W. Freeman

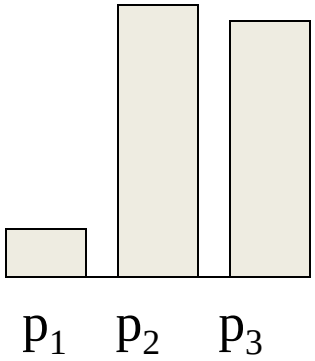
Color matching experiment 1



The primary color amounts needed for a match

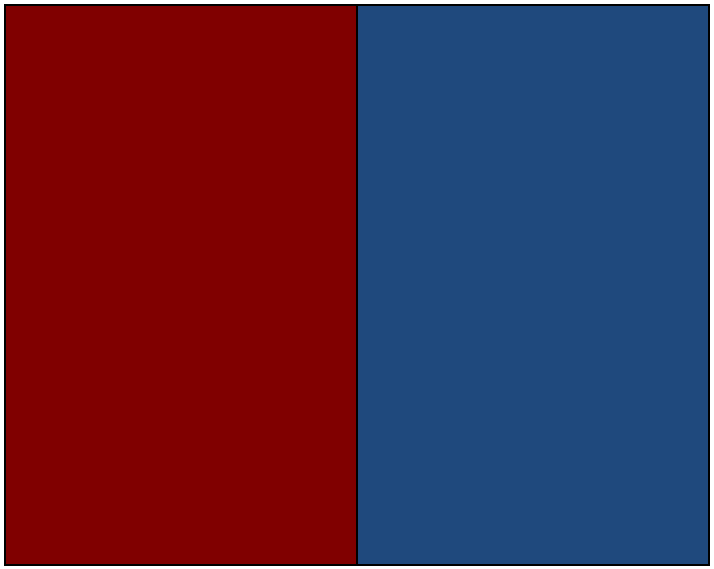


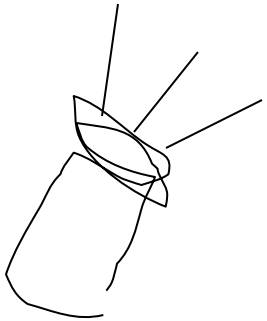




Source: W. Freeman

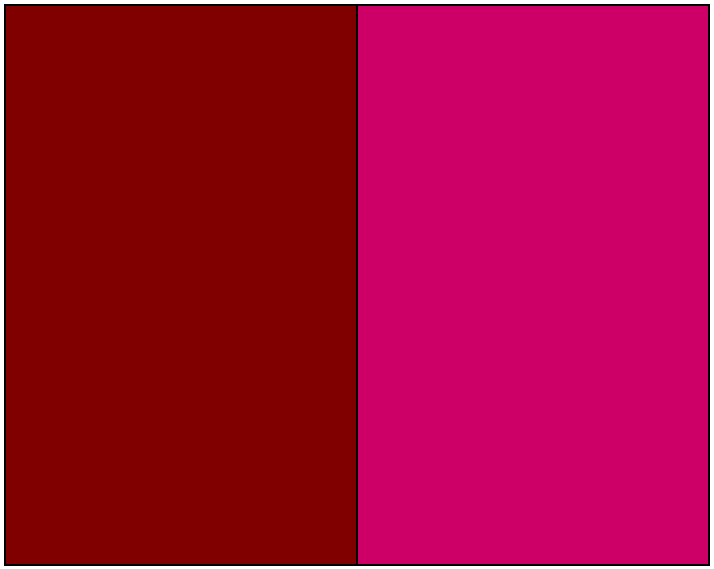
Color matching experiment 2

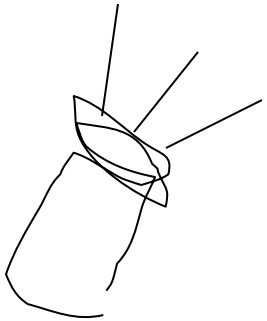


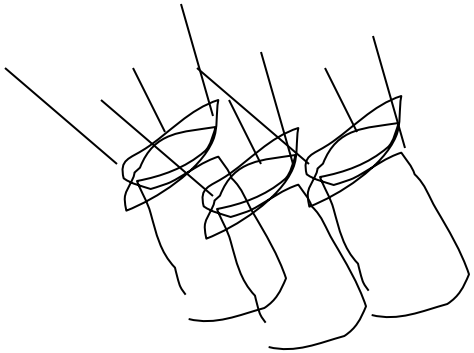


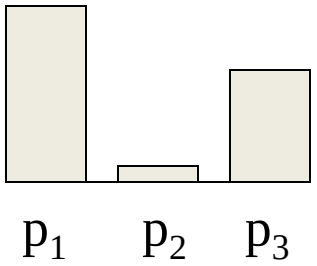
Source: W. Freeman

Color matching experiment 2



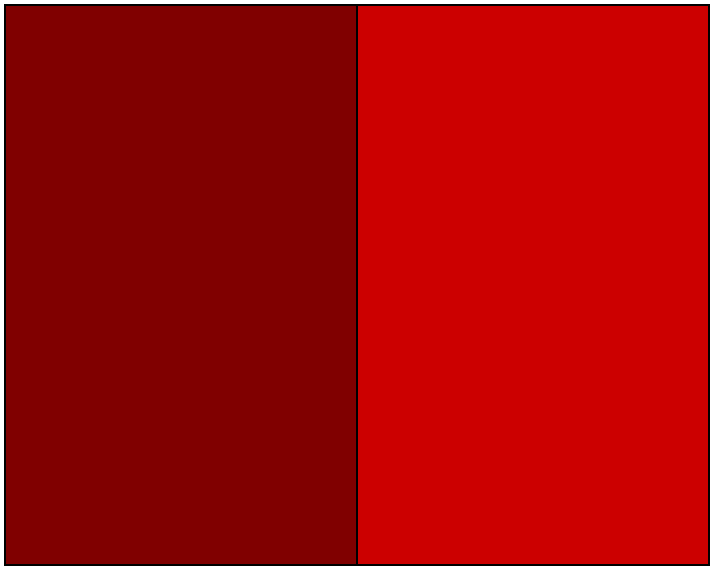


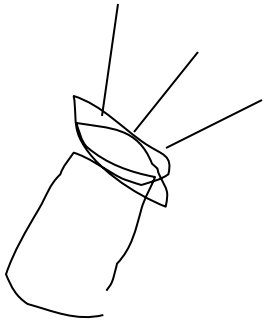


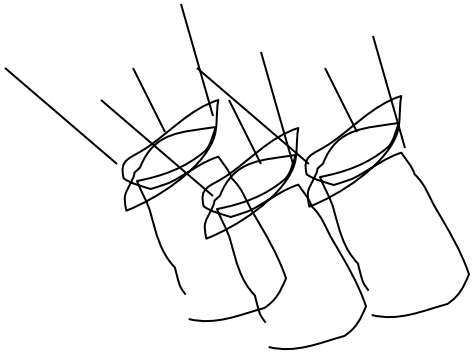


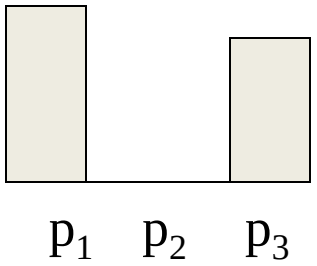
Source: W. Freeman

Color matching experiment 2





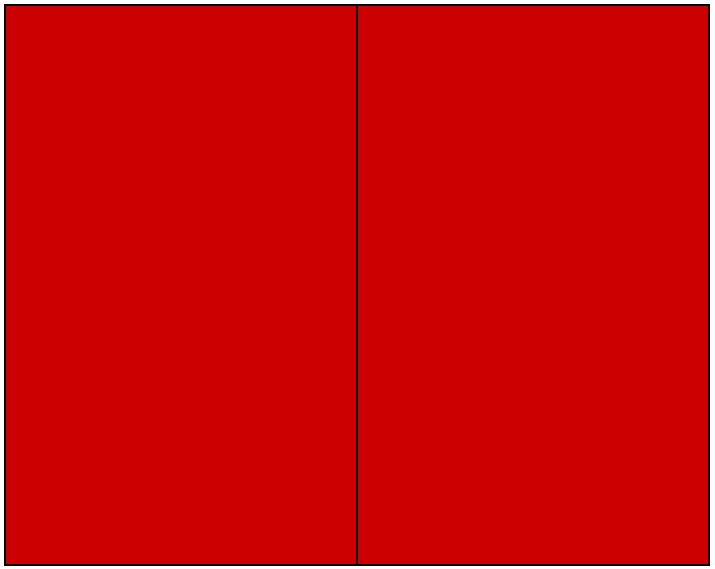




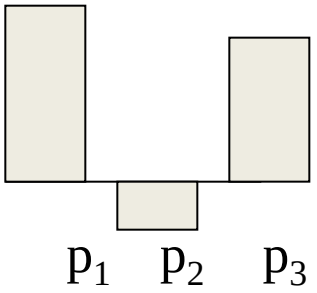
Source: W. Freeman

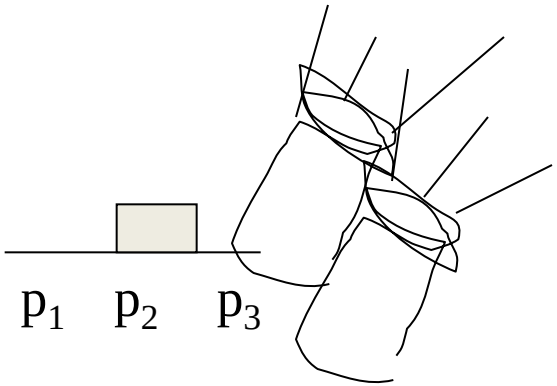
Color matching experiment 2

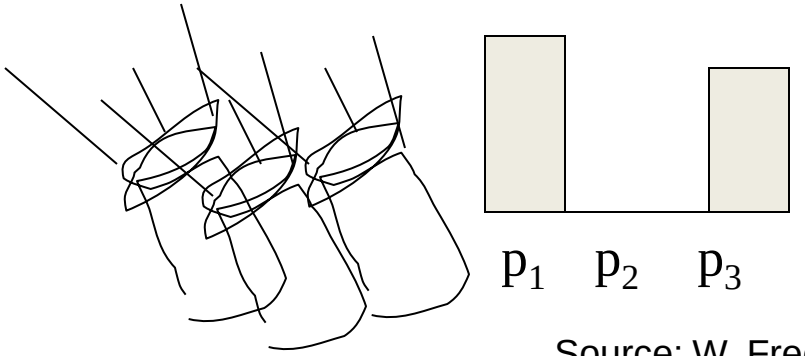
We say a "negative" amount of p_2 was needed to make the match, because we added it to the test color's side.



The primary color amounts needed for a match:







Source: W. Freeman

Trichromacy

- In color matching experiments, most people can match any given light with three primaries
  - Primaries must be *independent*
- For the same light and same primaries, most people select the same weights
  - Exception: color blindness
- Trichromatic color theory
  - Three numbers seem to be sufficient for encoding color
  - Dates back to 18th century (Thomas Young)

Grassman's Laws

- Color matching appears to be linear

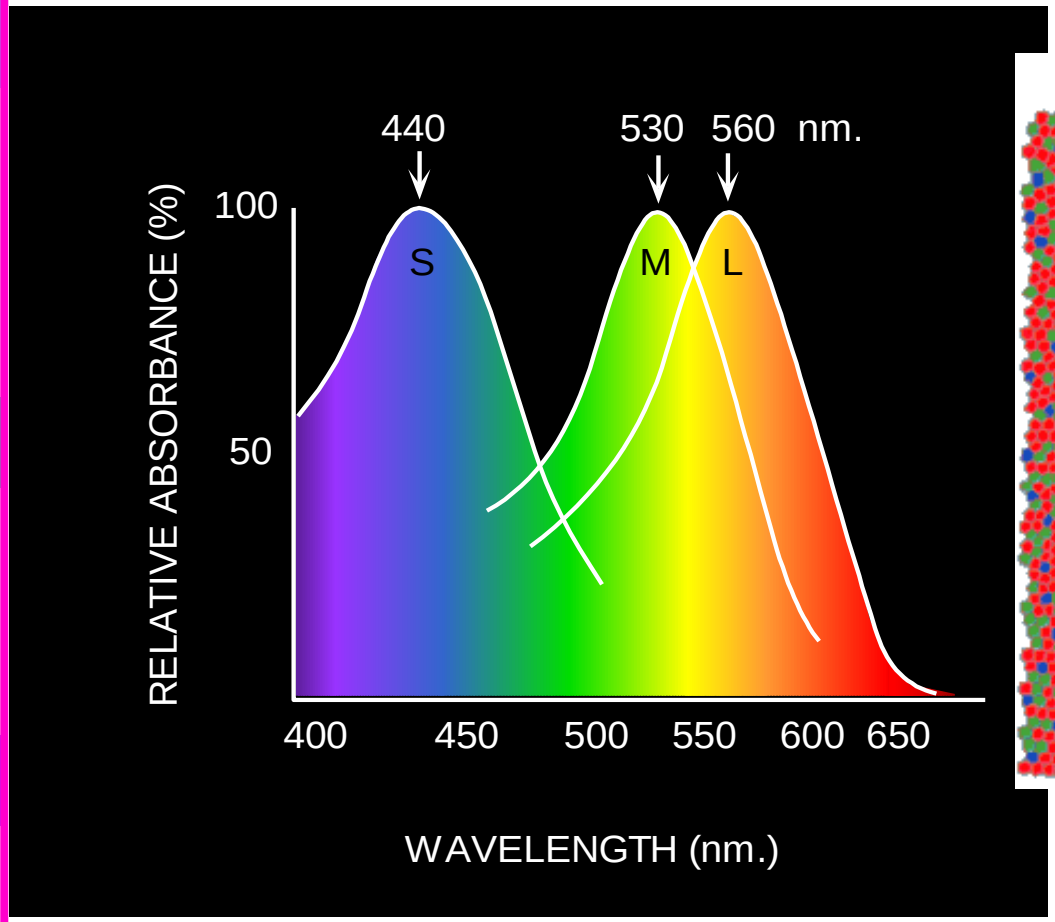
- If two test lights can be matched with the same set of weights, then they match each other:
  - Suppose $A = u_1 P_1 + u_2 P_2 + u_3 P_3$ and $B = u_1 P_1 + u_2 P_2 + u_3 P_3$. Then $A = B$.

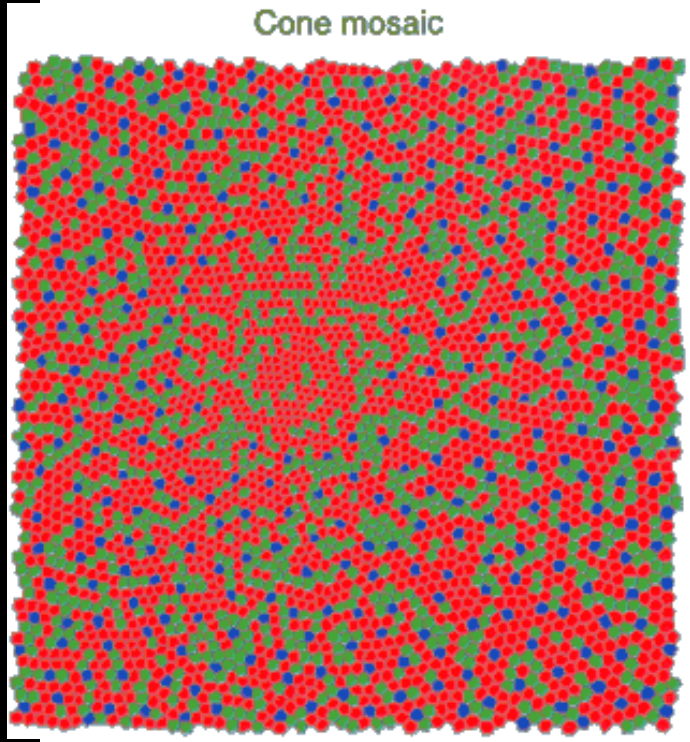
- If we mix two test lights, then mixing the matches will match the result:
  - Suppose $A = u_1 P_1 + u_2 P_2 + u_3 P_3$ and $B = v_1 P_1 + v_2 P_2 + v_3 P_3$. Then $A+B = (u_1+v_1) P_1 + (u_2+v_2) P_2 + (u_3+v_3) P_3$.

- If we scale the test light, then the matches get scaled by the same amount:
  - Suppose $A = u_1 P_1 + u_2 P_2 + u_3 P_3$. Then $kA = (ku_1) P_1 + (ku_2) P_2 + (ku_3) P_3$.

Physiology of Color Vision

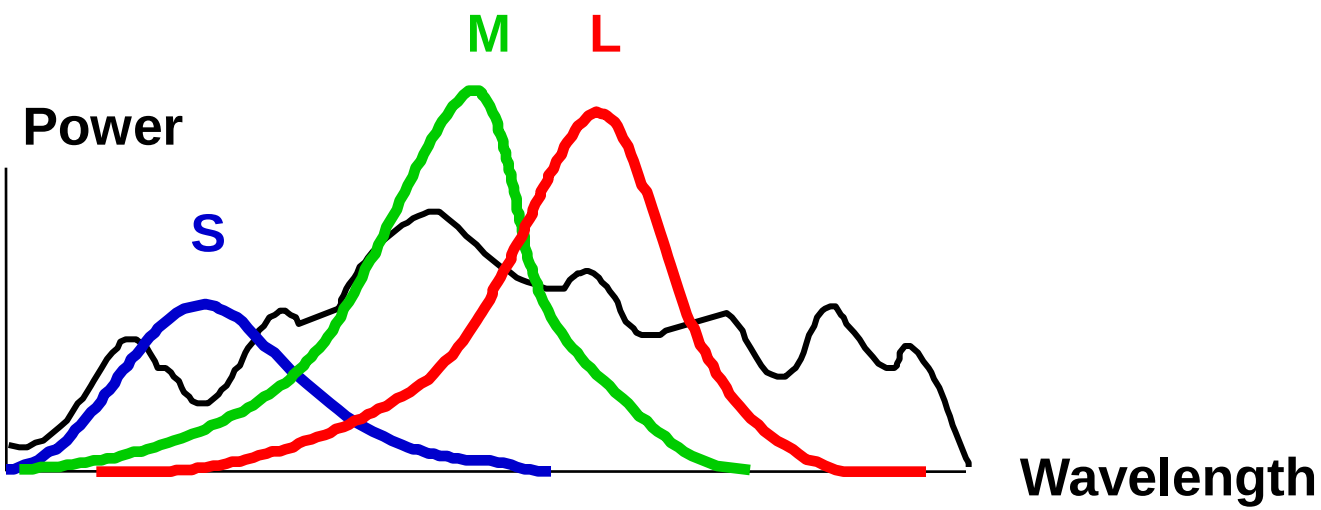
Three kinds of cones:





- Why are M and L cones so close?
- Are are there 3?

Color perception



Rods and cones act as filters on the spectrum

- To get the output of a filter, multiply its response curve by the spectrum, integrate over all wavelengths
  - Each cone yields one number

- Q: How can we represent an entire spectrum with 3 numbers?
- A: We can't! Most of the information is lost.
  - As a result, two different spectra may appear indistinguishable
    - » such spectra are known as **metamers**

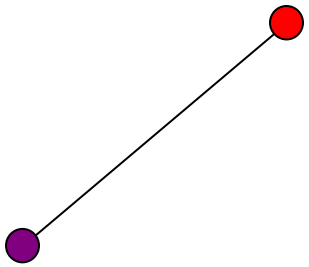
Slide by Steve Seitz

Overview of Color

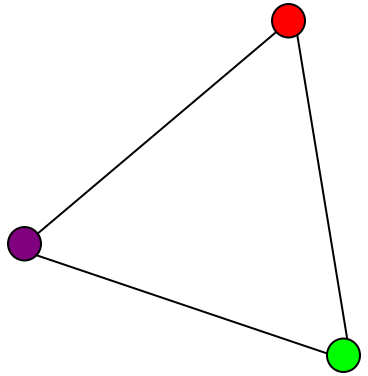
- Physics of color
- Human encoding of color
- Color spaces
- Inference from Color

Linear color spaces

- Defined by a choice of three *primaries*
- The coordinates of a color are given by the weights of the primaries used to match it

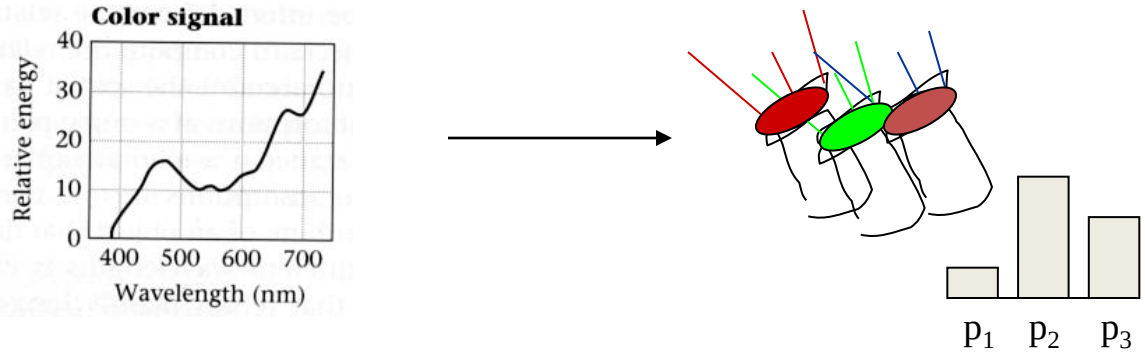


mixing two lights produces colors that lie along a straight line in color space



mixing three lights produces colors that lie within the triangle they define in color space

How to compute the weights of the primaries to match any spectral signal



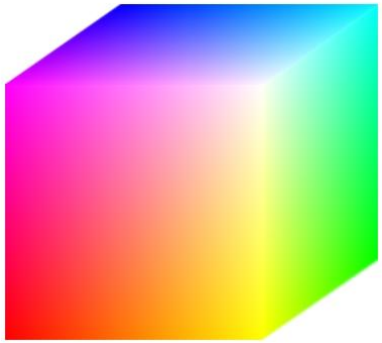
- Matching functions: the amount of each primary needed to match a monochromatic light source at each wavelength

Source: W. Freeman

RGB space

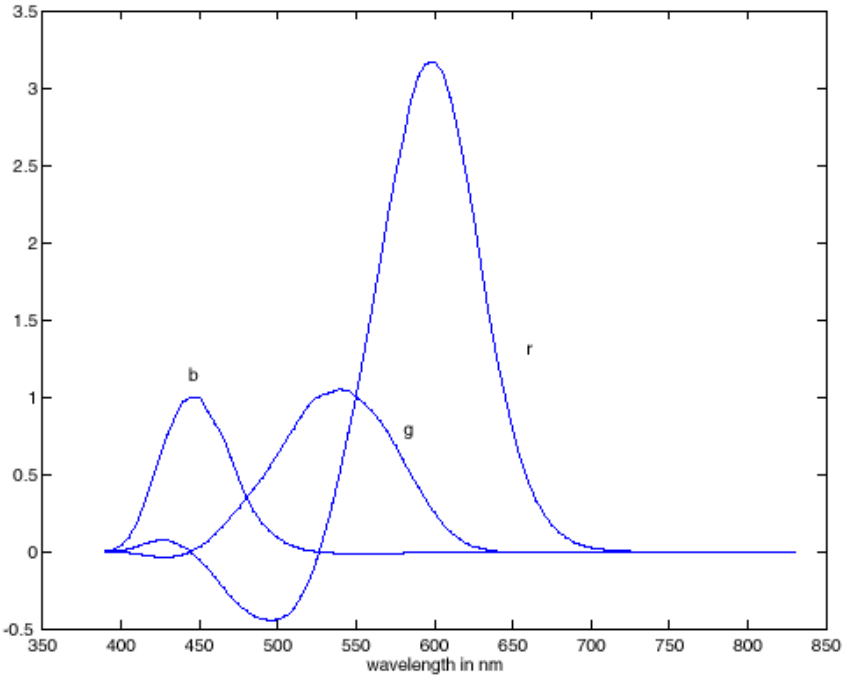
- Primaries are monochromatic lights (for monitors, they correspond to the three types of phosphors)
- *Subtractive matching* required for some wavelengths

RGB primaries



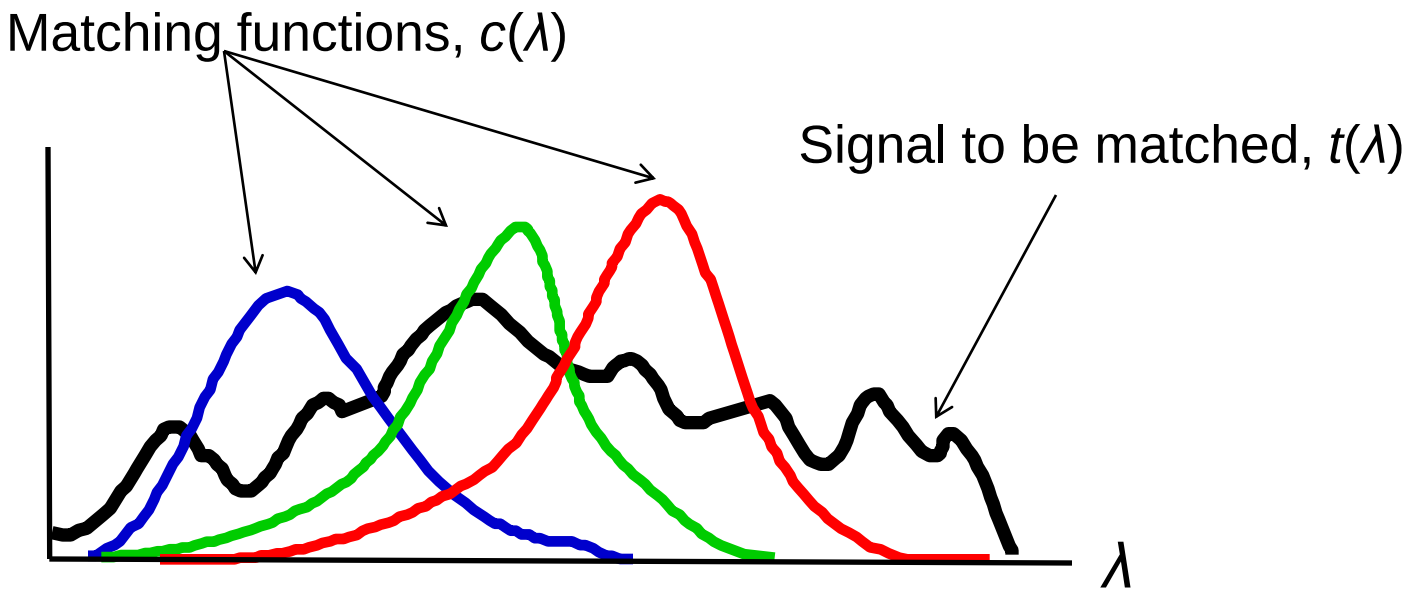
$p_1 = 645.2$ nm
$p_2 = 525.3$ nm
$p_3 = 444.4$ nm

RGB matching functions



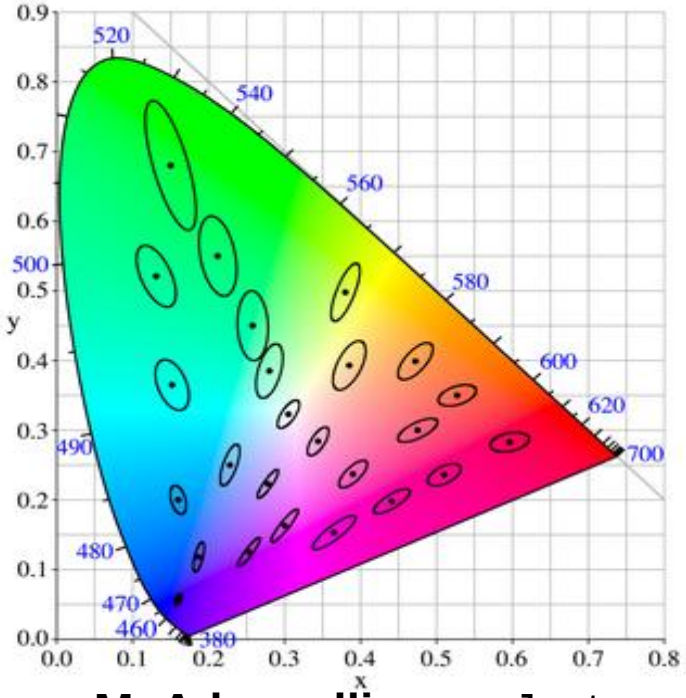
How to compute the weights of the primaries to match any spectral signal

- Let $c(\lambda)$ be one of the matching functions, and let $t(\lambda)$ be the spectrum of the signal. Then the weight of the corresponding primary needed to match t is

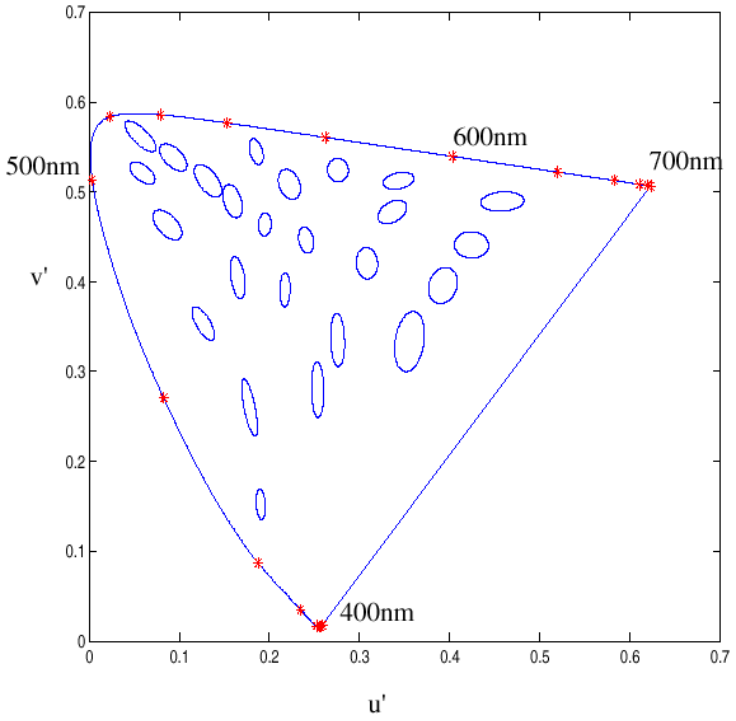


Uniform color spaces

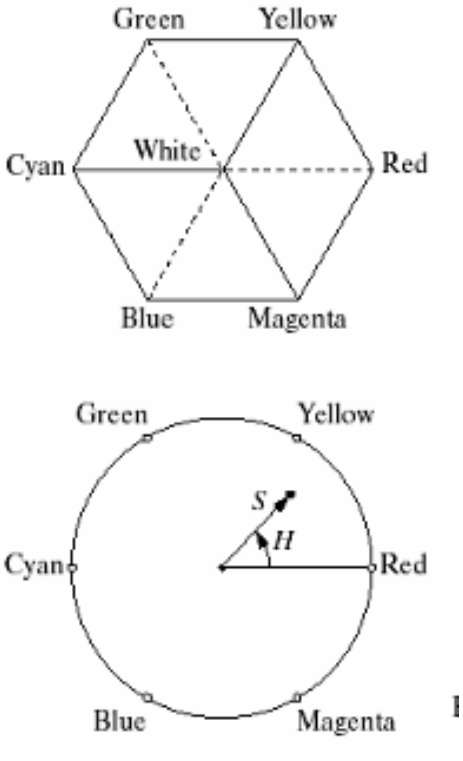
- Unfortunately, differences in x,y coordinates do not reflect perceptual color differences
- CIE u'v' is a projective transform of x,y to make the ellipses more uniform

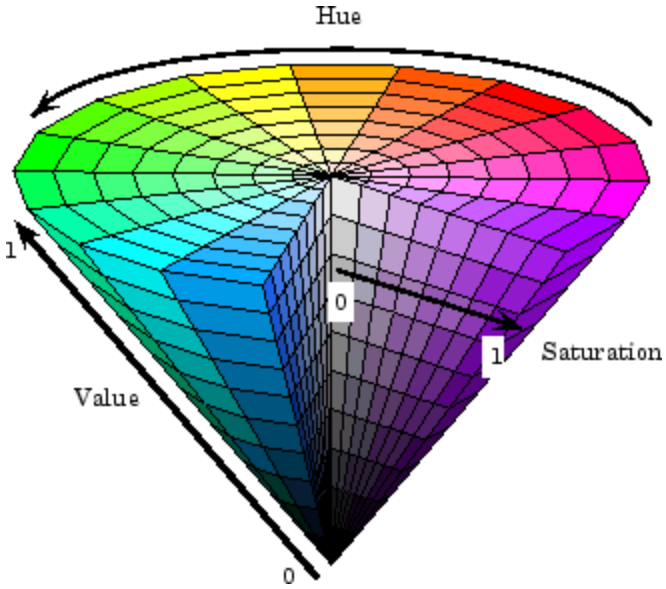


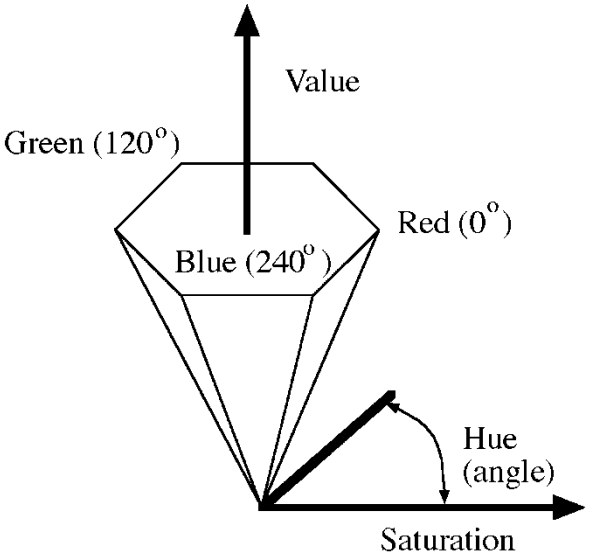
McAdam ellipses: Just noticeable differences in color



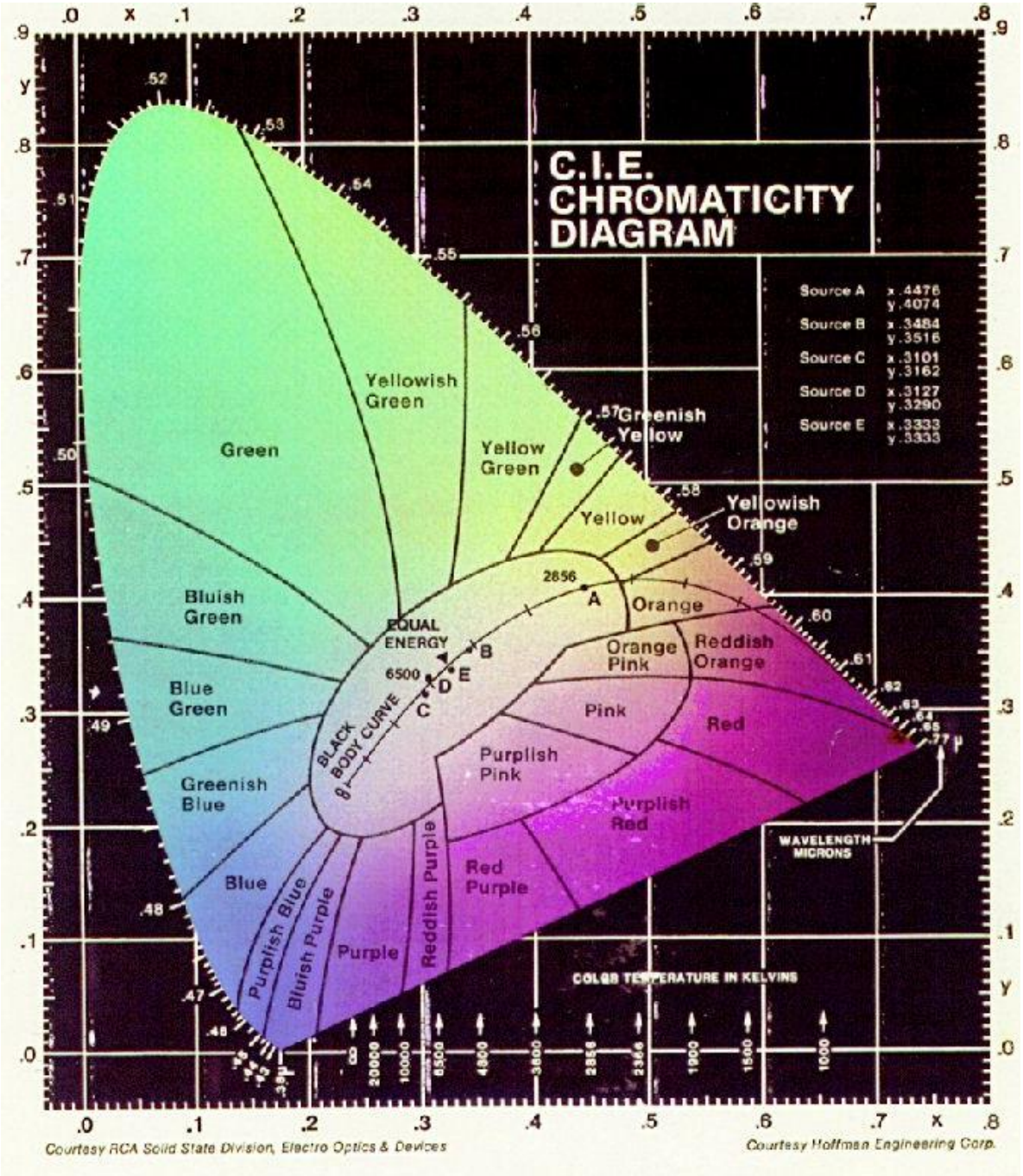
Nonlinear color spaces: HSV







- Perceptually meaningful dimensions: Hue, Saturation, Value (Intensity)
- RGB cube on its vertex

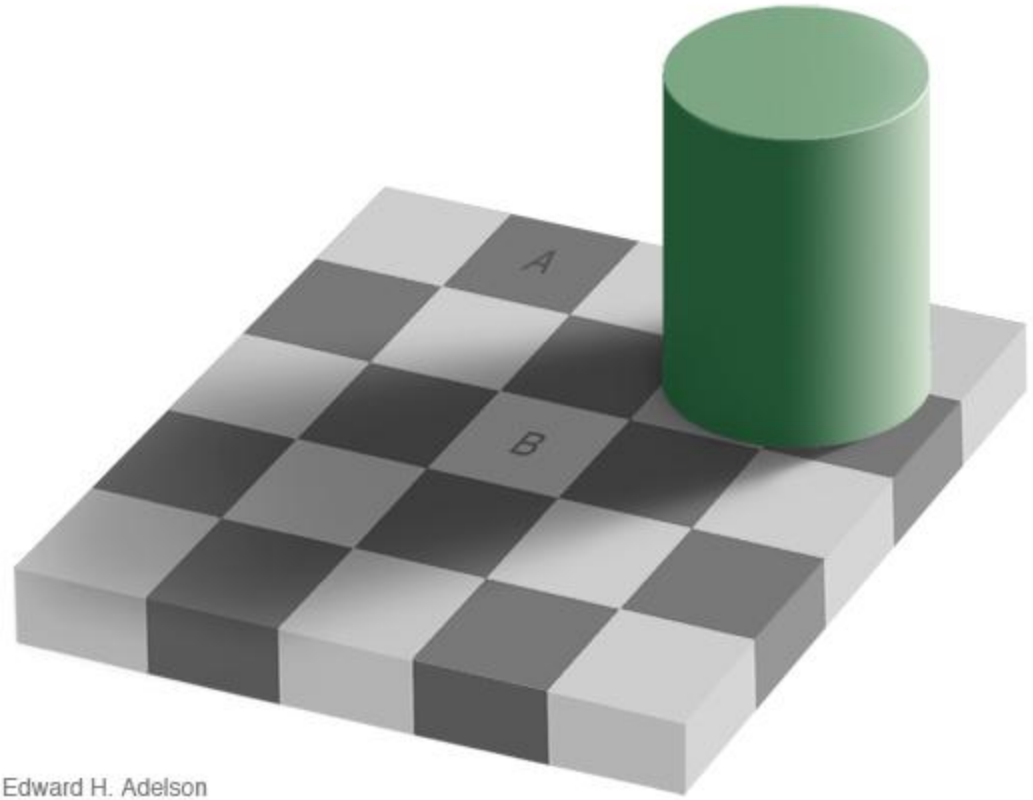

C.I.E.
CHROMATICITY
DIAGRAM
Source A x .4476 y .4074
Source B x .3484 y .3516
Source C x .3101 y .3162
Source D x .3127 y .3290
Source E x .3333 y .3333
Green
Yellowish Green
Greenish Yellow
Yellow Green
Yellow
Yellowish Orange
Orange
Reddish Orange
Orange Pink
Pink
Red
Purplish Pink
Purplish Red
Red Purple
Reddish Purple
Purple
Bluish Purple
Purplish Blue
Blue
Greenish Blue
Blue Green
Bluish Green
EQUAL ENERGY
BLACK BODY CURVE
2856
6500
A
B
C
D
E
WAVELENGTH MICRONS
COLOR TEMPERATURE IN KELVINS
Courtesy RCA Solid State Division, Electro Optics & Devices
Courtesy Hoffman Engineering Corp.

Color perception

- Color/lightness constancy
  - The ability of the human visual system to perceive the intrinsic reflectance properties of the surfaces despite changes in illumination conditions
- Instantaneous effects
  - Simultaneous contrast
  - Mach bands
- Gradual effects
  - Light/dark adaptation
  - Chromatic adaptation
  - Afterimages

J. S. Sargent, The Daughters of Edward D. Boit, 1882

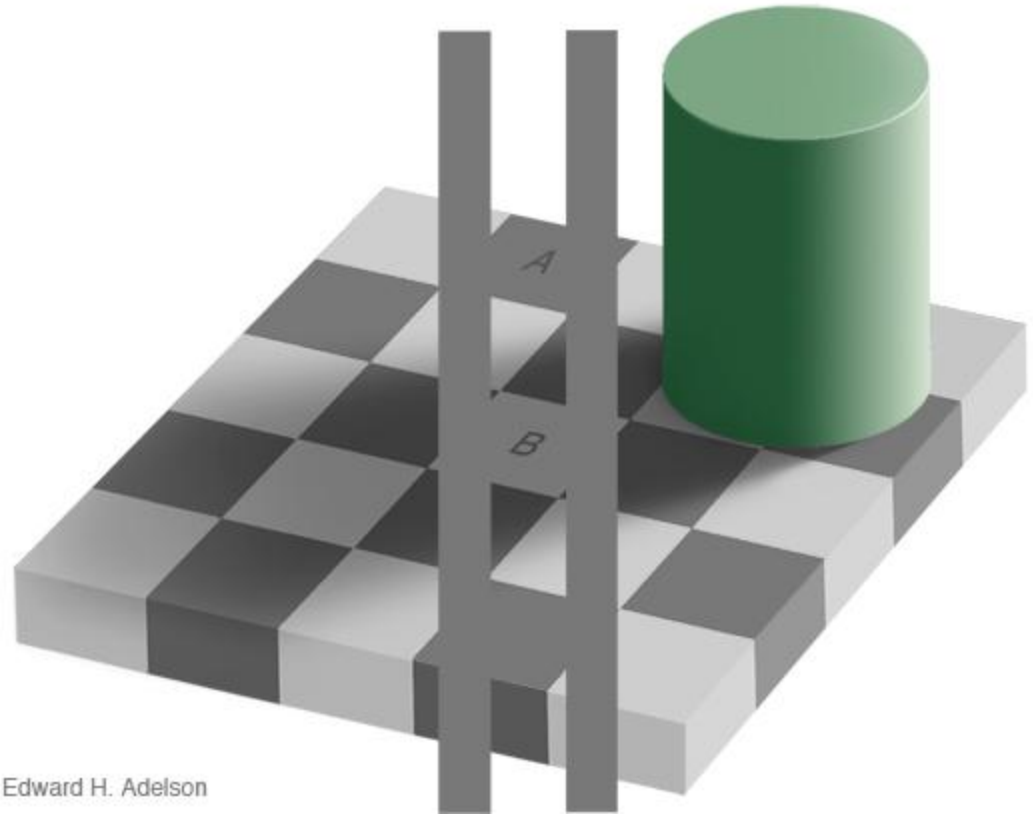
Checker shadow illusion



Edward H. Adelson

http://web.mit.edu/persci/people/adelson/checkershadow_illusion.html

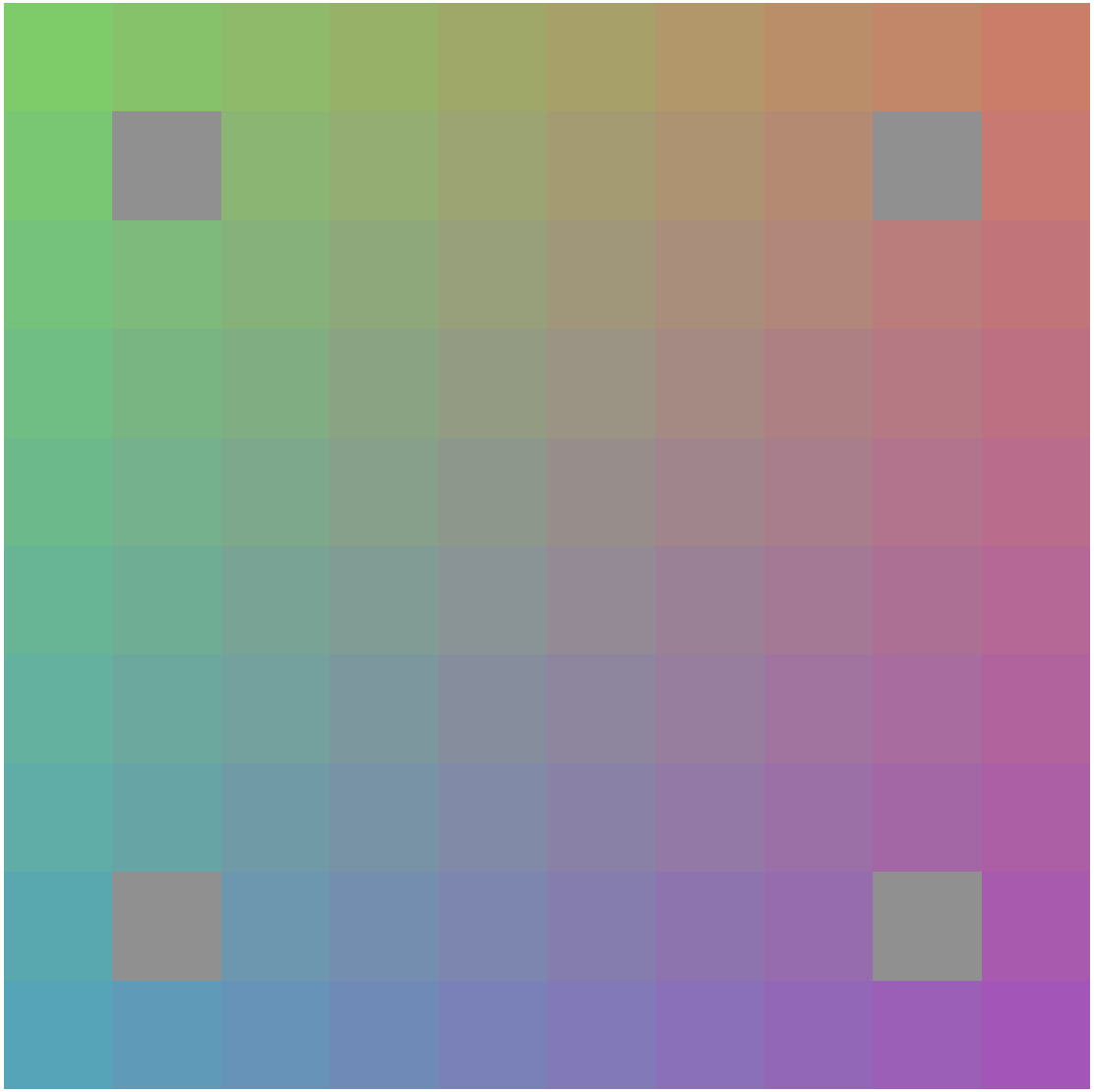
Checker shadow illusion



- Possible explanations
  - Simultaneous contrast
  - Reflectance edges vs. illumination edges

http://web.mit.edu/persci/people/adelson/checkershadow_illusion.html

Simultaneous contrast/Mach bands



Source: D. Forsyth

Chromatic adaptation

- The visual system changes its sensitivity depending on the luminances prevailing in the visual field
  - The exact mechanism is poorly understood
- Adapting to different brightness levels
  - Changing the size of the iris opening (i.e., the aperture) changes the amount of light that can enter the eye
  - Think of walking into a building from full sunshine
- Adapting to different color temperature
  - The receptive cells on the retina change their sensitivity
  - For example: if there is an increased amount of red light, the cells receptive to red decrease their sensitivity until the scene looks white again
  - We actually adapt better in brighter scenes: This is why candlelit scenes still look yellow

http://www.schorsch.com/kbase/glossary/adaptation.html

White balance

- When looking at a picture on screen or print, our eyes are adapted to the illuminant of the room, not to that of the scene in the picture
- When the white balance is not correct, the picture will have an unnatural color “cast”

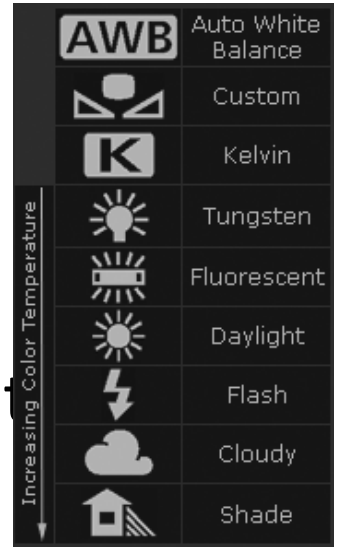
incorrect white balance

correct white balance

http://www.cambridgeincolour.com/tutorials/white-balance.htm

White balance

- Film cameras:
  - Different types of film or different filters for different illumination conditions
- Digital cameras:
  - Automatic white balance
  - White balance settings corresponding t several common illuminants
  - Custom white balance using a reference object



http://www.cambridgeincolour.com/tutorials/white-balance.htm

White balance

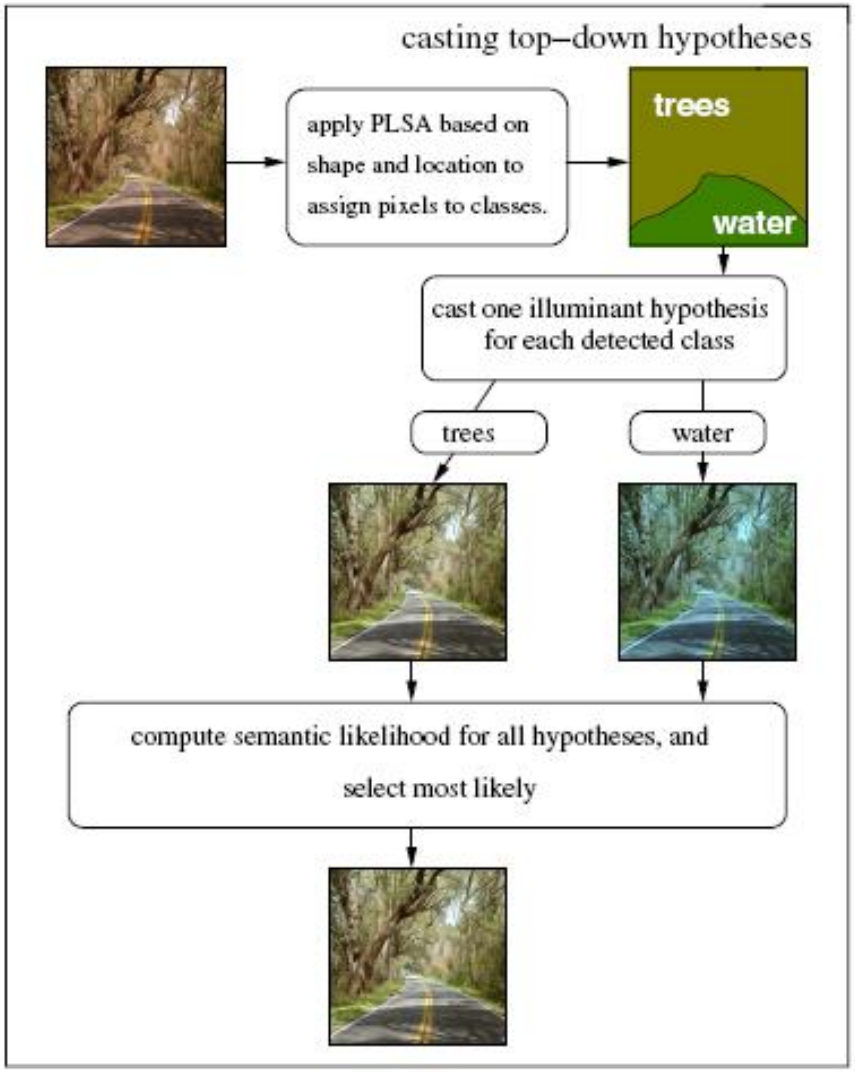
- Von Kries adaptation
  - Multiply each channel by a gain factor

- Best way: gray card
  - Take a picture of a neutral object (white or gray)
  - Deduce the weight of each channel
    - If the object is recoded as r_w, g_w, b_w
      use weights $1/r_w$, $1/g_w$, $1/b_w$

White balance

- Without gray cards: we need to "guess" which pixels correspond to white objects
- Gray world assumption
  - The image average r_{ave}, g_{ave}, b_{ave} is gray
  - Use weights $1/r_{ave}$, $1/g_{ave}$, $1/b_{ave}$
- Brightest pixel assumption
  - Highlights usually have the color of the light source
  - Use weights inversely proportional to the values of the brightest pixels
- Gamut mapping
  - Gamut: convex hull of all pixel colors in an image
  - Find the transformation that matches the gamut of the image to the gamut of a "typical" image under white light
- Use image statistics, learning techniques

White balance by recognition



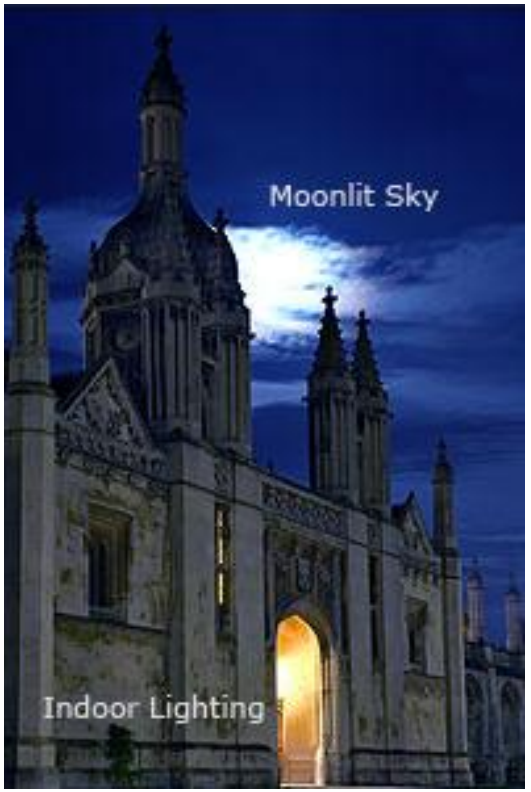
- Key idea: For each of the semantic classes present in the image, compute the illuminant that transforms the pixels assigned to that class so that the average color of that class matches the average color of the same class in a database of "typical" images

J. Van de Weijer, C. Schmid and J. Verbeek, Using High-Level Visual Information for Color Constancy, ICCV 2007.

Mixed illumination

- When there are several types of illuminants in the scene, different reference points will yield different results

Reference: moon

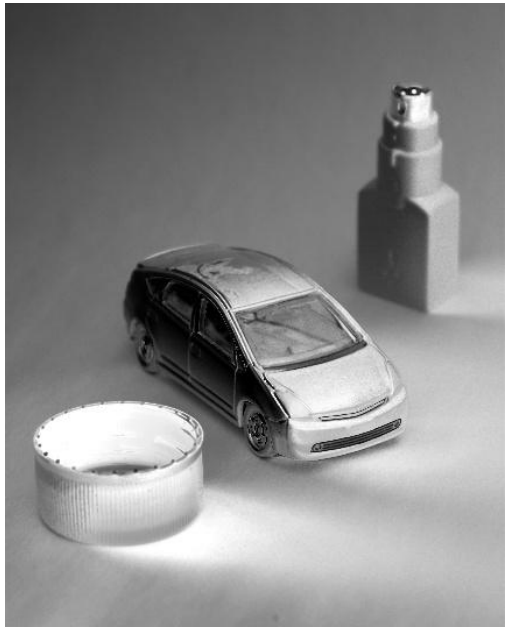


Reference: stone

http://www.cambridgeincolour.com/tutorials/white-balance.htm

Spatially varying white balance

Input



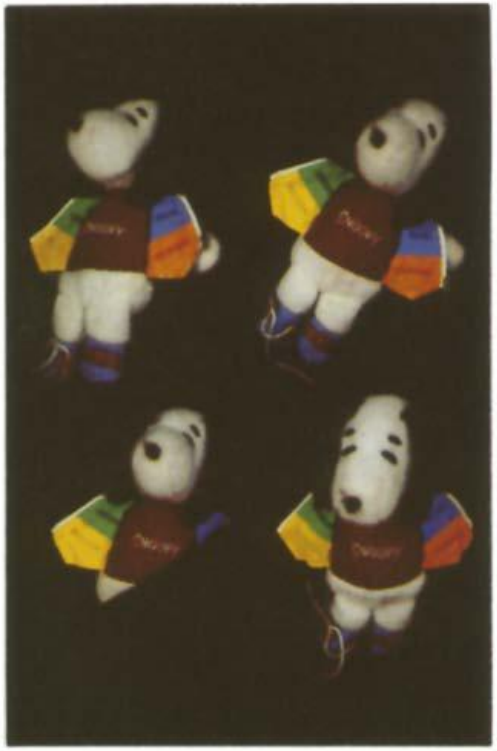
Alpha map

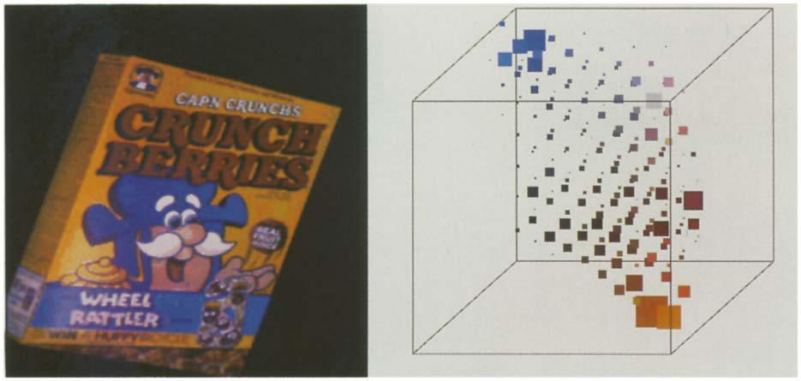
Output

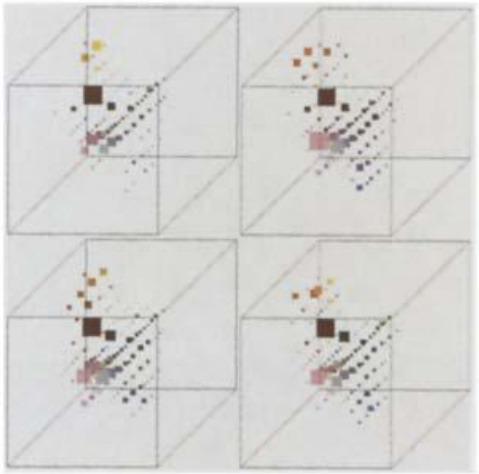
E. Hsu, T. Mertens, S. Paris, S. Avidan, and F. Durand, "Light Mixture Estimation for Spatially Varying White Balance," SIGGRAPH 2008

Uses of color in computer vision

Color histograms for image matching

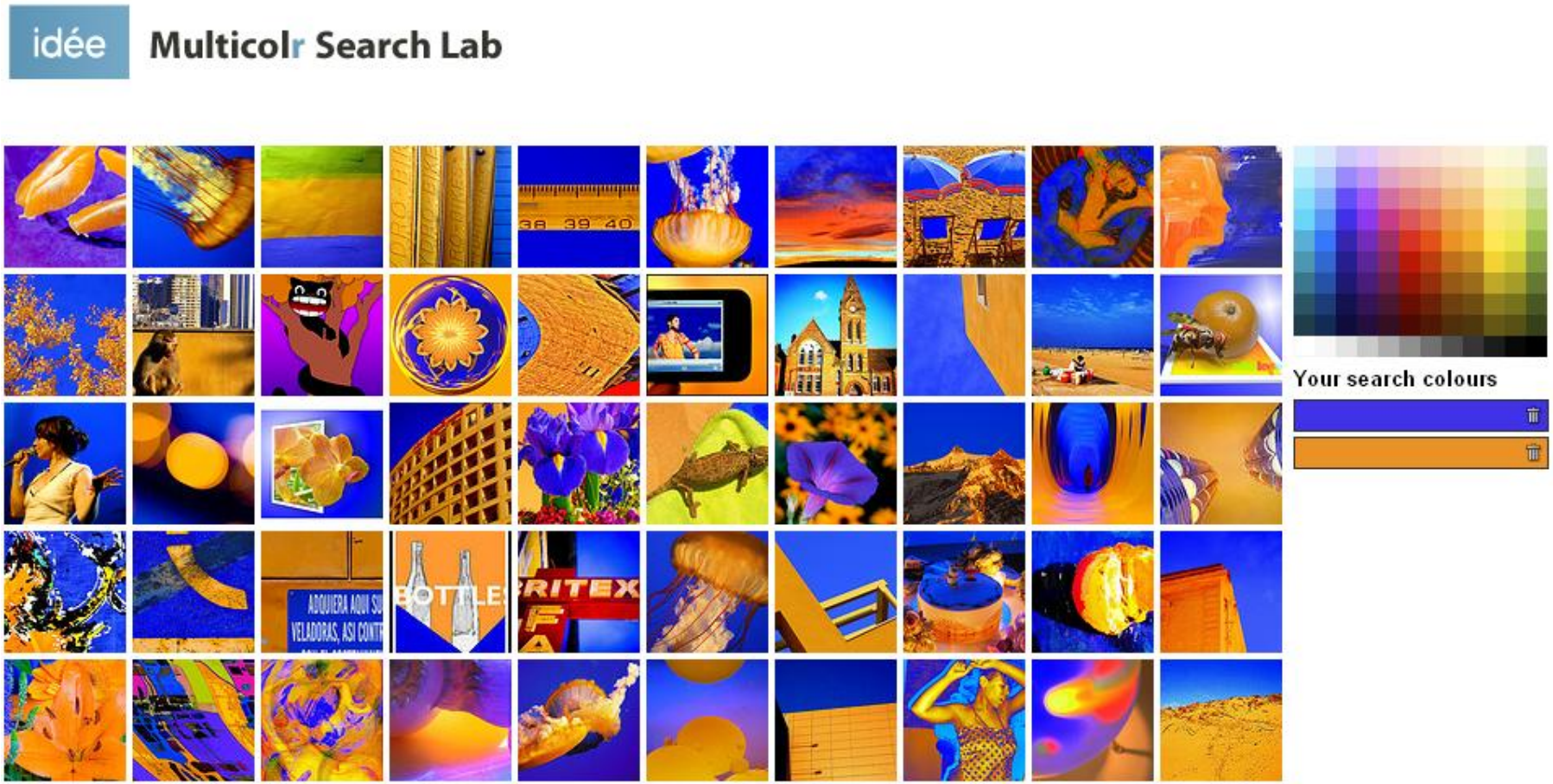






Swain and Ballard, Color Indexing, IJCV 1991.

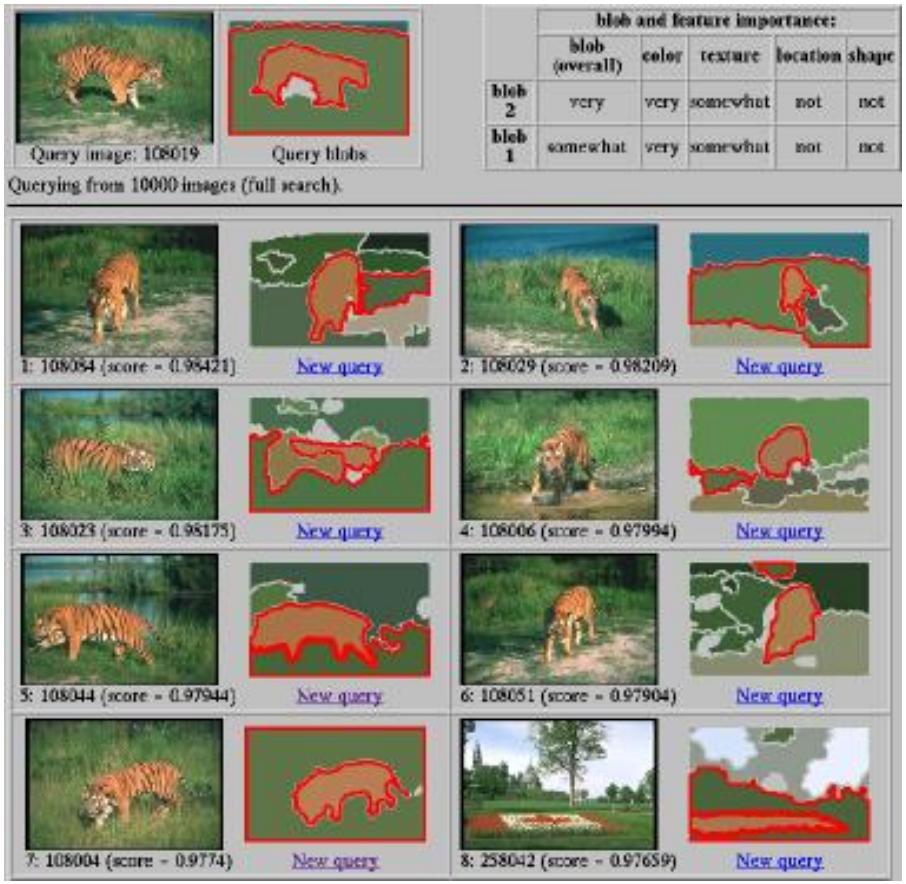
Uses of color in computer vision



http://labs.ideeinc.com/multicolr

Uses of color in computer vision

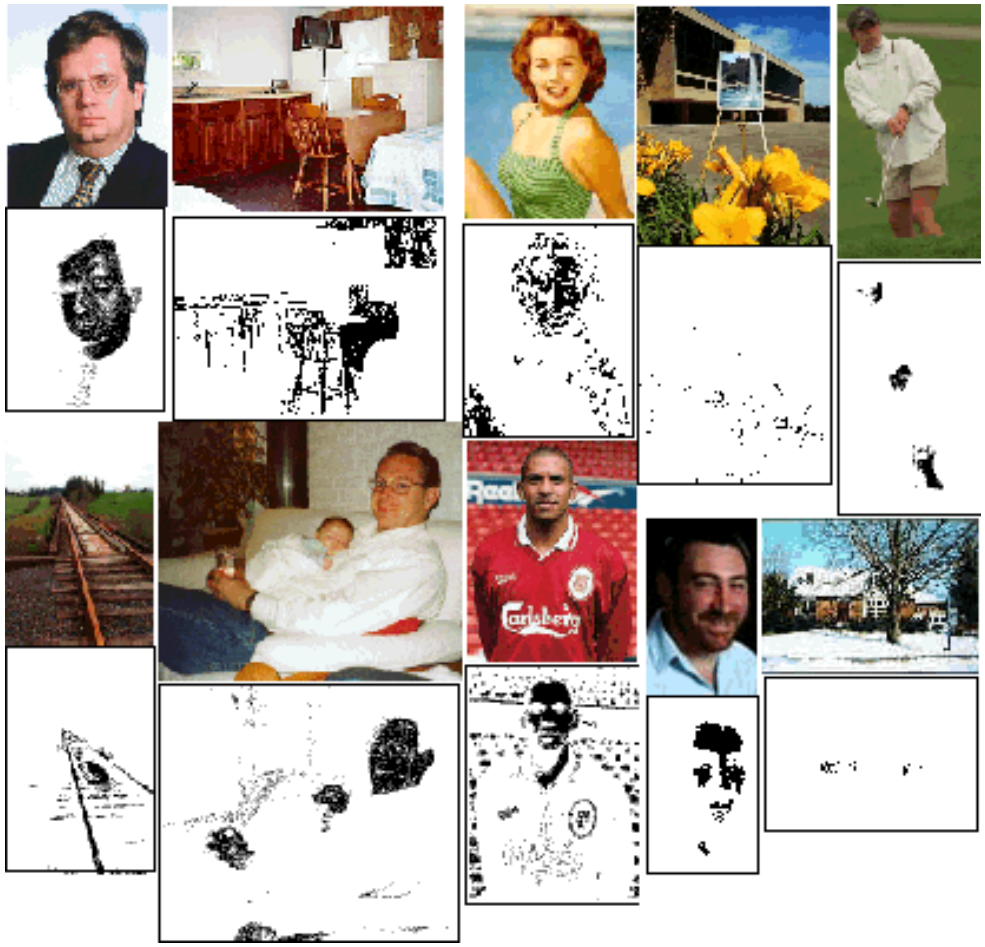
Image segmentation and retrieval



C. Carson, S. Belongie, H. Greenspan, and Ji. Malik, Blobworld: Image segmentation using Expectation-Maximization and its application to image querying, ICVIS 1999.

Uses of color in computer vision

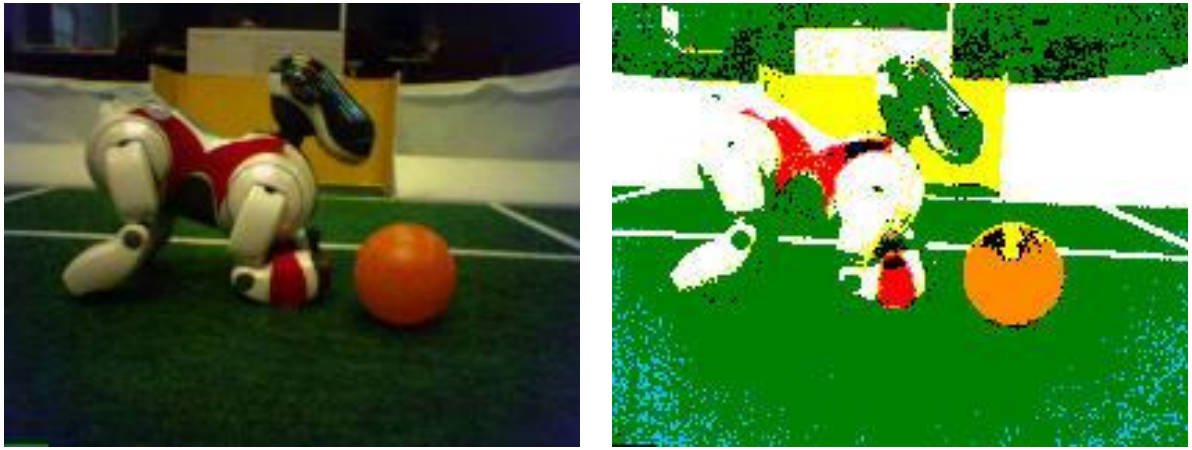
Skin detection



M. Jones and J. Rehg, Statistical Color Models with Application to Skin Detection, IJCV 2002.

Uses of color in computer vision

Robot soccer

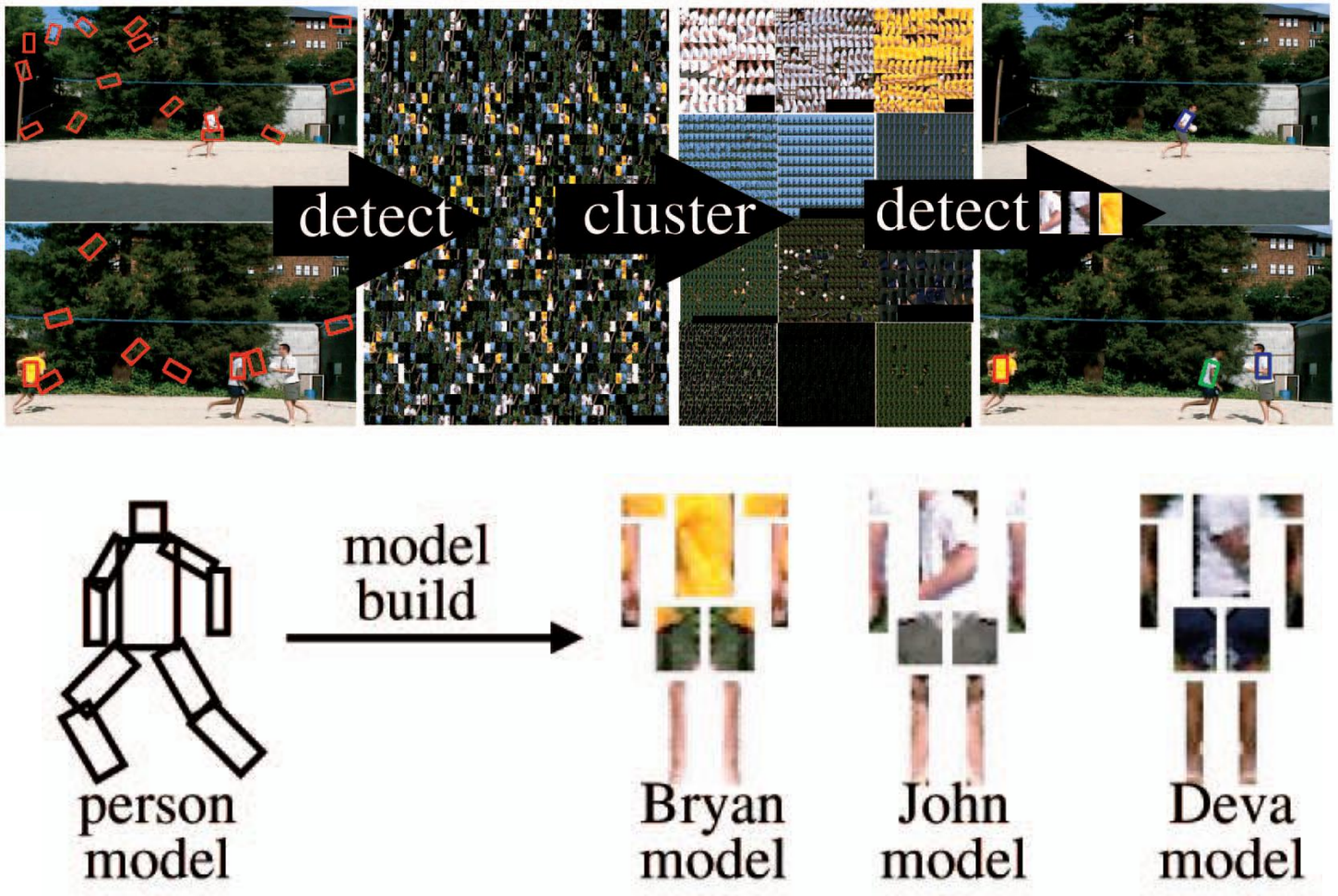


M. Sridharan and P. Stone, Towards Eliminating Manual Color Calibration at RoboCup. RoboCup-2005: Robot Soccer World Cup IX, Springer Verlag, 2006

Source: K. Grauman

Uses of color in computer vision

Building appearance models for tracking



D. Ramanan, D. Forsyth, and A. Zisserman. Tracking People by Learning their Appearance. PAMI 2007.

Local Image Features

- Properties of detectors
  - Edge detectors
  - Corners
  - Blobs
  - Scale invariant detection
- Properties of descriptors
  - HOG
  - SIFT
  - Color
  - Texture
  - Shape context

Texture

Google

texture

Search

SafeSearch strict ▼

About 45,000,000 results (0.31 seconds)

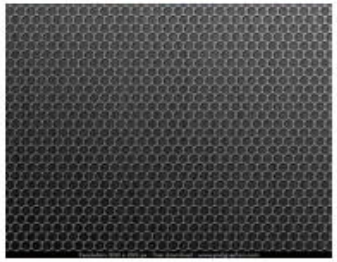
Advanced search

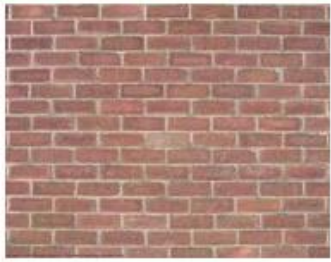













Google

no texture

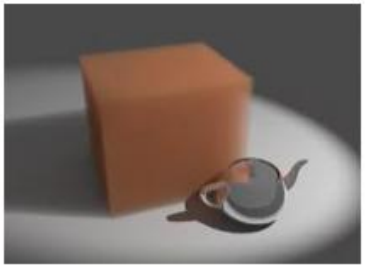
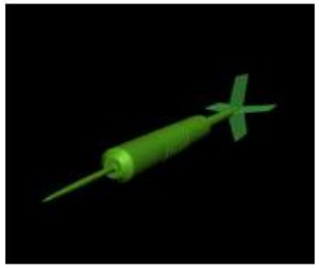
Search

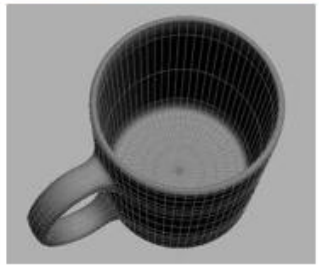
SafeSearch strict ▾

About 30,800,000 results (0.81 seconds)

Advanced search









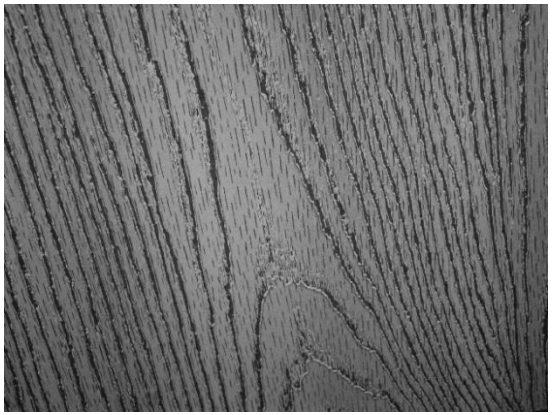


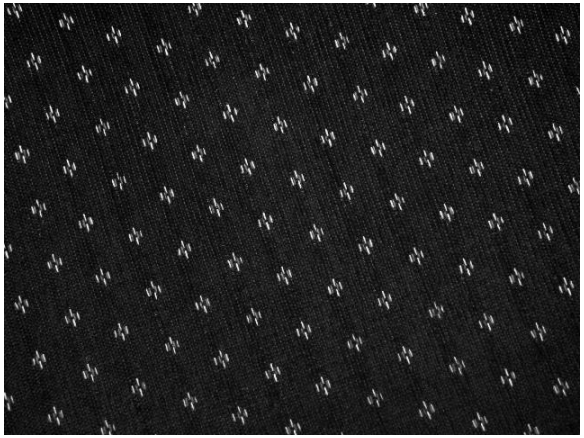


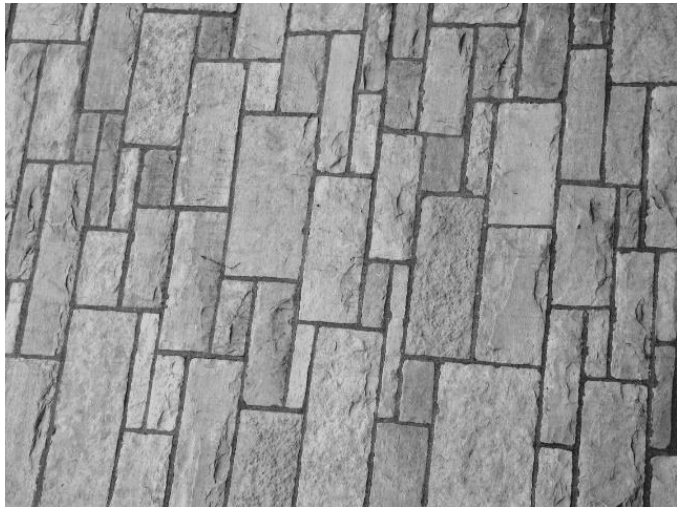
Texture

- Patterns of structure from
  - changes in surface albedo (eg printed cloth)
  - changes in surface shape (eg bark)
  - many small surface patches (eg leaves on a bush)
- Hard to define; but texture tells us
  - what a surface is like
  - (sometimes) object identity
  - (sometimes) surface shape

Texture and Material

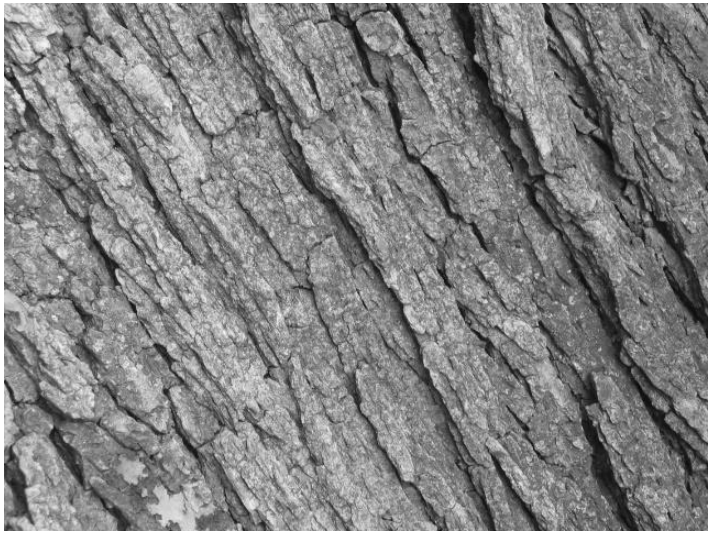


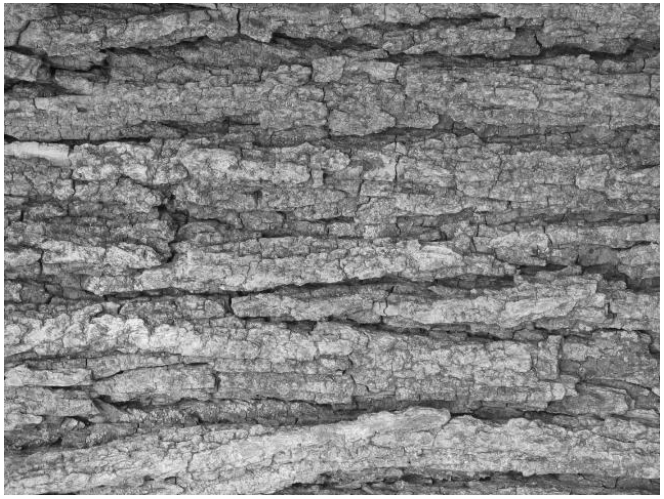




http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/

Texture and Orientation





http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/

Texture and Scale

http://www-cvr.ai.uiuc.edu/ponce_grp/data/texture_database/samples/

Texture: Core Problems

- Represent complex surface textures to recognize
  - objects
  - materials
  - textures
- Synthesize texture from examples
  - to create big textures for computer graphics
  - to fill in holes in images caused by editing

Texture representation

- Core idea: Textures consist of
  - a set of elements
  - repeated in some way

- Representations
  - identify the elements
  - summarize the repetition

FIGURE 6.1: Although texture is difficult to define, it has some important and valuable properties. In this image, there are many repeated elements (some leaves form repeated "spots"; others, and branches, form "bars" at various scales; and so on). Our perception of the material is quite intimately related to the texture (what would the surface feel like if you ran your fingers over it? what is soggy? what is prickly? what is smooth?). Notice how much information you are getting about the type of plants, their shape, the shape of free space, and so on, from the textures. *Geoff Brightling © Dorling Kindersley, used with permission.*

Notice how the change in pattern elements and repetitions is the main difference between different textured surfaces
(the plants, the ground, etc.)

Different materials tend to have different textures (though these are not the same ideas)

FIGURE 6.2: Typically, different materials display different image textures. These are example images from a collection of 1,000 material images, described in by Sharan *et al.* (2009); there are 100 images in each of the ten categories, including the two categories shown here (fabric and stone). Notice how (a) the textures vary widely, even within a material category; and (b) different materials seem to display quite different textures. *This figure shows elements of a database collected by C. Liu, L. Sharan, E. Adelson, and R. Rosenholtz, and published at* `http://people.csail.mit.edu/lavanya/research_sharan.html`. *Figure by kind permission of the collectors.*

Filter based texture representations

- Choose a set of filters, each representing a pattern element
  - typically a spot and some oriented bars
- Filter the image at a variety of scales
- Rectify the filtered images
  - typically half wave, to avoid averaging contrast reversals
    - eg should not average dark spot on light background, light spot on dark background to zero
- Compute summaries of rectified filtered images
  - eg smoothed average
  - at a variety of scales to capture
    - nearby pattern elements and general picture of pattern elements
- Now describe each pixel by vector of summaries
  - which could be very long

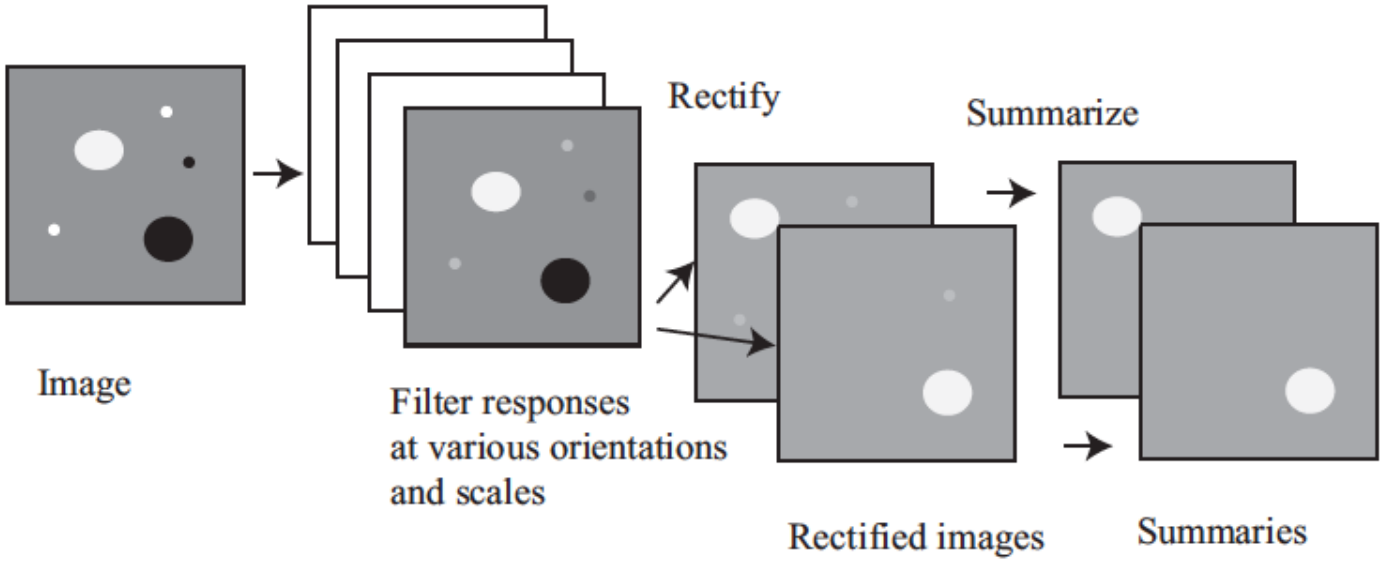


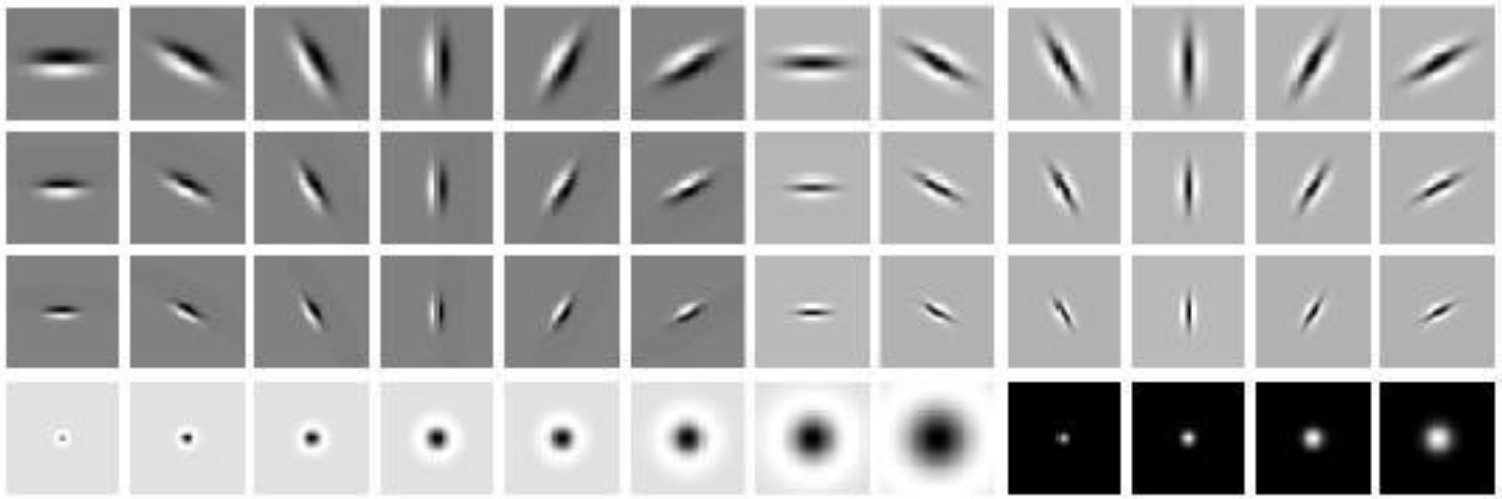
FIGURE 6.3: Local texture representations can be obtained by filtering an image with a set of filters at various scales, and then preparing a summary. Summaries ensure that, at a pixel, we have a representation of what texture appears near that pixel. The filters are typically spots and bars (see Figure 6.4). Filter outputs can be enhanced by rectifying them (so that positive and negative responses do not cancel), then computing a local summary of the rectified filter outputs. Rectifying by taking the absolute value means that we do not distinguish between light spots on a dark background and dark spots on a light background; the alternative, half-wave rectification (described in the text), preserves this distinction at the cost of a fuller representation. One can summarize either by smoothing (which will tend to suppress noise, as in the schematic example above) or by taking the maximum over a neighborhood. Compare this figure to Figure 6.7, which shows a representation for a real image.

How can we represent texture?

- Compute responses of blobs and edges at various orientations and scales

Overcomplete representation: filter banks

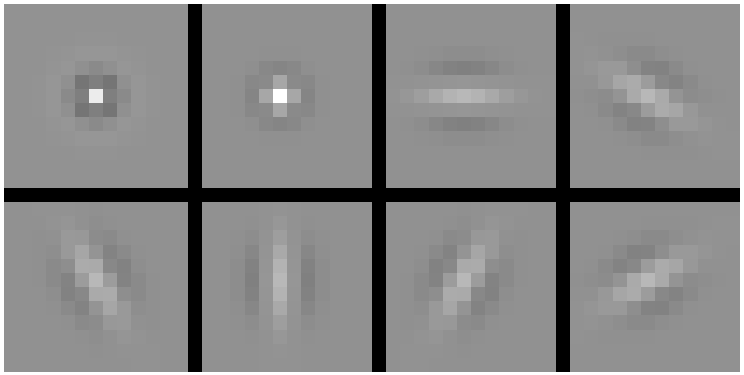
The Leung-Malik（LM）Filter Bank

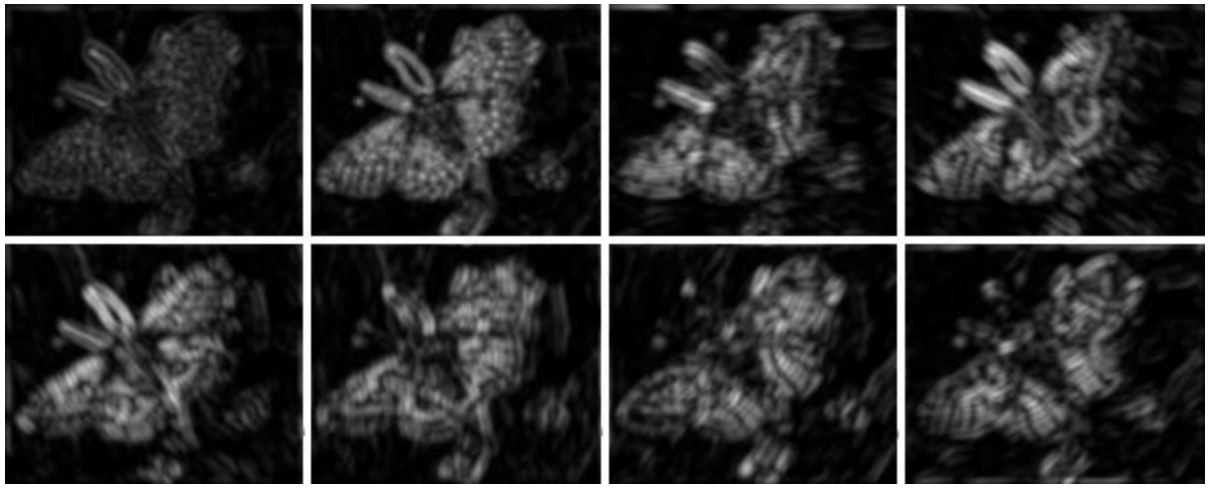


Code for filter banks: *www.robots.ox.ac.uk/~vgg/research/texclass/filters.html*

Filter banks

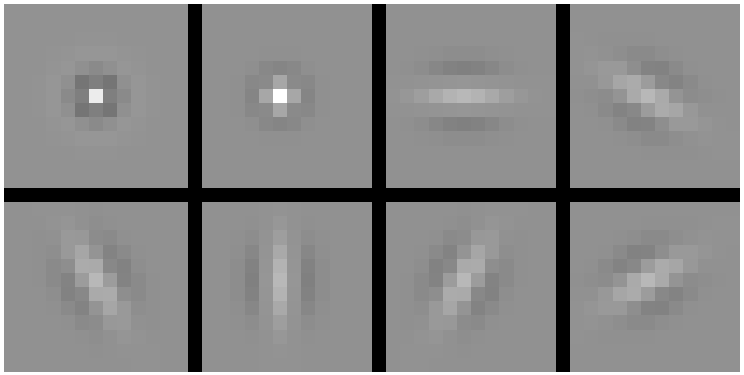
- Process image with each filter and keep responses (or squared/abs responses)

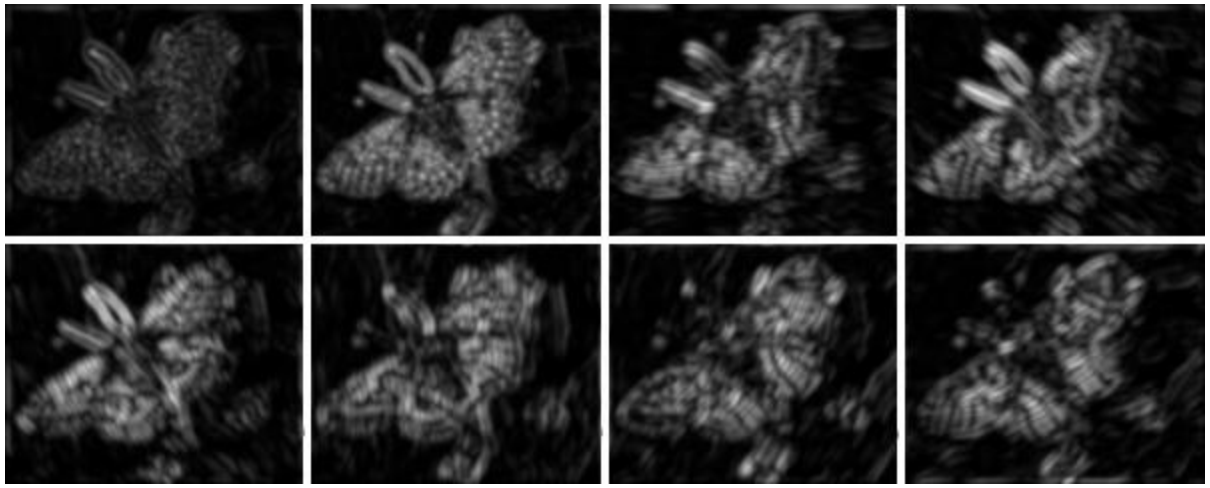




Representing texture

- Take the vectors of filter responses at each pixel and cluster them, then take histograms



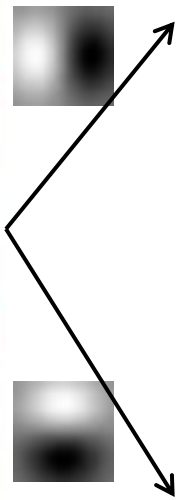


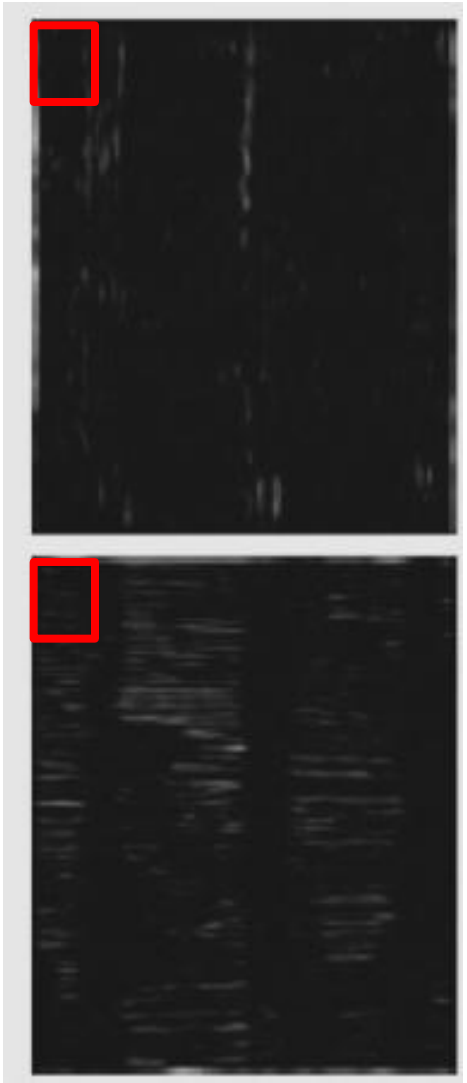
Texture representation

- Textures are made up of repeated local patterns, so:
  - Find the patterns
    - Use filters that look like patterns (spots, bars, raw patches…)
    - Consider magnitude of response
  - Describe their statistics within each local window
    - Mean, standard deviation
    - Histogram
    - Histogram of “prototypical” feature occurrences

Texture representation: example

original image





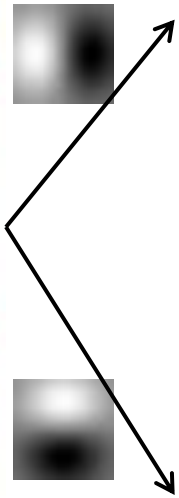
derivative filter responses, squared

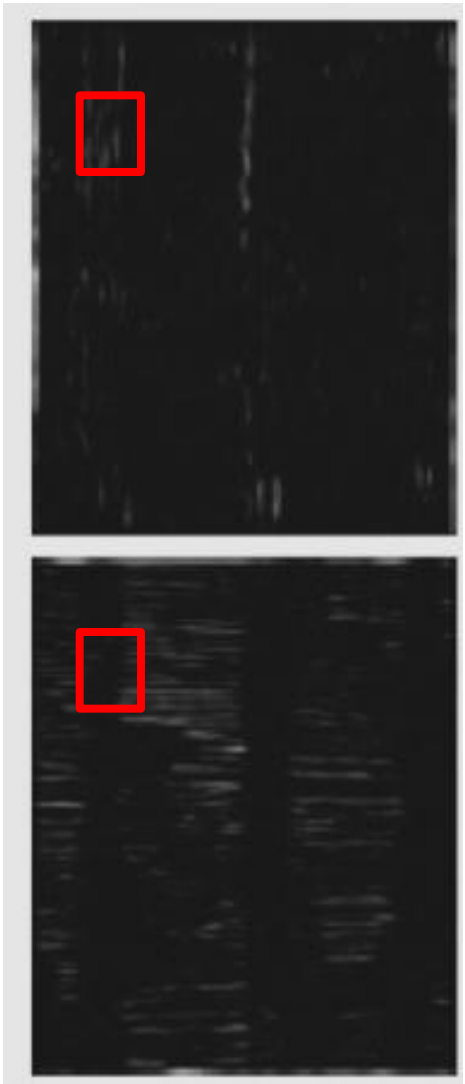
	mean d/dx value	mean d/dy value
Win. #1	4	10

statistics to summarize patterns in small windows

Texture representation: example

original image





derivative filter responses, squared

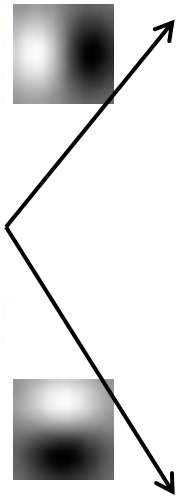
	mean d/dx value	mean d/dy value
Win. #1	4	10
Win.#2	18	7

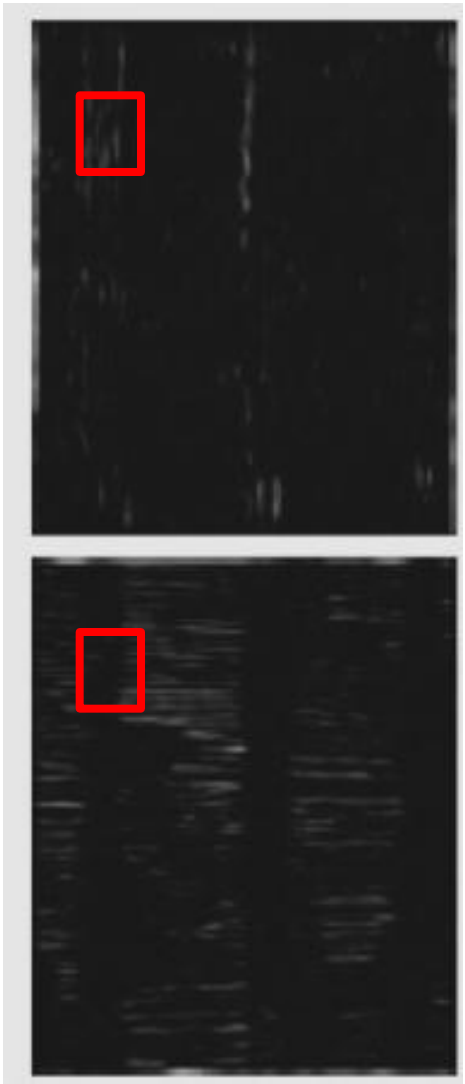
⋮

statistics to summarize patterns in small windows

Texture representation: example

original image





derivative filter responses, squared

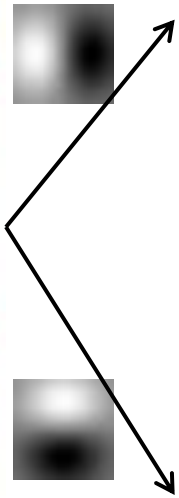
	mean d/dx value	mean d/dy value
Win. #1	4	10
Win.#2	18	7

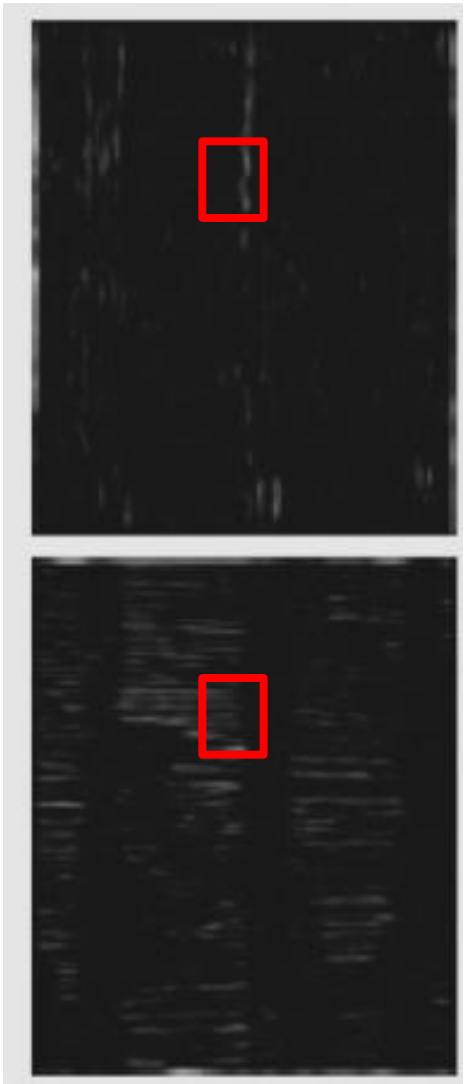
⋮

statistics to summarize patterns in small windows

Texture representation: example

original image





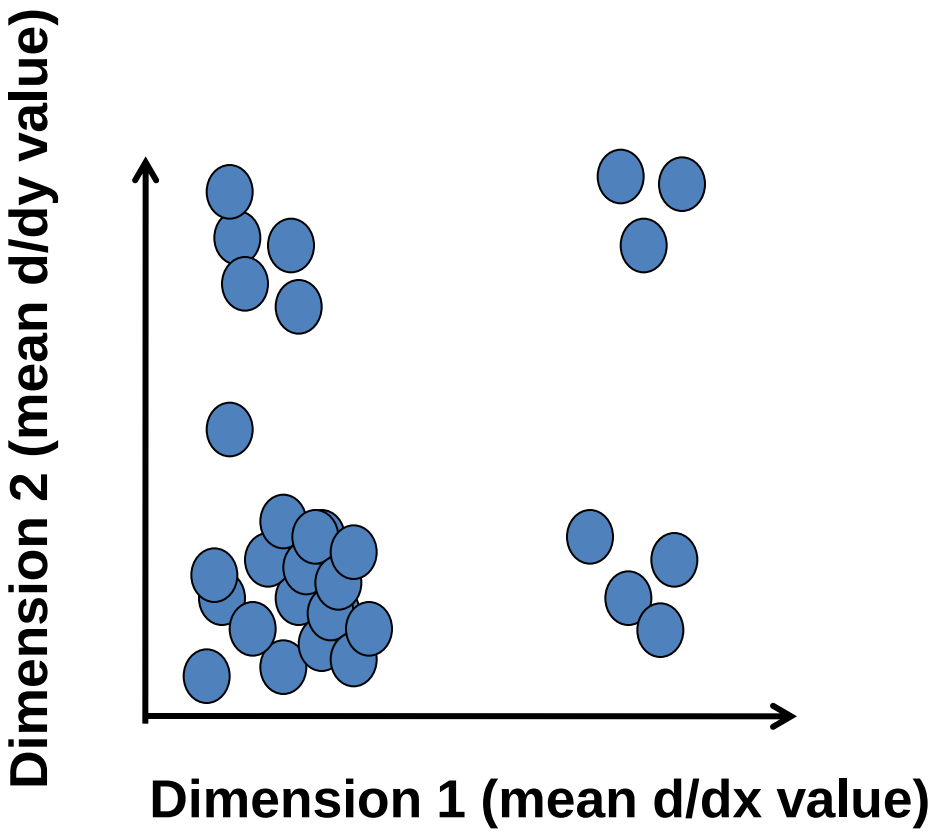
derivative filter responses, squared

	mean d/dx value	mean d/dy value
Win. #1	4	10
Win.#2	18	7
⋮		
Win.#9	20	20

⋮

statistics to summarize patterns in small windows

Texture representation: example

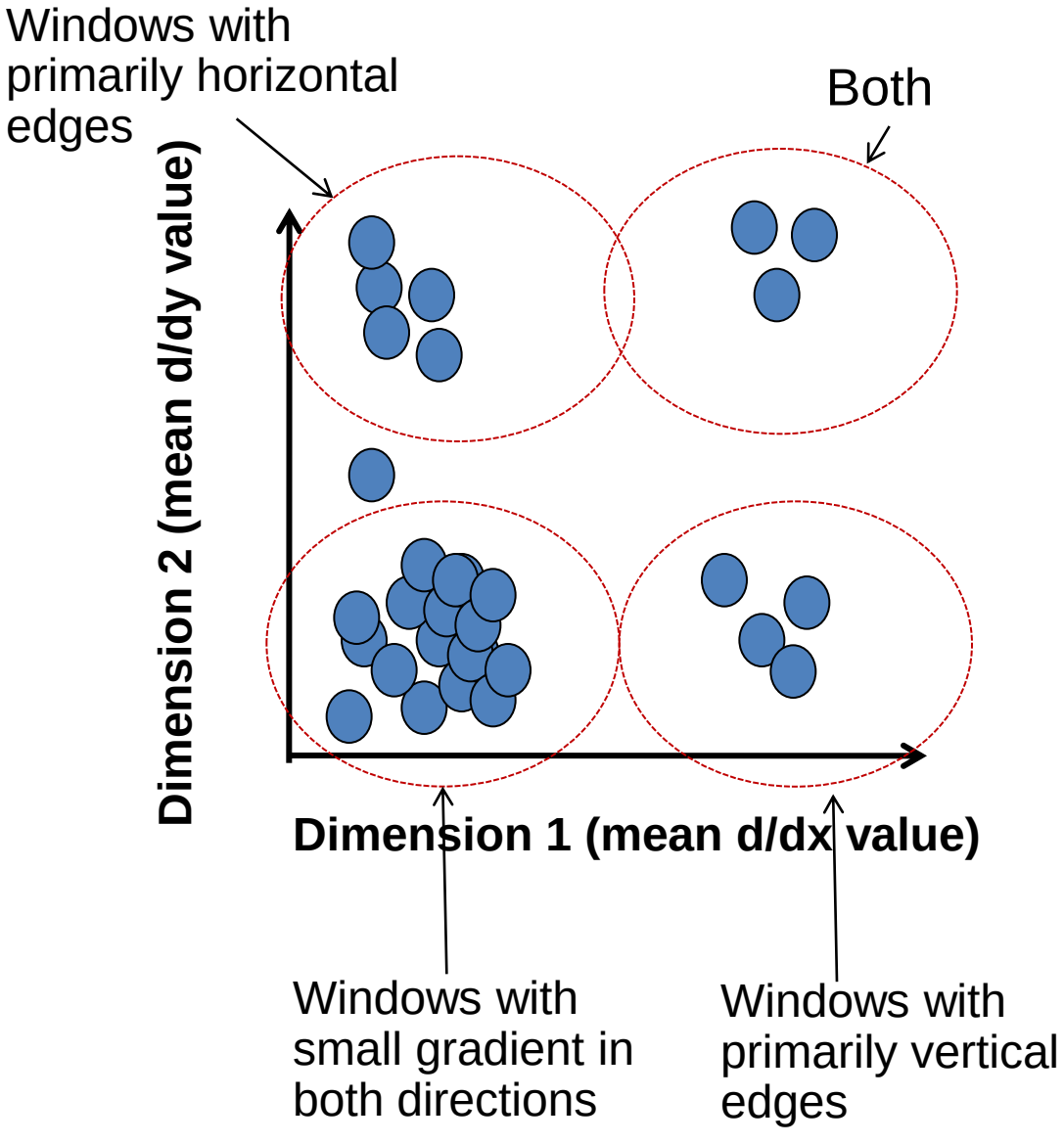


	mean d/dx value	mean d/dy value
Win. #1	4	10
Win.#2	18	7
⋮		
Win.#9	20	20

⋮

statistics to summarize patterns in small windows

Texture representation: example



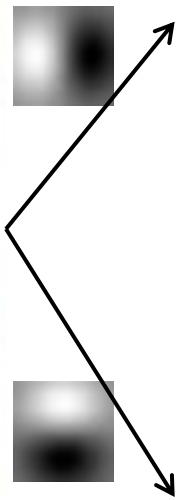
	mean d/dx value	mean d/dy value
Win. #1	4	10
Win.#2	18	7
⋮		
Win.#9	20	20

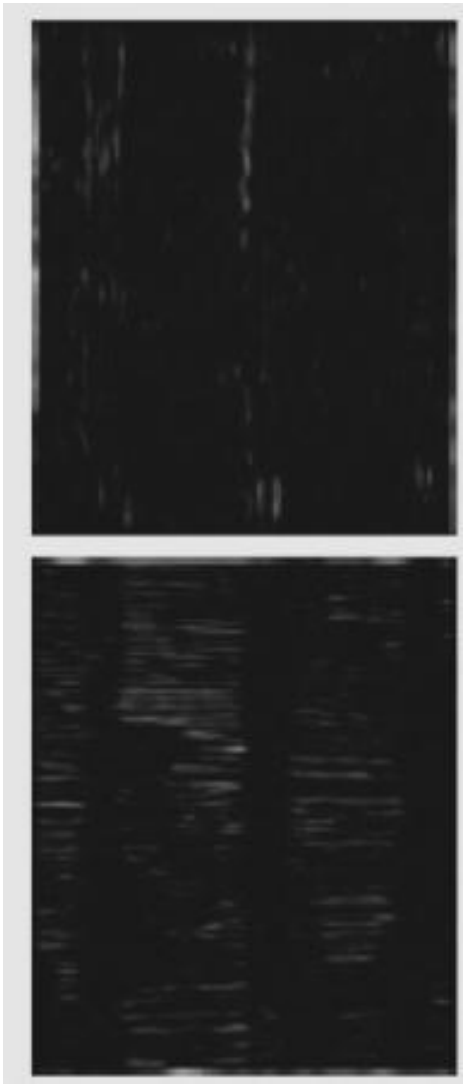
⋮

statistics to summarize patterns in small windows

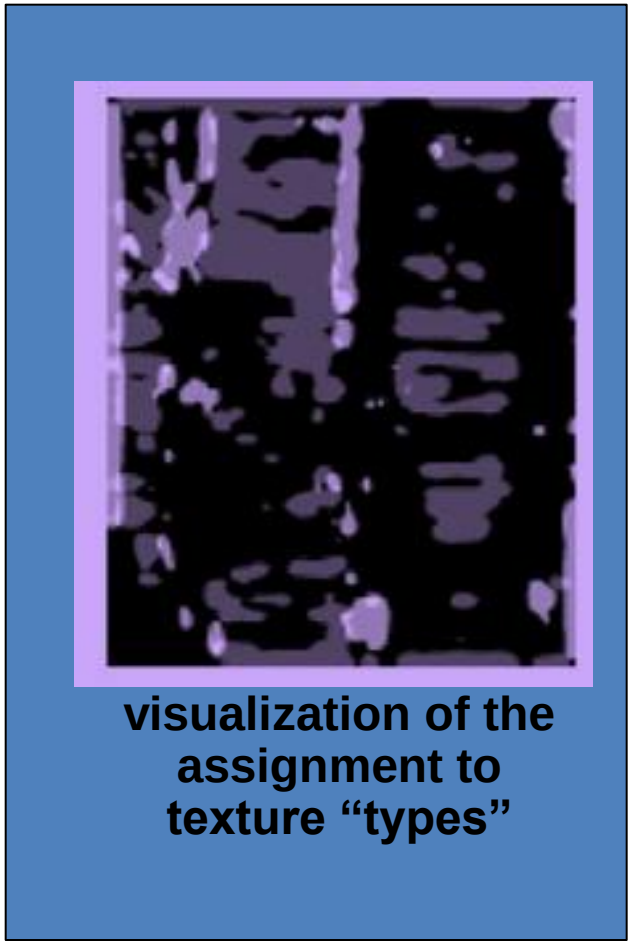
Texture representation: example

original image



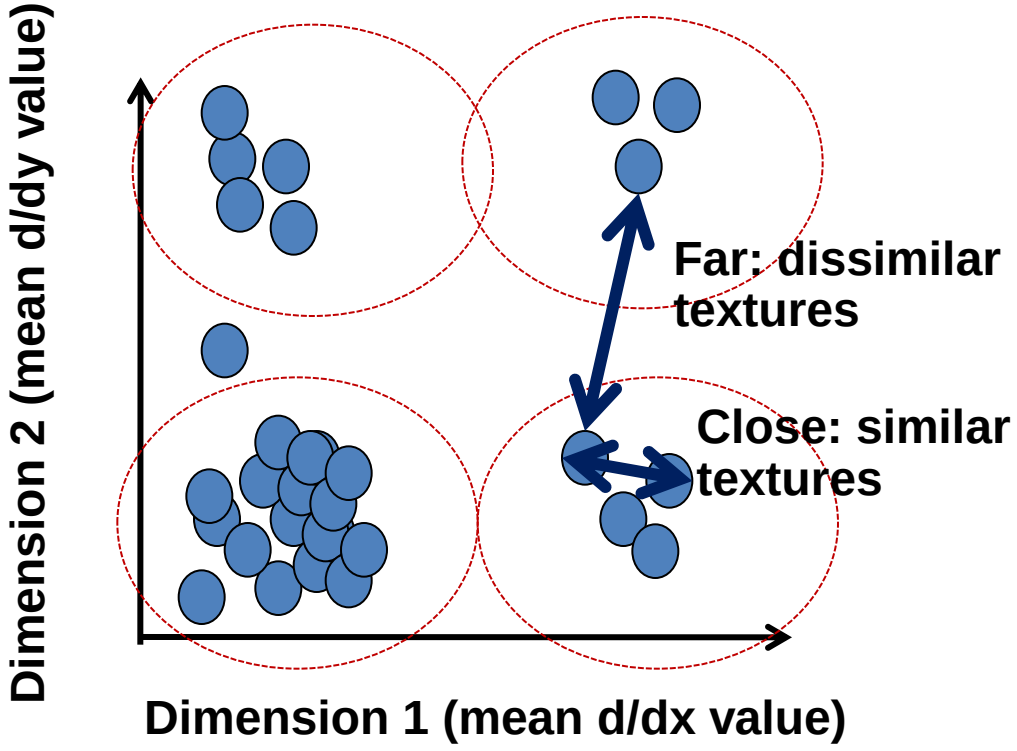


derivative filter responses, squared



visualization of the assignment to texture “types”

Texture representation: example



	mean d/dx value	mean d/dy value
Win. #1	4	10
Win.#2	18	7
⋮		
Win.#9	20	20
	⋮	

statistics to summarize patterns in small windows

Filter banks

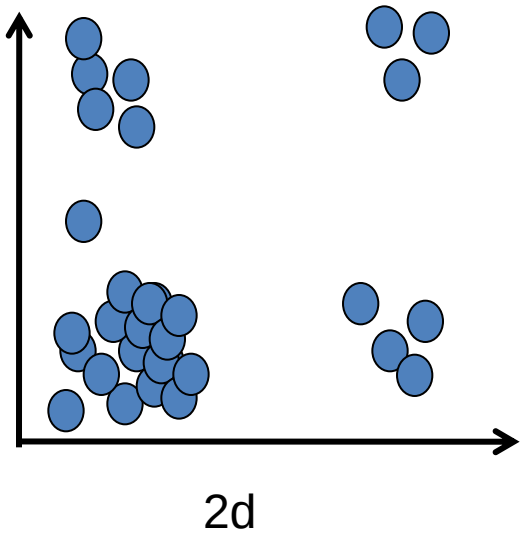
- Our previous example used two filters, and resulted in a 2-dimensional feature vector to describe texture in a window.
  - x and y derivatives revealed something about local structure.
- We can generalize to apply a collection of multiple (*d*) filters: a "filter bank"
- Then our feature vectors will be *d*-dimensional.
  - still can think of nearness, farness in feature space

Adapted from Kristen Grauman

Computing distances with *d*-dimensional features

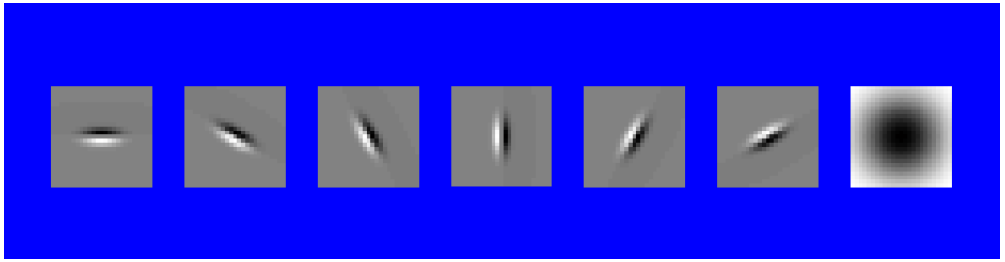
$$D(a,b) = \sqrt{\sum_{i=1}^{d} (a_i - b_i)^2}$$

Euclidean distance (L_2)

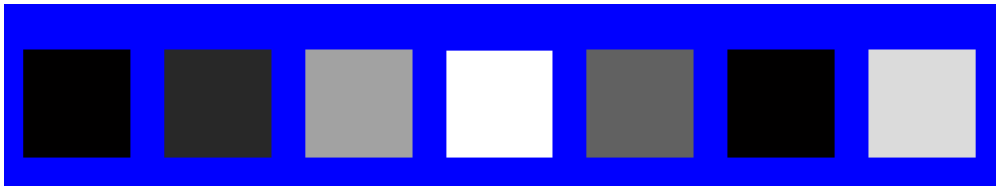


You try: Can you match the texture to the response?

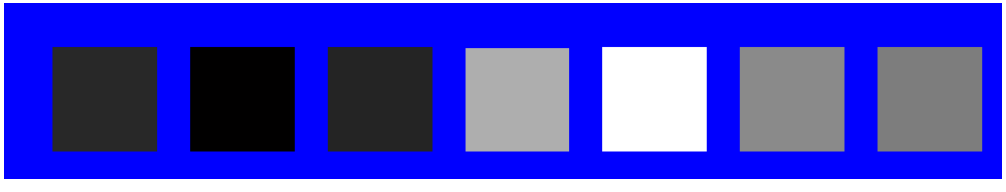
Filters



1



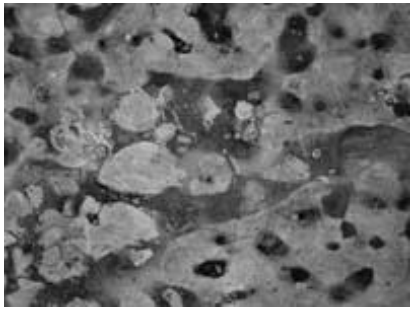
2



3

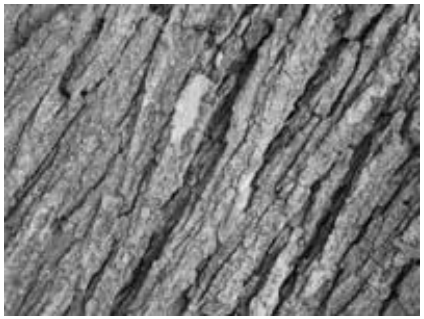
Mean abs responses

A



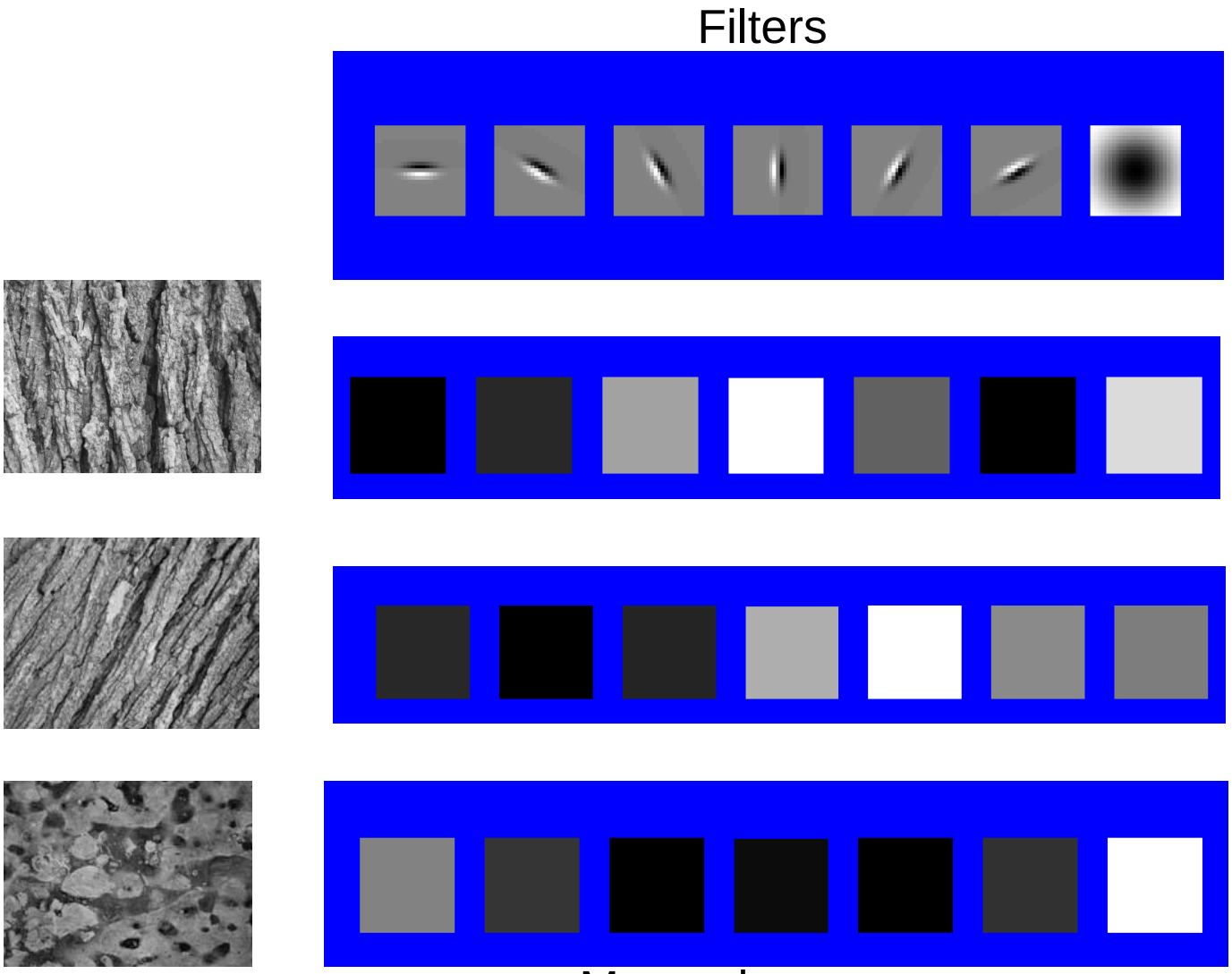
B

C



Derek Hoiem

Representing texture by mean abs response



Derek Hoiem

Classifying materials, “stuff”

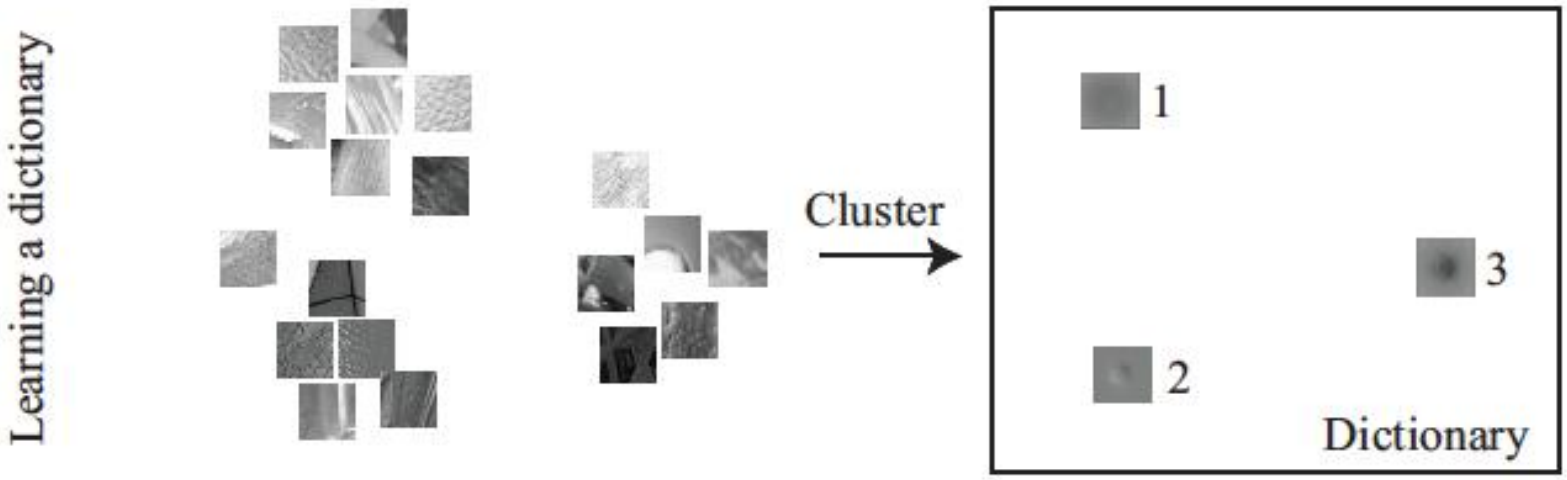
Figure by Varma & Zisserman

Example based texture representations

Q: how does one choose the filters?

- Responses to over-complete filter banks
  - build histograms over filter responses
  - describe textures using clusters of responses
- Alternative
  - build a vocabulary of pattern elements from pictures
  - describe textures using this vocabulary

Building a vocabulary



- There are 2 steps to building a pooled texture representation for texture in an image domain.
  1. Build a dictionary representing the range of possible pattern elements, using a large number of texture patches
  2. Vectorize the patches in images using the clusters learned

Building a dictionary

- Collect many training example textures
- Construct the vectors $\boldsymbol{x}$ for the relevant pixels; these could be a reshaping of a patch around the pixel or a vector of filter outputs computed at the pixel
- Obtain k cluster centers $\boldsymbol{c}$ for these examples

Representing an image domain

- For each relevant pixel pixel I in the image
  - Compute the vector representation $\boldsymbol{x_i}$ for that pixel
  - Obtain j, the index of the cluster center $\boldsymbol{c_j}$ closest to that pixel
  - Insert j into a histogram for that domain

Clustering the examples

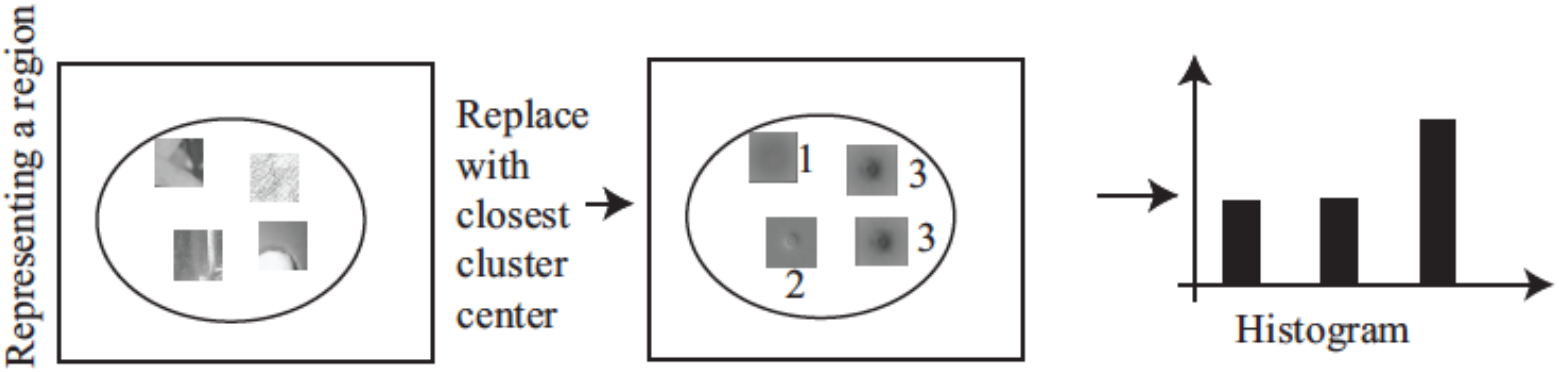
- For using k-means we can represent patches with
  - intensity vector
  - vector of filter responses over pixel/patch

Choose k data points to act as cluster centers
Until the cluster centers change very little
    Allocate each data point to cluster whose center is nearest.
    Now ensure that every cluster has at least
        one data point; one way to do this is by
        supplying empty clusters with a point chosen at random from
        points far from their cluster center.
    Replace the cluster centers with the mean of the elements
        in their clusters.
end

Algorithm 6.3: Clustering by K-Means.

Representing a region

- Vector Quantization
  - Represent a high-dimensional data item with a single number
    - Find the number of the nearest cluster center in dictionary
    - Use that to represent the cluster
- Summarize the pattern of patches
  - Cut region into patches
  - Vector quantize - vector quantized image patches are sometimes called visual words



Build a dictionary:

Collect many training example textures

Construct the vectors $\boldsymbol{x}$ for relevant pixels; these could be a reshaping of a patch around the pixel, a vector of filter outputs computed at the pixel, or the representation of Section 6.1.

Obtain k cluster centers $\boldsymbol{c}$ for these examples

Represent an image domain:

For each relevant pixel i in the image

Compute the vector representation $\boldsymbol{x}_i$ of that pixel

Obtain j, the index of the cluster center $\boldsymbol{c}_j$ closest to that pixel

Insert j into a histogram for that domain

Algorithm 6.2: Texture Representation Using Vector Quantization.

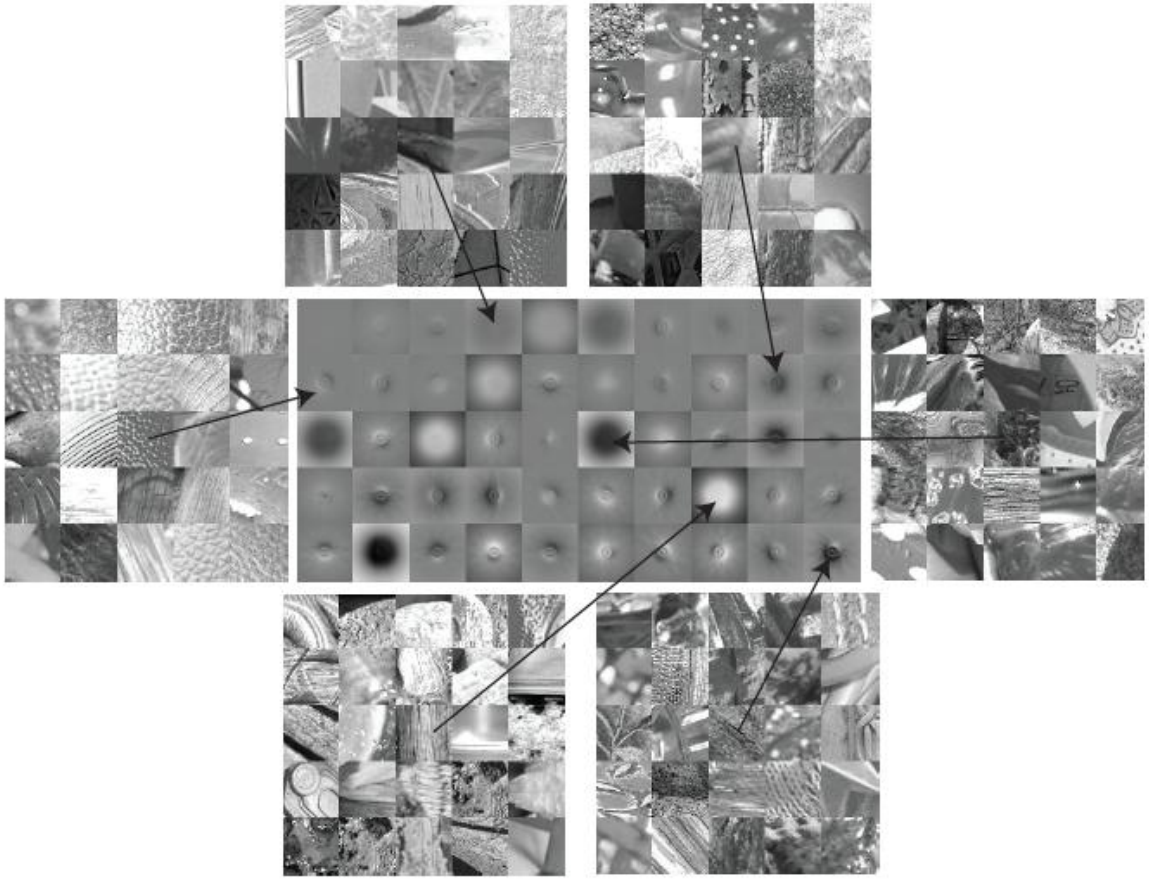


FIGURE 6.9: Pattern elements can be identified by vector quantizing vectors of filter outputs, using k-means. Here we show the top 50 pattern elements (or textons), obtained from all 1,000 images of the collection of material images described in Figure 6.2. These were filtered with the complete set of oriented filters from Figure 6.4. Each subimage here illustrates a cluster center. For each cluster center, we show the linear combination of filter kernels that would result in the set of filter responses represented by the cluster center. For some cluster centers, we show the 25 image patches in the training set whose filter representation is closest to the cluster center. *This figure shows elements of a database collected by C. Liu, L. Sharan, E. Adelson, and R. Rosenholtz, and published at* `http://people.csail.mit.edu/lavanya/research_sharan.html`. *Figure by kind permission of the collectors.*

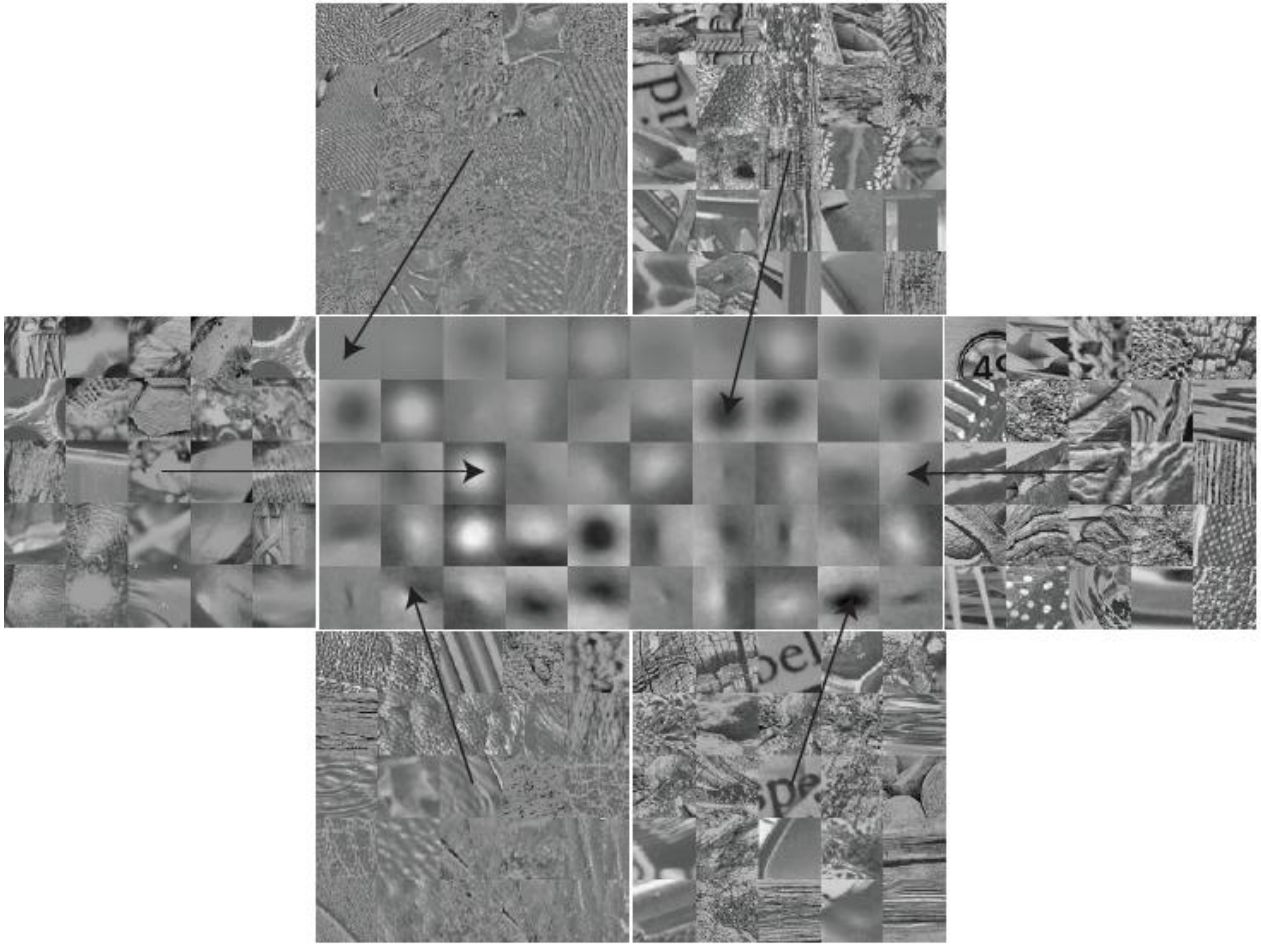


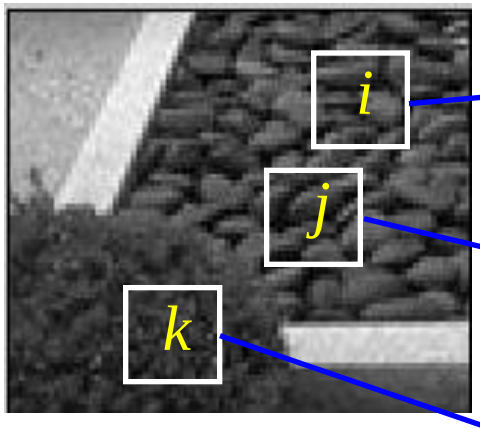
FIGURE 6.10: Pattern elements can also be identified by vector quantizing vectors obtained by reshaping an image window centered on each pixel. Here we show the top 50 pattern elements (or textons), obtained using this strategy from all 1,000 images of the collection of material images described in Figure 6.2. Each subimage here illustrates a cluster center. For some cluster centers, we show the closest 25 image patches. To measure distance, we first subtracted the average image intensity, and we weighted by a Gaussian to ensure that pixels close to the center of the patch were weighted higher than those far from the center. *This figure shows elements of a database collected by C. Liu, L. Sharan, E. Adelson, and R. Rosenholtz, and published at* `http://people.csail.mit.edu/lavanya/research_sharan.html`. *Figure by kind permission of the collectors.*

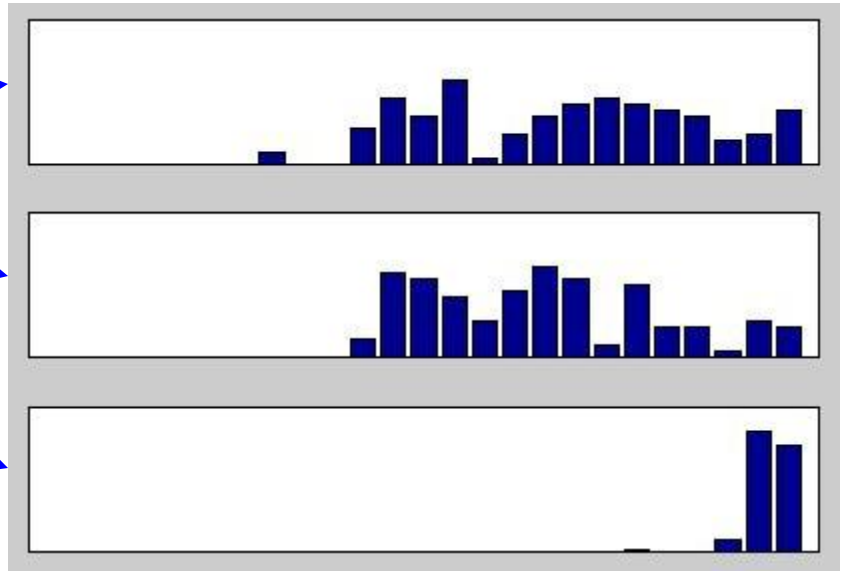
How/why to compare

- Simplest comparison is SSD（ Sum of Squared Differences ）, many others.
- Can view probabilistically.
  - Histogram is a set of samples from a probability distribution.
  - With many samples it approximates distribution.
  - Test probability samples drawn from same distribution. Ie., is difference greater than expected when two samples come from same distribution?

Chi square distance between texton histograms

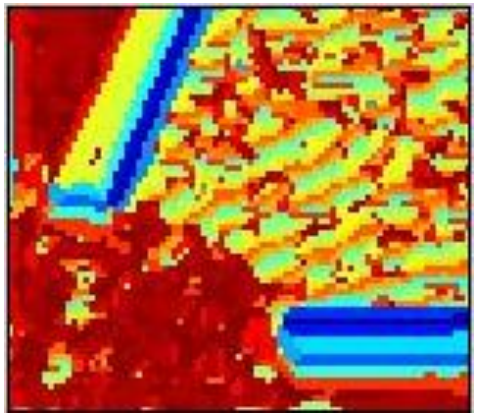
Chi-square





0.1

0.8



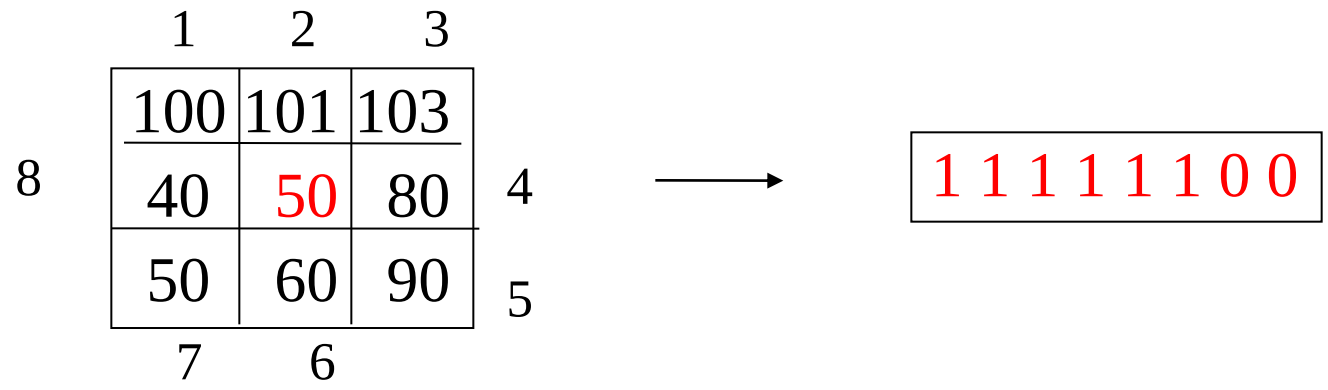
$$\chi^2(h_i, h_j) = \frac{1}{2}\sum_{m=1}^{K}\frac{[h_i(m) - h_j(m)]^2}{h_i(m) + h_j(m)}$$

(Malik)

Local Binary Pattern Measure

- For each pixel p, create an 8-bit number $b_1\ b_2\ b_3\ b_4\ b_5\ b_6\ b_7\ b_8$, where $b_i = 0$ if neighbor i has value less than or equal to p's value and 1 otherwise.

- Represent the texture in the image (or a region) by the histogram of these numbers.

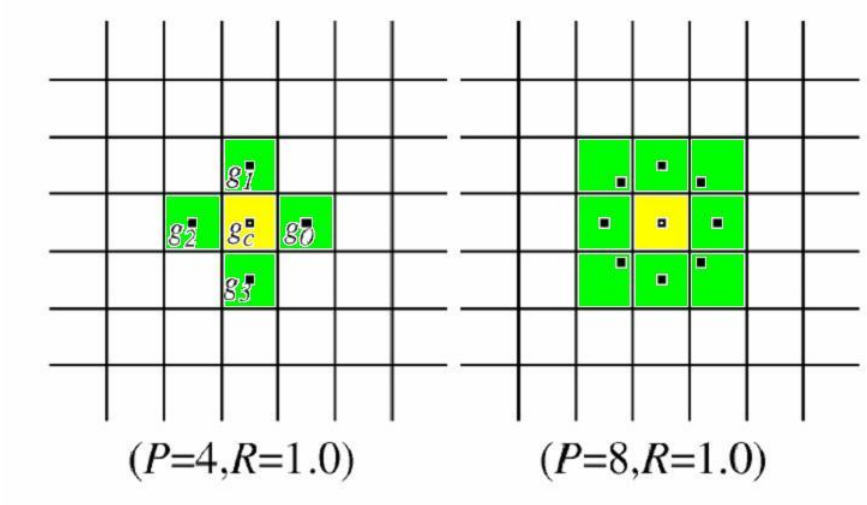


Ojala, T., Pietik ¨ ainen, M., Harwood, D.:
A comparative study of texture measures with classification based on feature distributions. Pattern Recognition **29** (1996) 51–59

Local Binary Pattern (LBP)

The primitive LBP (P,R) number that characterizes the spatial structure of the local image texture is defined as:

$$LBP_{P,R} = \sum_{p=0}^{P-1} s(x)2^p \;,\; x = g_p - g_c \qquad \text{where ,} \qquad s(x) = \begin{cases} 1 \; if \;\; x \geq 0 \\ 0 \; if \;\; x < 0 \end{cases}$$



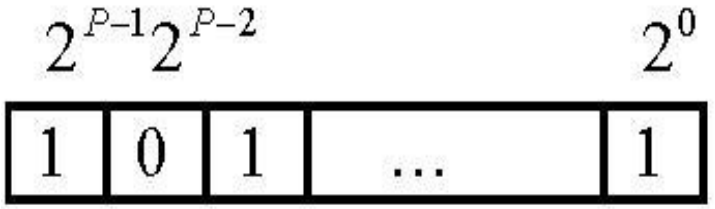
Circularly symmetric neighbor sets (P: angular resolution, R: spatial resolution)

2^7	2^0	2^1
2^6	g_c	2^2
2^5	2^4	2^3

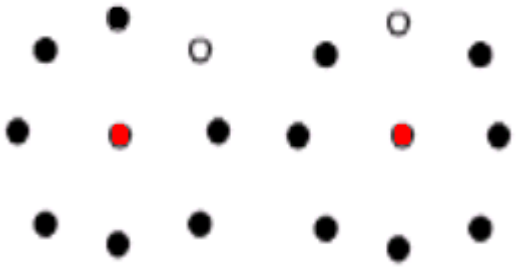
LBP values in a 3 x 3 block

Rotation Invariant LBP

- The LBP(P,R) operator produces 2^P different patterns.



- When the image is rotated, the pixels values will correspondingly move along the perimeter of the circle around.



Rotation Invariant LBP …

- In order to remove the effect of rotation and assign a unique identifier to each, Rotation Invariant Local Binary Pattern is defined as:

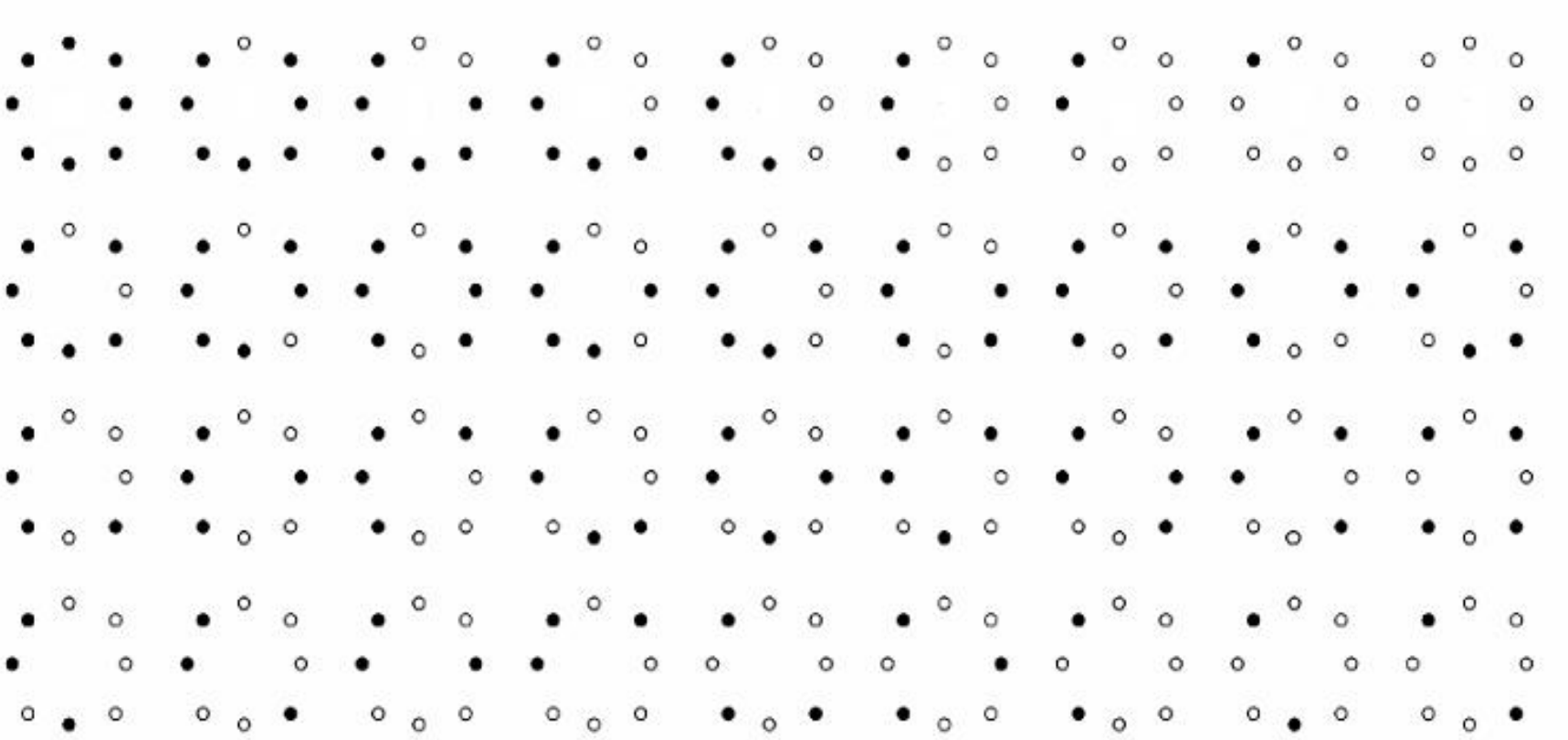
$$LBP_{P,R}^{ri} = \min\left\{ ROR(LBP_{P,R}, i) \quad \middle| \quad i = 0,1,..., P-1 \right\}$$

where *ROR(x,i)* performs a circular bit-wise right shift on P-bit number x , i time.

- 36 unique rotation invariant binary patterns can occur in the circularly symmetric neighbor set of $LBP_{8,1}$.

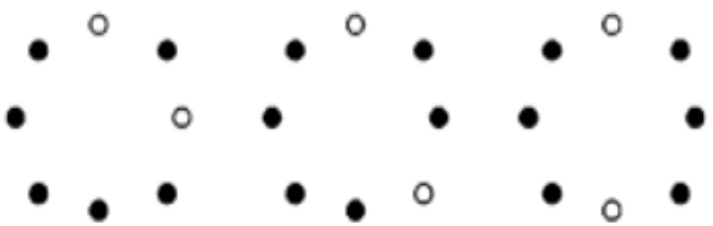
Rotation Invariant LBP …

- This figure shows 36 unique rotation invariant binary patterns.

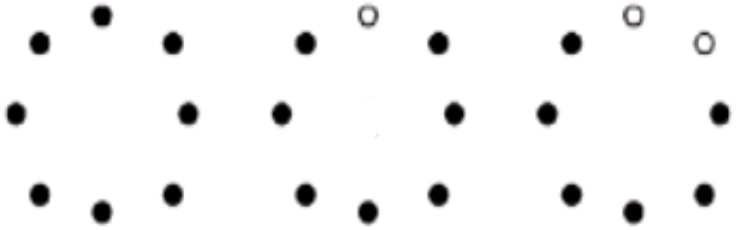


Rotation Invariant LBP …

- Rotation Invariant LBP patterns include:
  - Uniform patterns
    - At most two transitions from 0 to 1
  - Non-uniform patterns
    - More than two transitions from 0 to 1



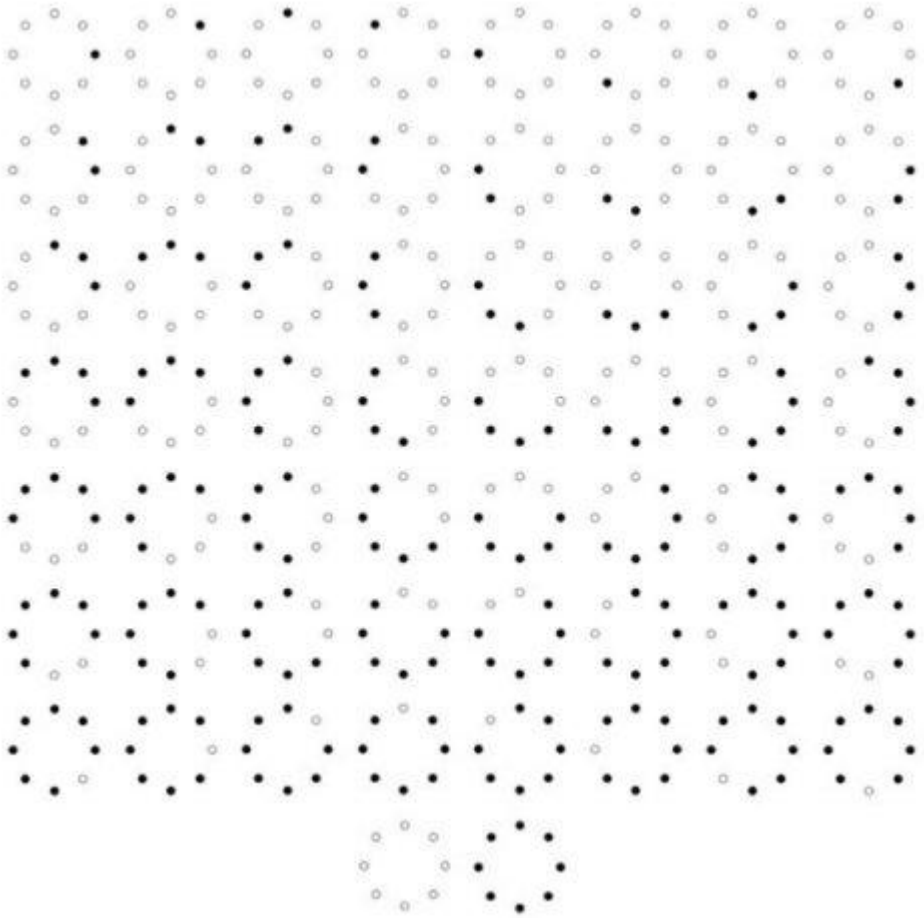
Samples of non-uniform patterns



Samples of uniform patterns

Uniform patterns

- This figure shows 58 of uniform patterns of binary patterns.



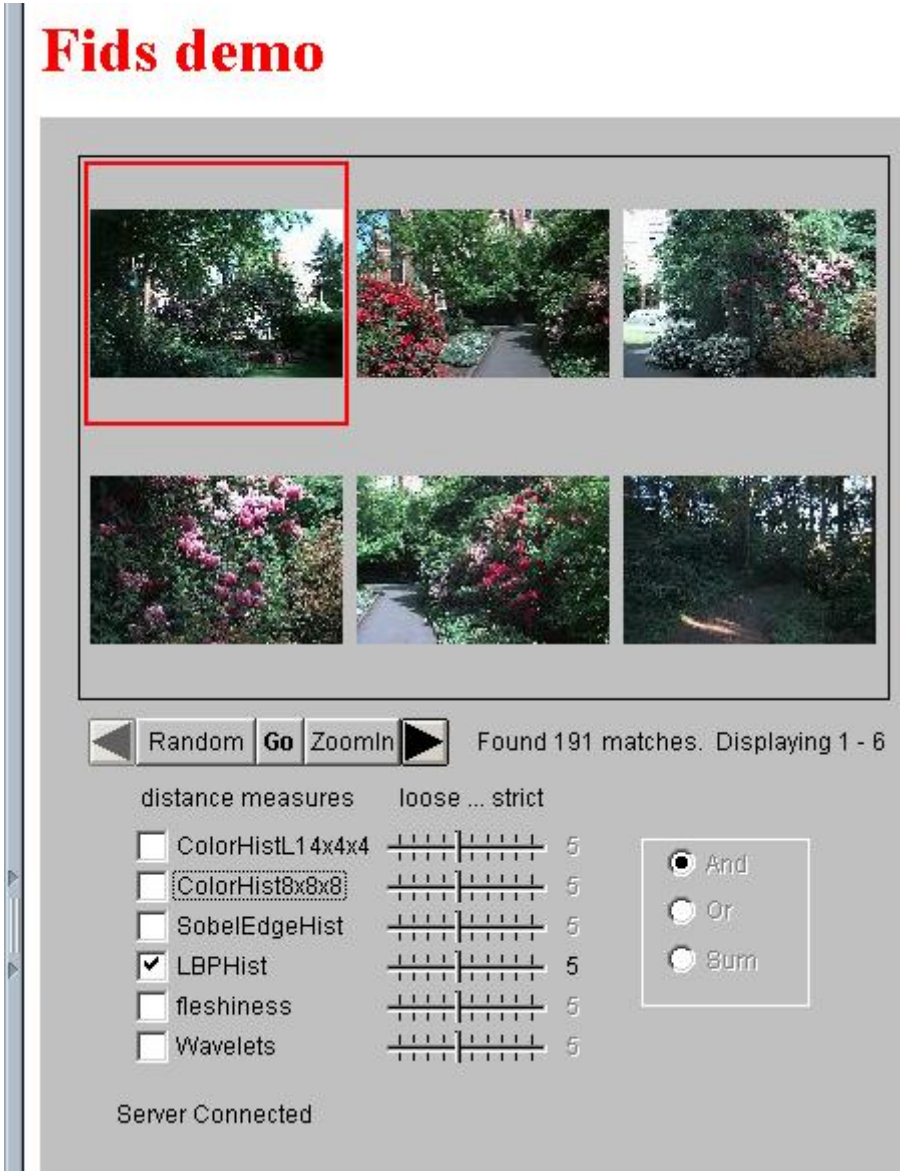
Uniform LBP (ULBP)

- It is observed that the uniform patterns are the majority, sometimes over 90 percent, of all 3 x 3 neighborhood pixels present in the observed textures.
- They function as templates for microstructures such as :
  - Bright spot (0)
  - Flat area or dark spot (8)
  - Edges of varying positive and negative curvature (1-7)

9 Rotation Invariant Uniform Local Binary Patterns

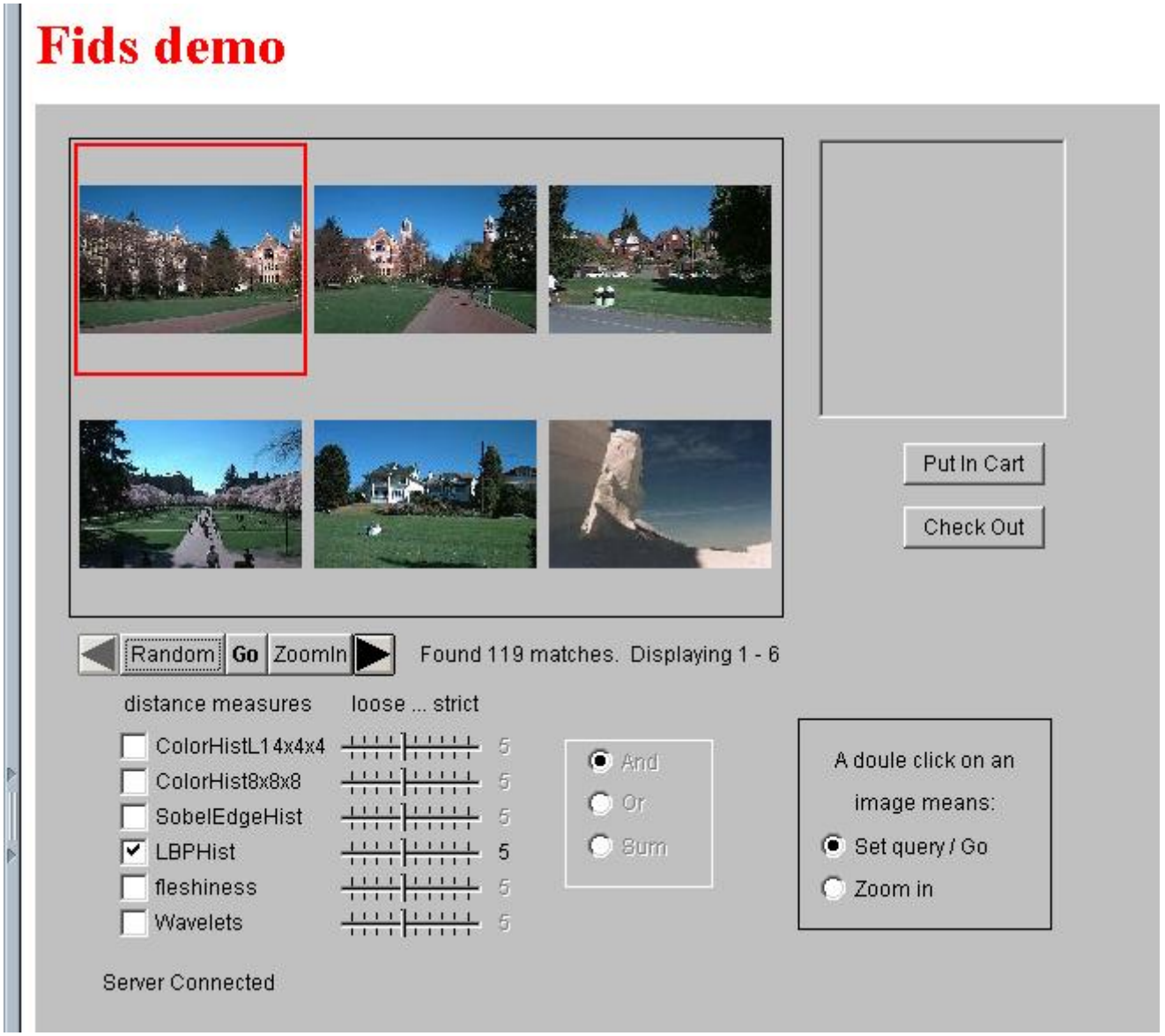
Example

Fids (Flexible Image Database System) is retrieving images similar to the query image using LBP texture as the texture measure and comparing their LBP histograms



Example

Low-level measures don't always find semantically similar images.



Texture representations

- A vector summarizing the trends in pattern elements
  - either overall trend in filter responses
  - or histogram of vector quantized patches

- At a pixel
  - compute representations for domains centered on the pixel

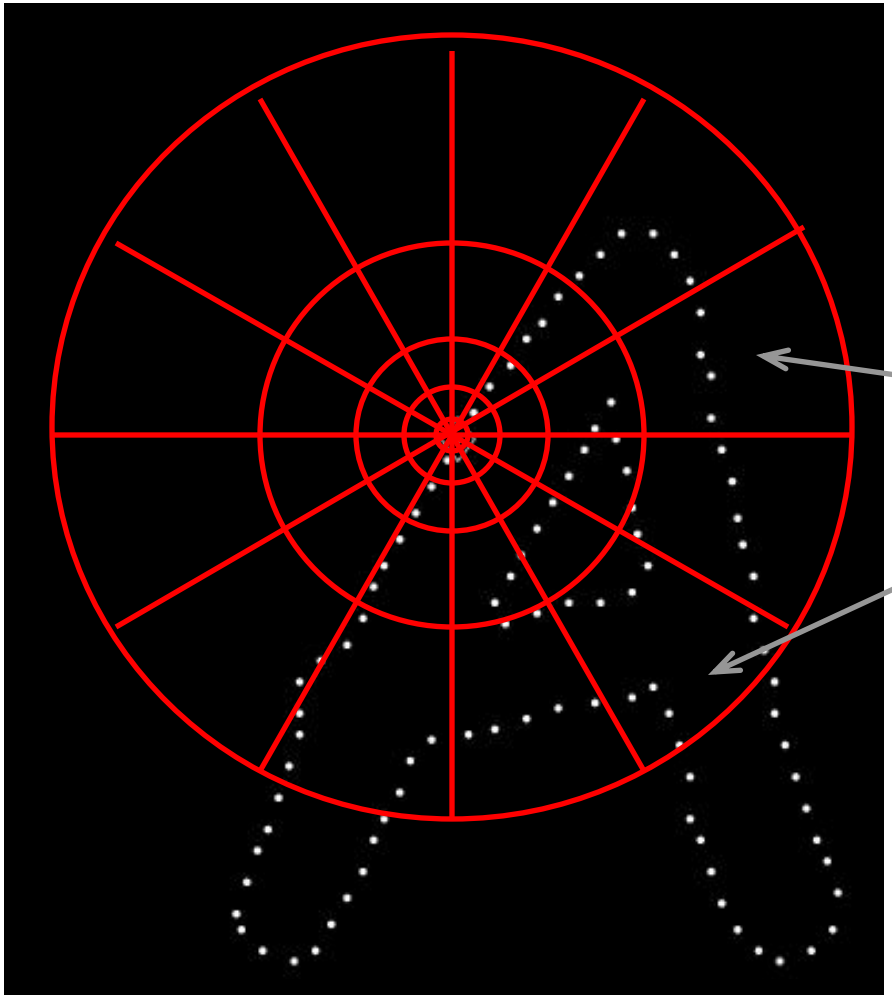
- For a region
  - compute representations the whole region

Local Image Features

- Properties of detectors
  - Edge detectors
  - Corners
  - Blobs
  - Scale invariant detection
- Properties of descriptors
  - HOG
  - SIFT
  - Color
  - Texture
  - Shape context

Shape context, Geometric Blur and more

Local Descriptors: Shape Context



Count the number of points inside each bin, e.g.:

Count = 4

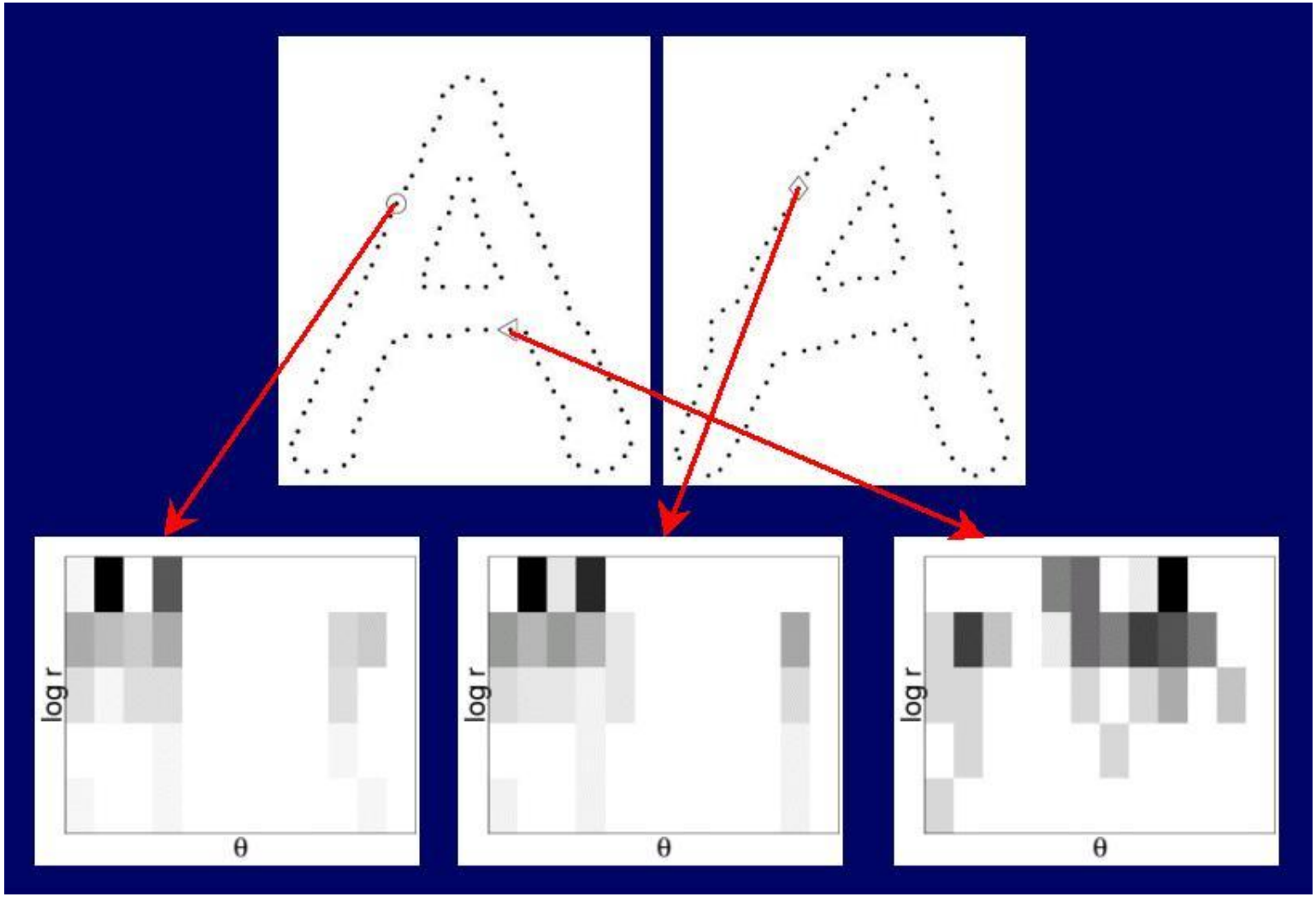
⋮

Count = 10

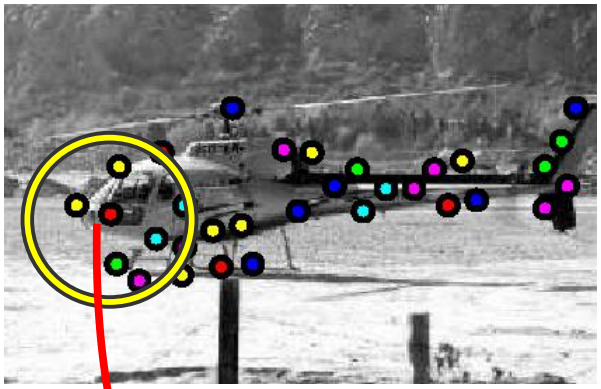
Log-polar binning: more precision for nearby points, more flexibility for farther points.

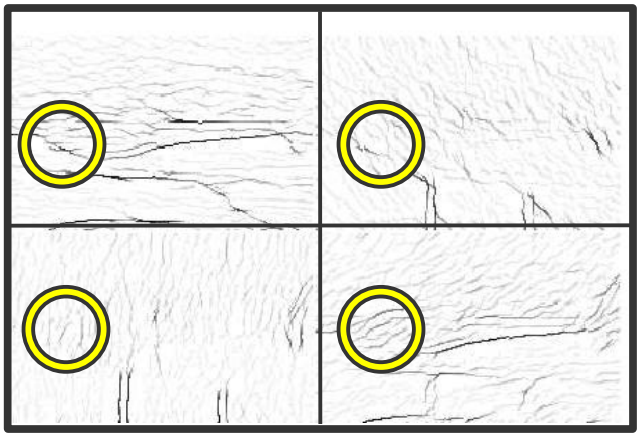
Belongie & Malik, ICCV 2001

Shape Context Descriptor



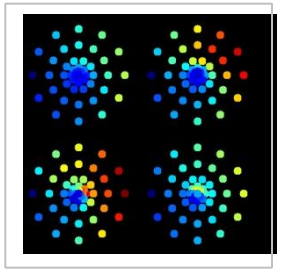
Local Descriptors: Geometric Blur



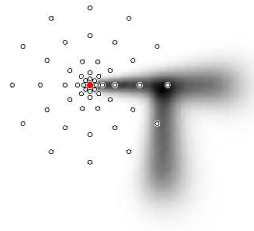


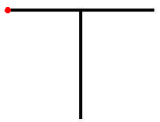
Compute edges at four orientations

Extract a patch in each channel



Example descriptor





(Idealized signal)

Apply spatially varying blur and sub-sample

- Berg & Malik, CVPR 2001

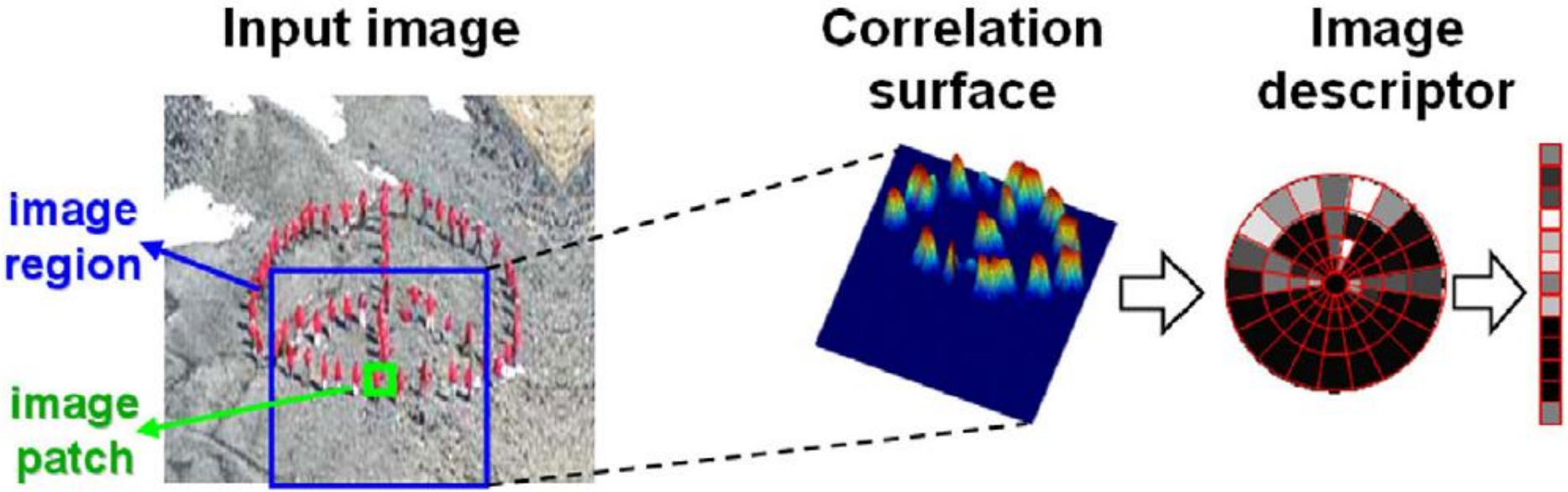
- K. Grauman, B. Leibe

Self-similarity Descriptor

Figure 1. *These images of the same object (a heart) do NOT share common image properties (colors, textures, edges), but DO share a similar geometric layout of local internal self-similarities.*

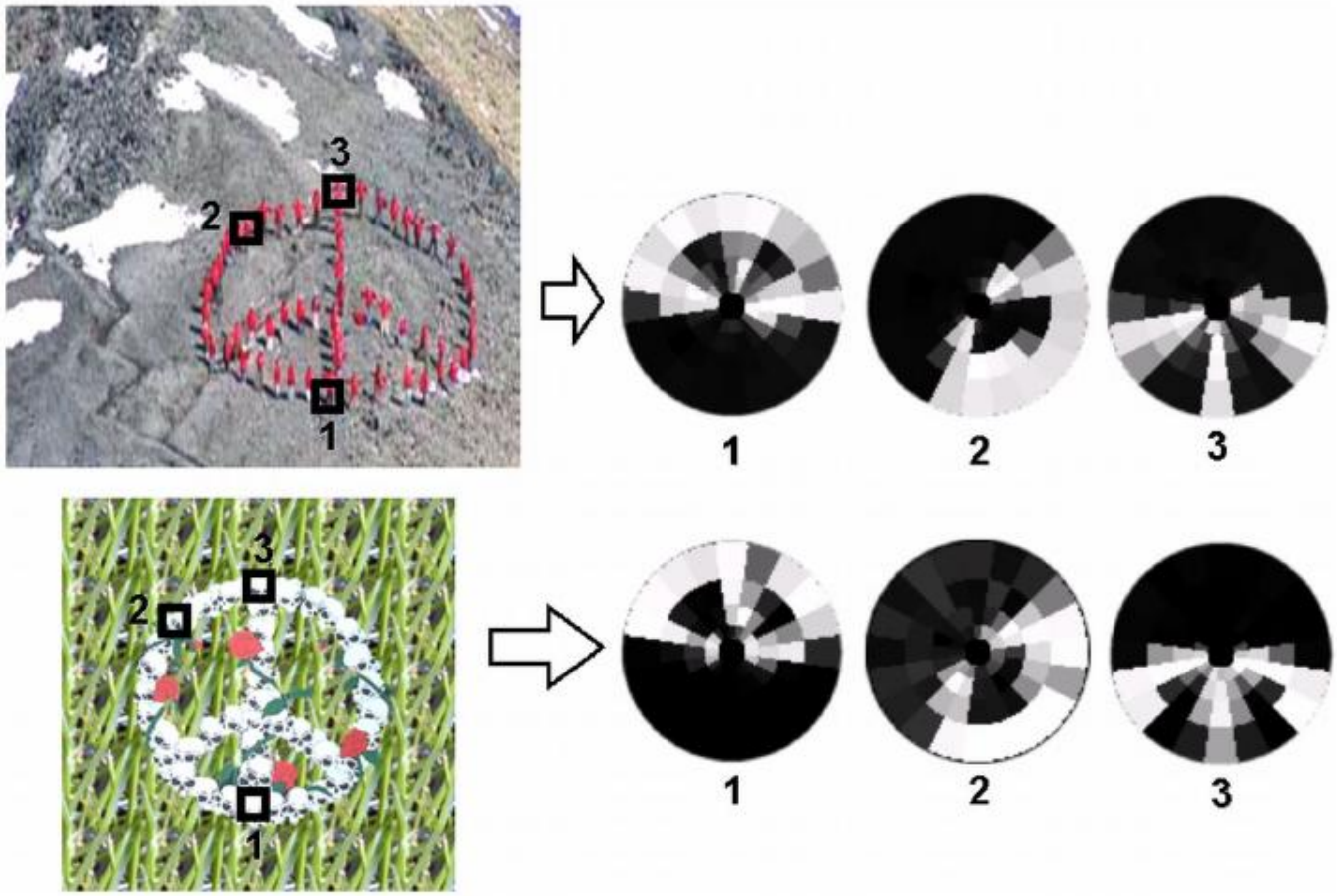
Matching Local Self-Similarities across Images and Videos, Shechtman and Irani, 2007

Self-similarity Descriptor



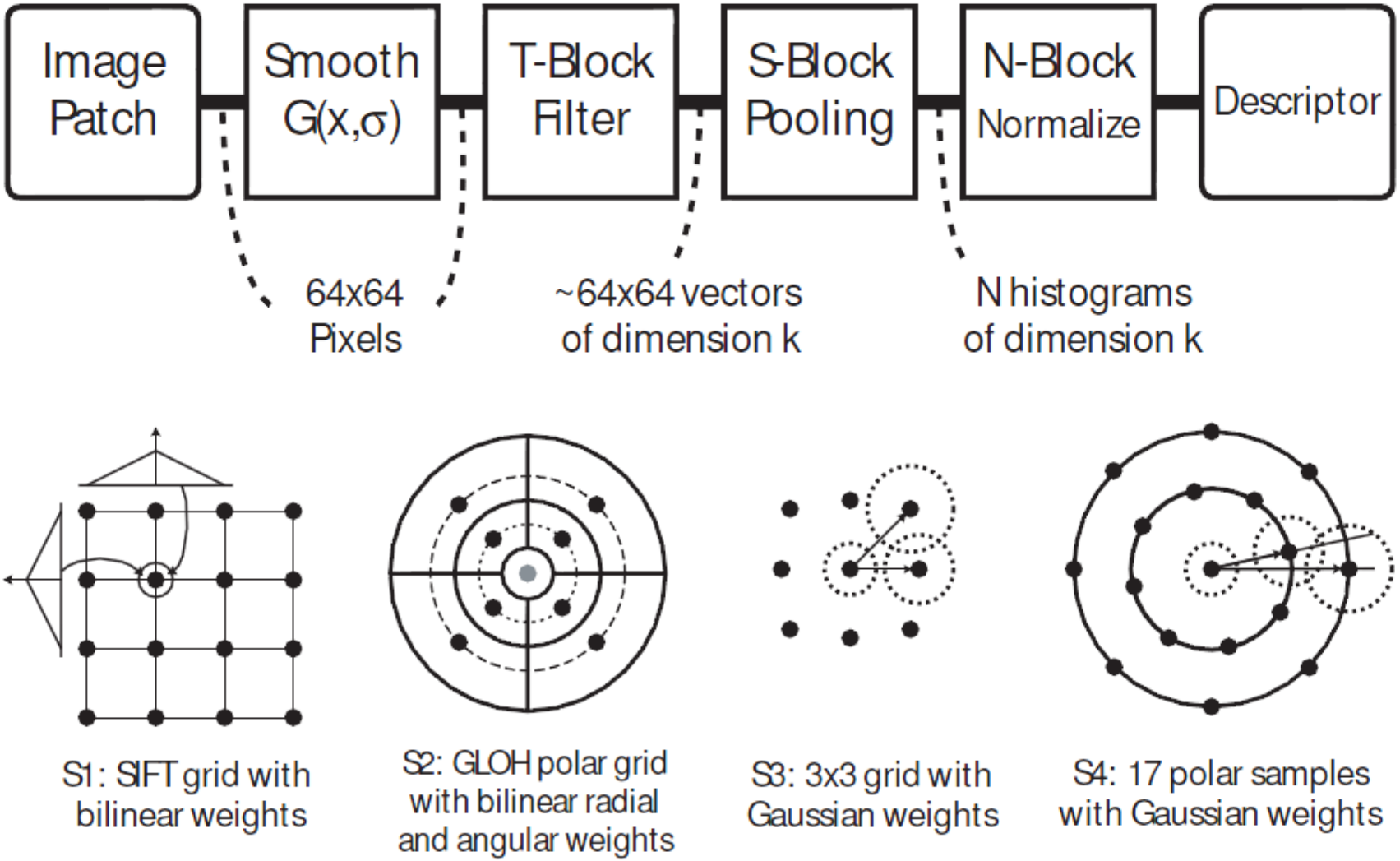
Matching Local Self-Similarities across Images and Videos, Shechtman and Irani, 2007

Self-similarity Descriptor



Matching Local Self-Similarities across Images and Videos, Shechtman and Irani, 2007

Learning Local Image Descriptors, Winder and Brown, 2007

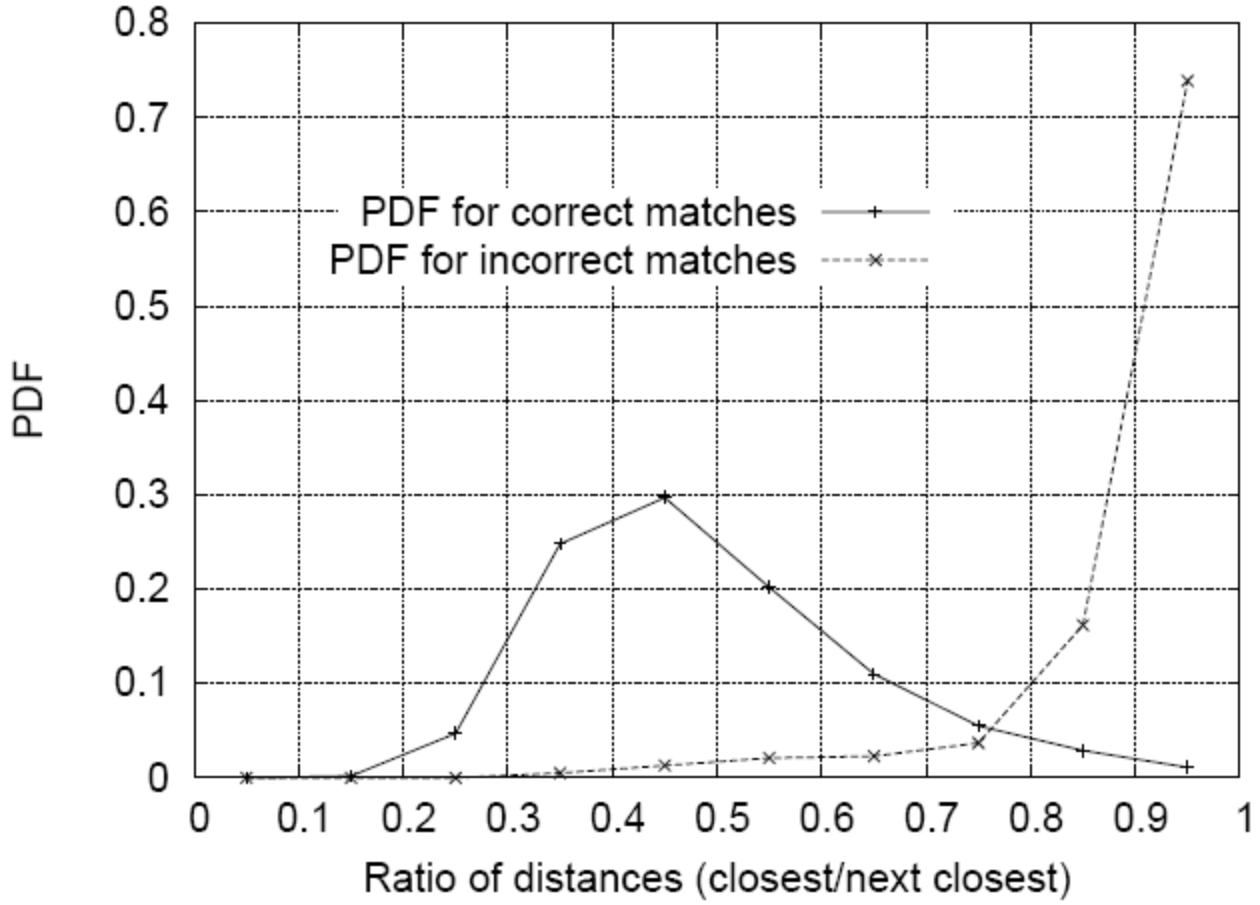


Local Descriptors

- Most features can be thought of as templates, histograms (counts), or combinations
- The ideal descriptor should be
  - Robust
  - Distinctive
  - Compact
  - Efficient
- Most available descriptors focus on edge/gradient information
  - Capture texture information
  - Color rarely used

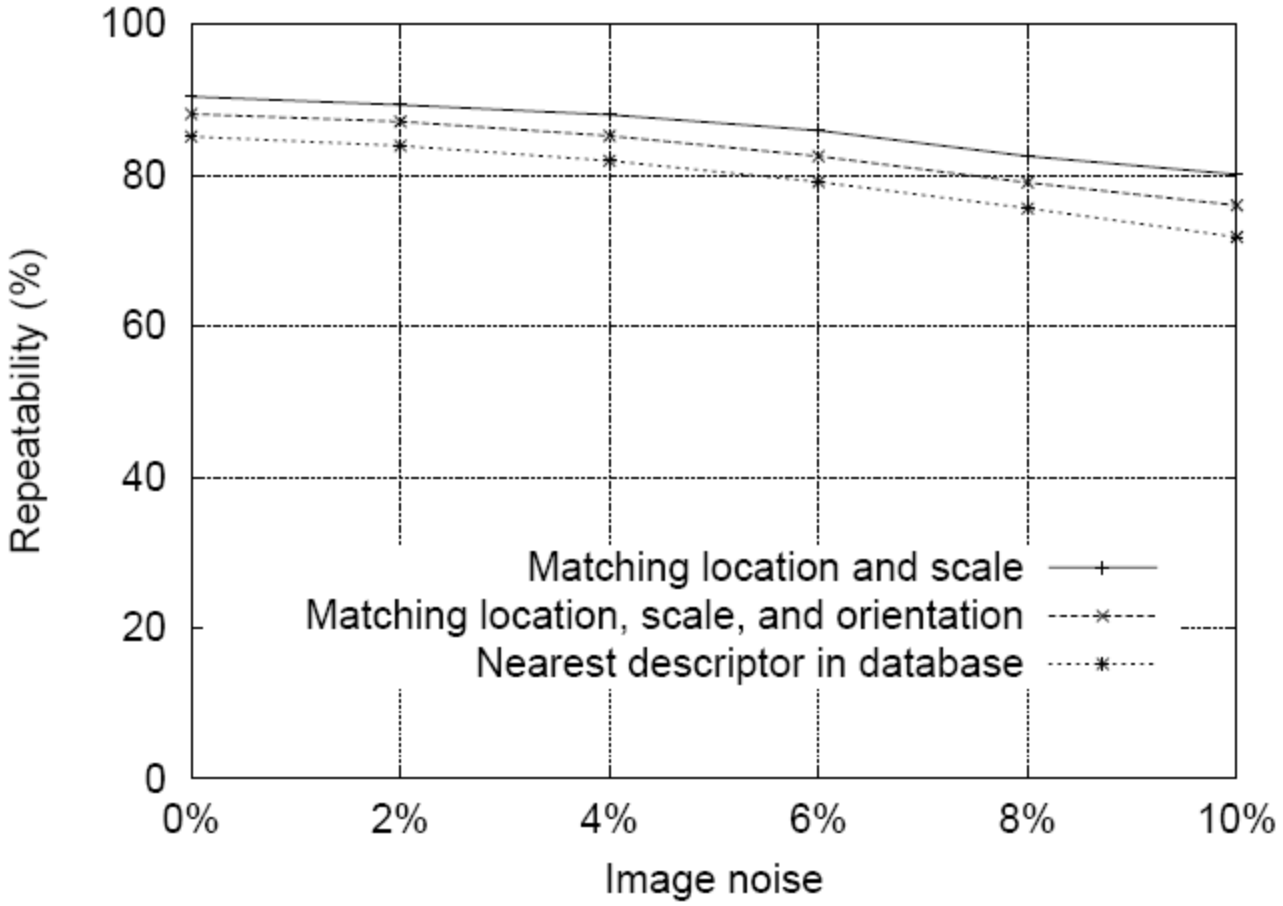
Matching Local Features

- Nearest neighbor (Euclidean distance)
- Threshold ratio of nearest to 2nd nearest descriptor



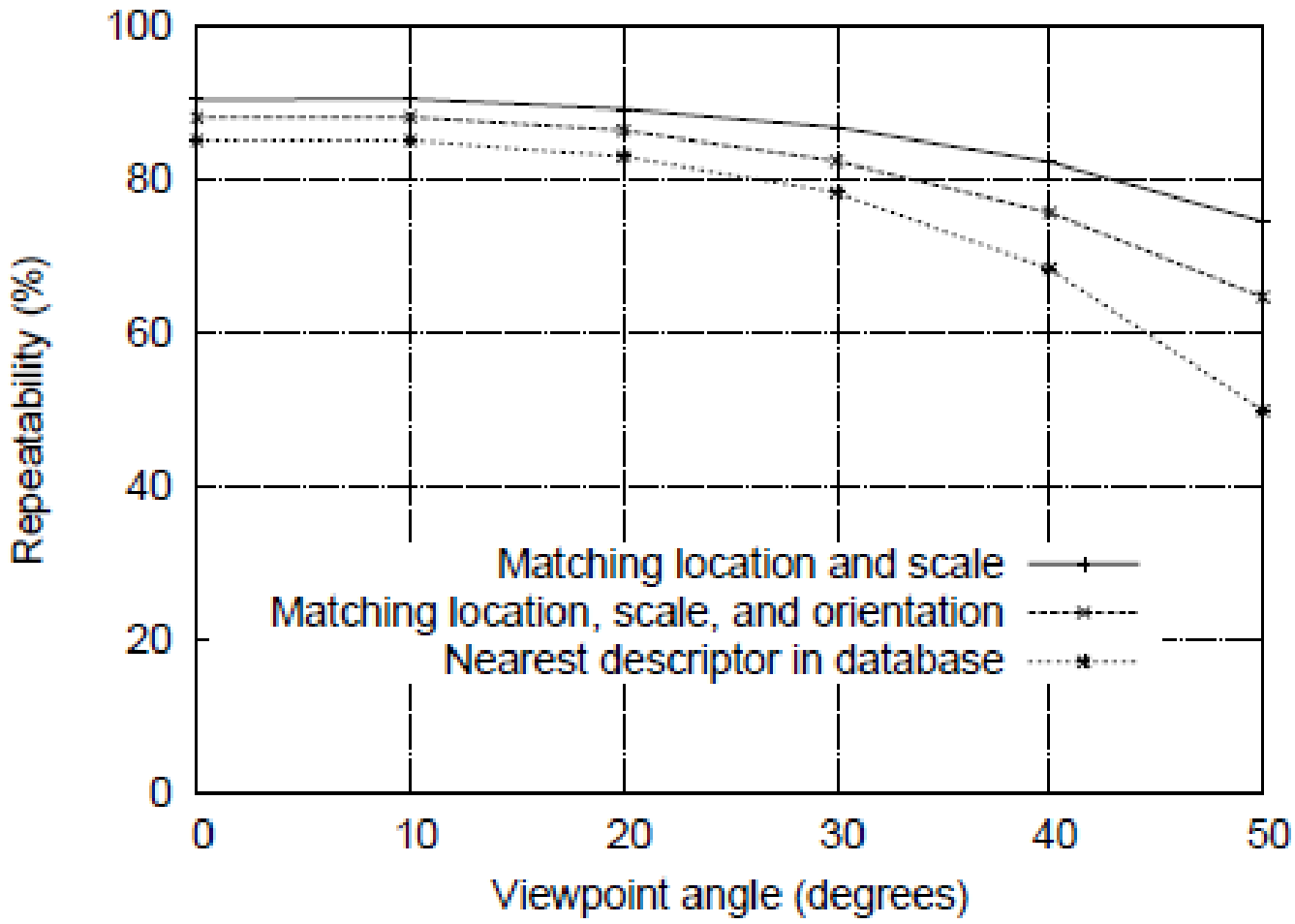
Lowe IJCV 2004

SIFT Repeatability

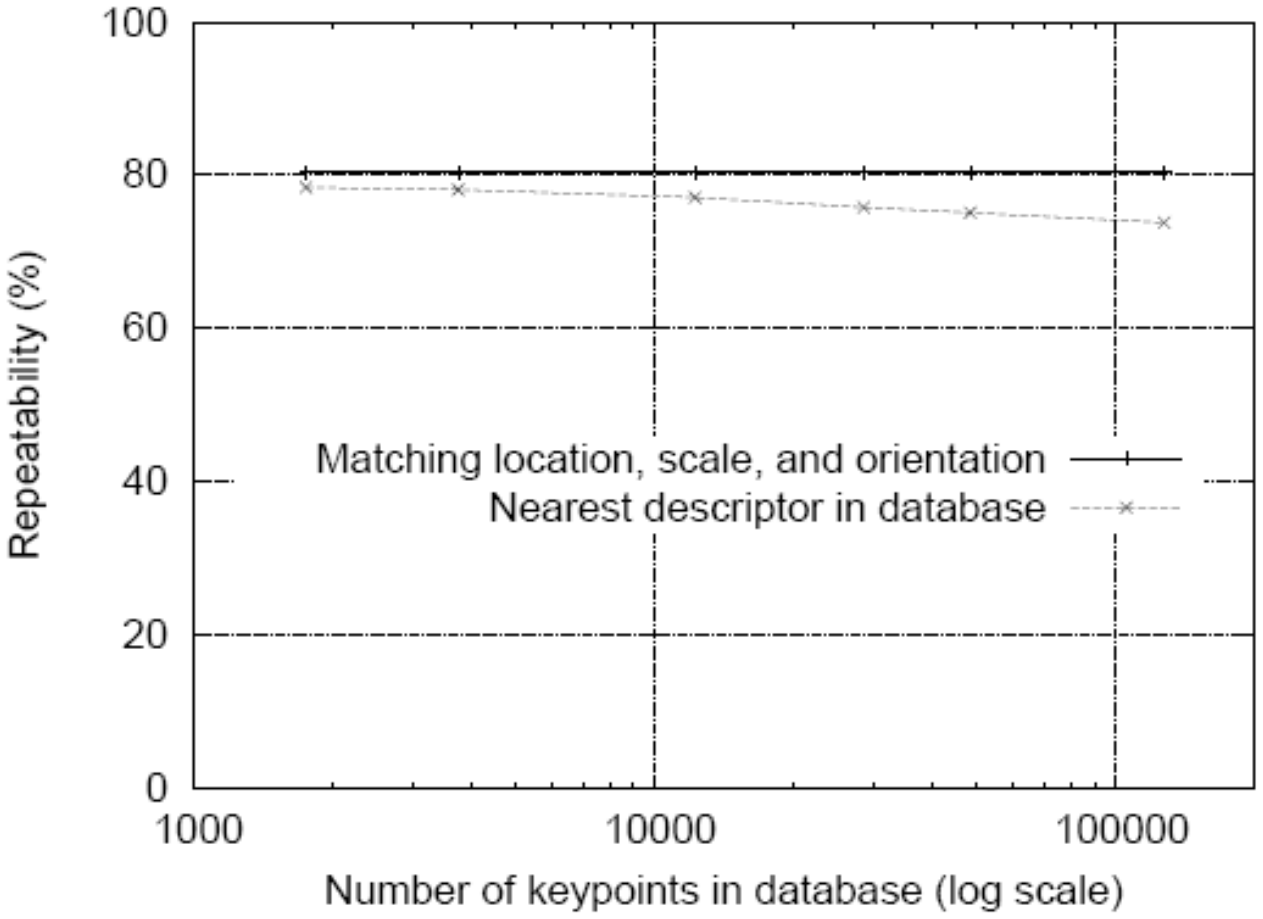


Lowe IJCV 2004

SIFT Repeatability



SIFT Repeatability



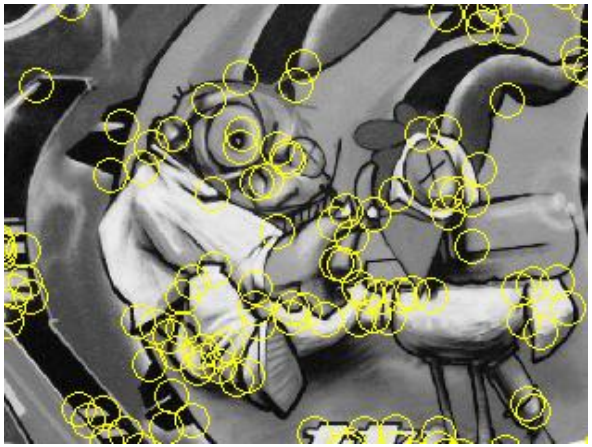
Lowe IJCV 2004

Choosing a descriptor

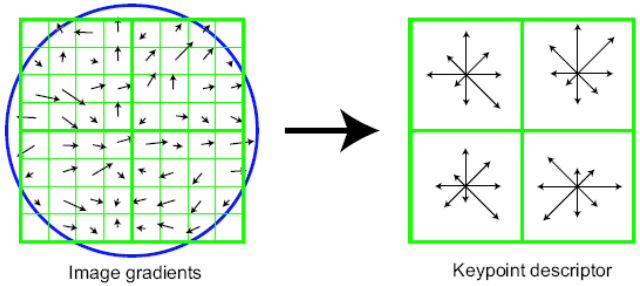
- Again, need not stick to one
- For object instance recognition or stitching, SIFT or variant is a good choice

Things to remember

- Keypoint detection: repeatable and distinctive
  - Corners, blobs, stable regions
  - Harris, DoG



- Descriptors: robust and selective
  - spatial histograms of orientation
  - SIFT



- Chapter 5.2.2, 5.3.2,5.4.2, 3.1-3 and 6.1-2

Classifying textures with filter banks：
http://www.robots.ox.ac.uk/~vgg/research/texclass/filters.html
A general LBP implementation for Matlab:
http://www.cse.oulu.fi/CMV/Downloads/LBPMatlab