

Computer Vision

CSE 455

Color and Texture

Linda Shapiro

Professor of Computer Science & Engineering

Professor of Electrical Engineering

We don't really understand vision

- Visual cortex - highly studied part of the brain
- Only rough idea of what different components do
- New discoveries in vision all the time
 - Eye uses blinking to reset its rotational orientation
 - Visual cortex can make some “high-level” decisions



Is vision easy or hard for humans?

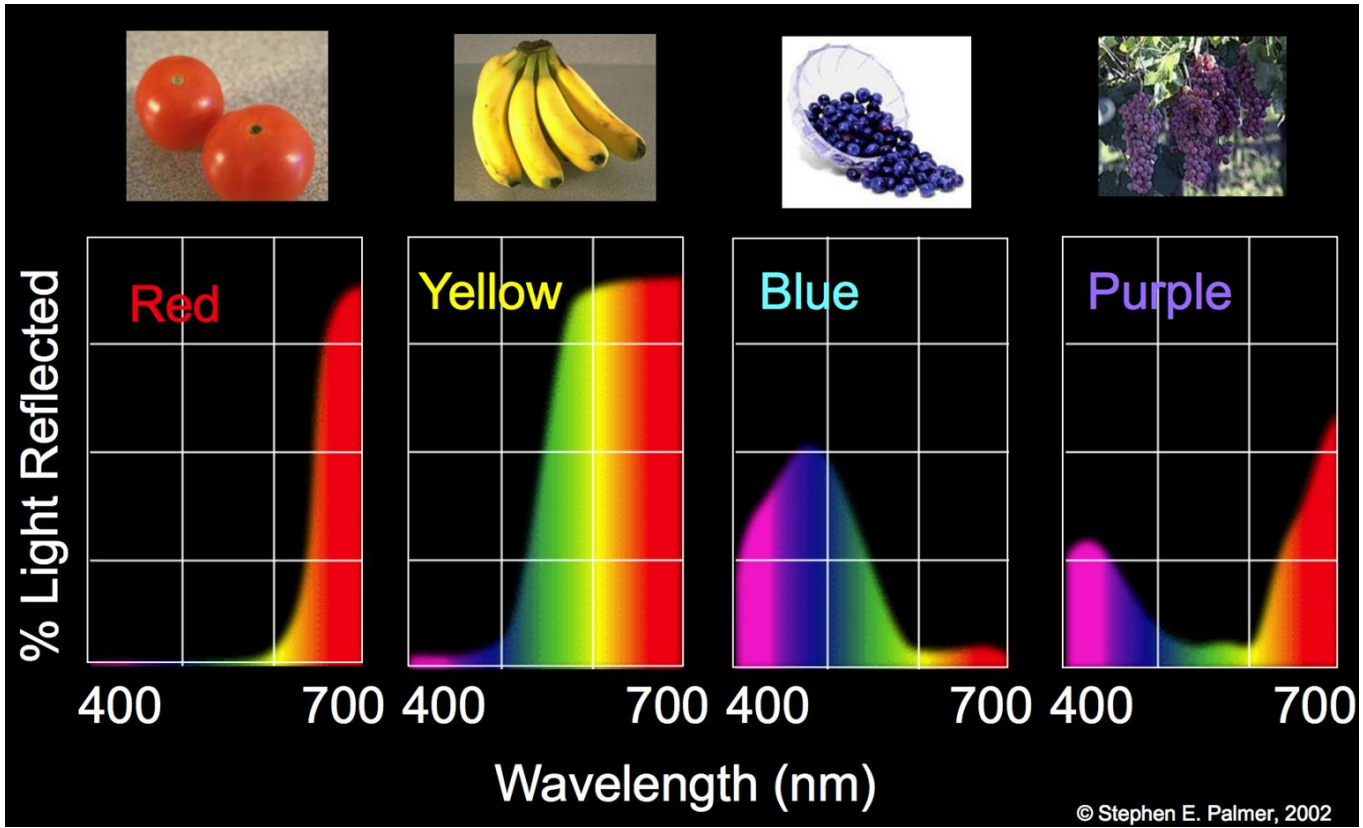
- What do you think?

Is vision easy or hard for humans?

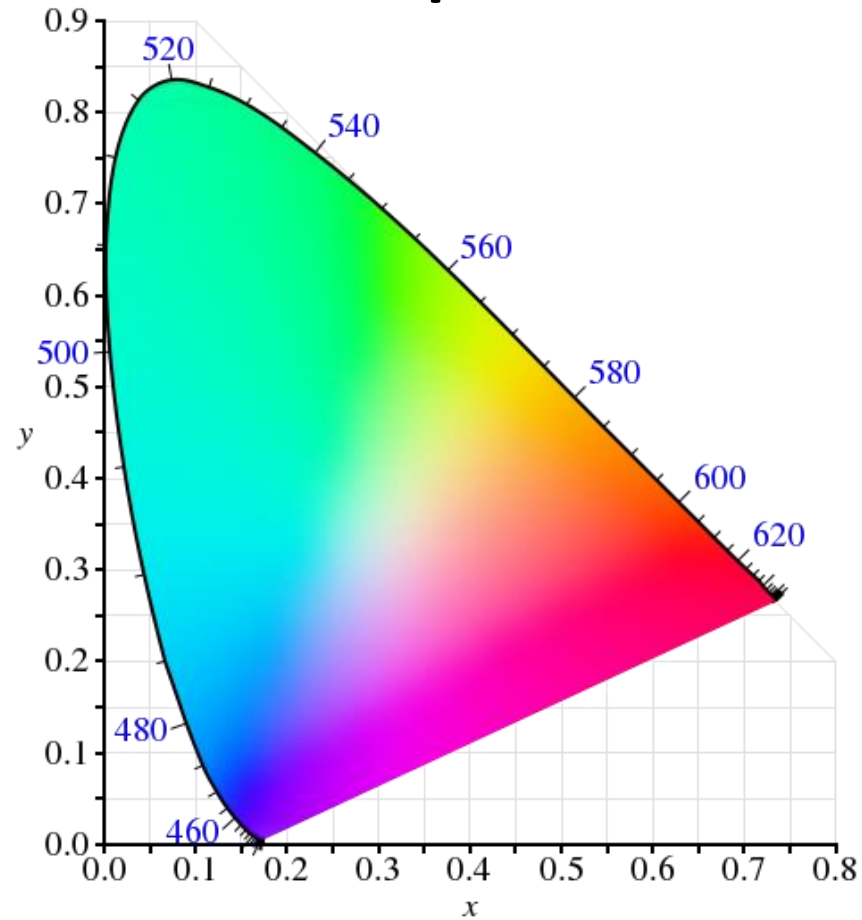
- What do you think now?



Objects reflect only some light



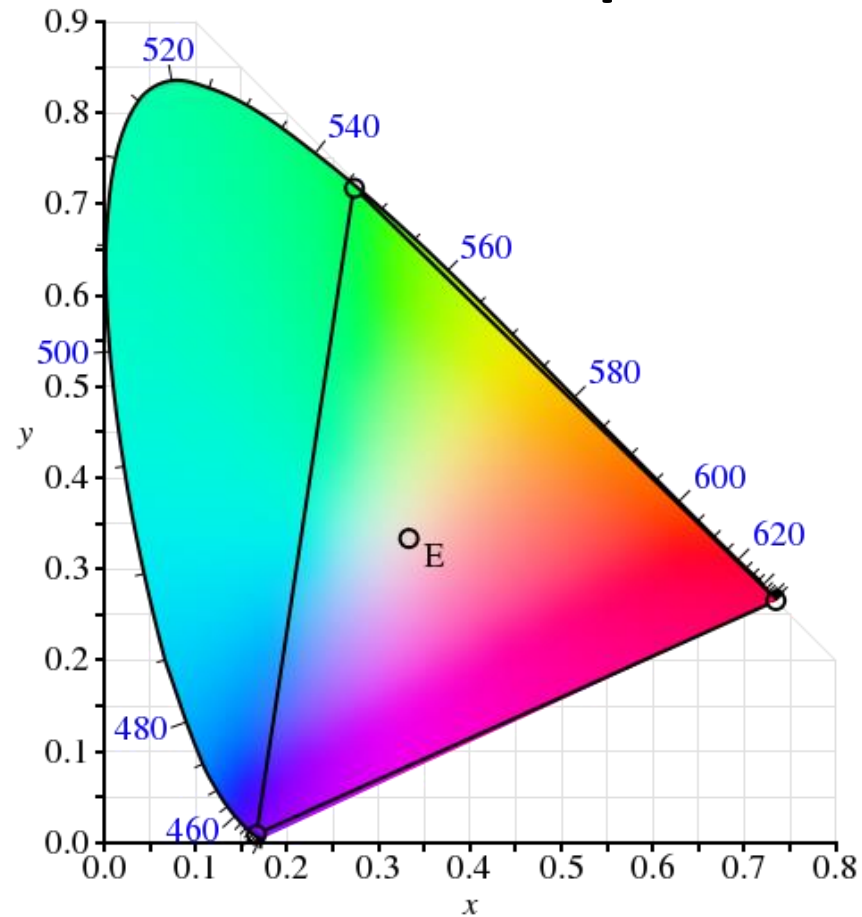
We can make a map of color



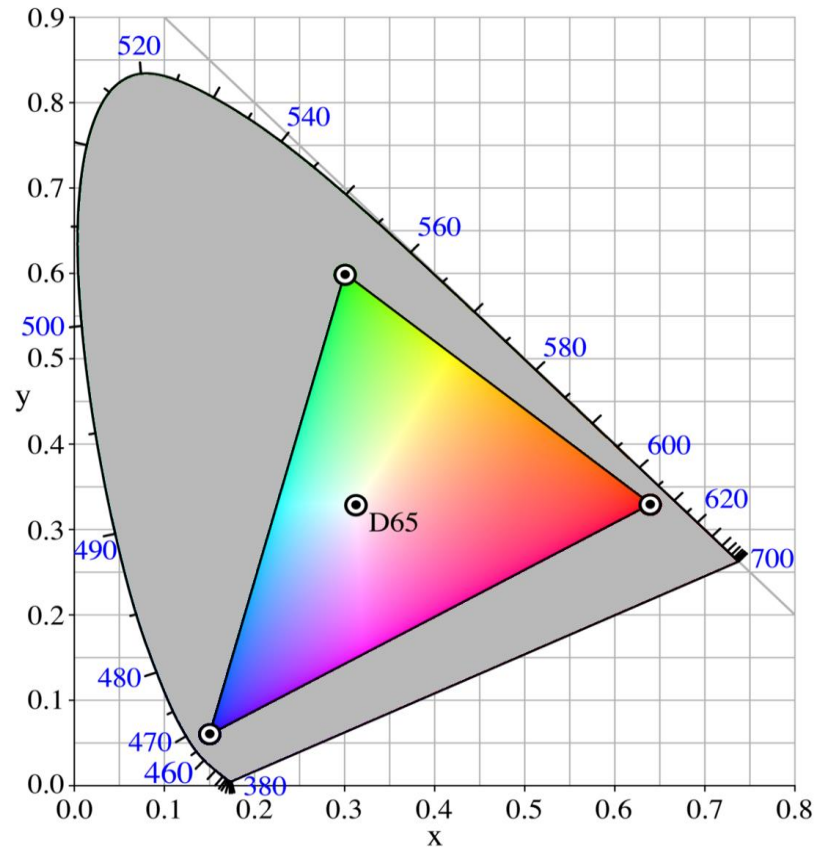
Linear colorspace

- Pick some primaries
 - Can mix those primaries to match any color inside the triangle
 - There is a commision that studies color!
-
- Commission internationale de l'éclairage (**CIE**) is a 100 year old organization that creates international standards related to light and color.

“Theoretical” CIE RGB primaries



Practical sRGB primaries, MSFT 1996



sRGB (standard Red Green Blue) is an RGB color space that HP and Microsoft created cooperatively in 1996 to use on monitors, printers, and the Internet. It was subsequently standardized by the IEC (International Electrotechnical Commission) as IEC 61966-2-1:1999.

What does this mean for computers?

- We represent images as grids of pixels
- Each pixel has a color, 3 components: RGB
- Not every color can be represented in RGB!
 - Have to go out in the real world sometimes
- RGB is not fully accurate
- We can represent a color with 3 numbers
 - #ff00ff; (1.0, 0.0, 1.0); 255,0,255
 - WHAT COLOR IS THAT?

Linda Shapiro's Schedule: Spring 2020

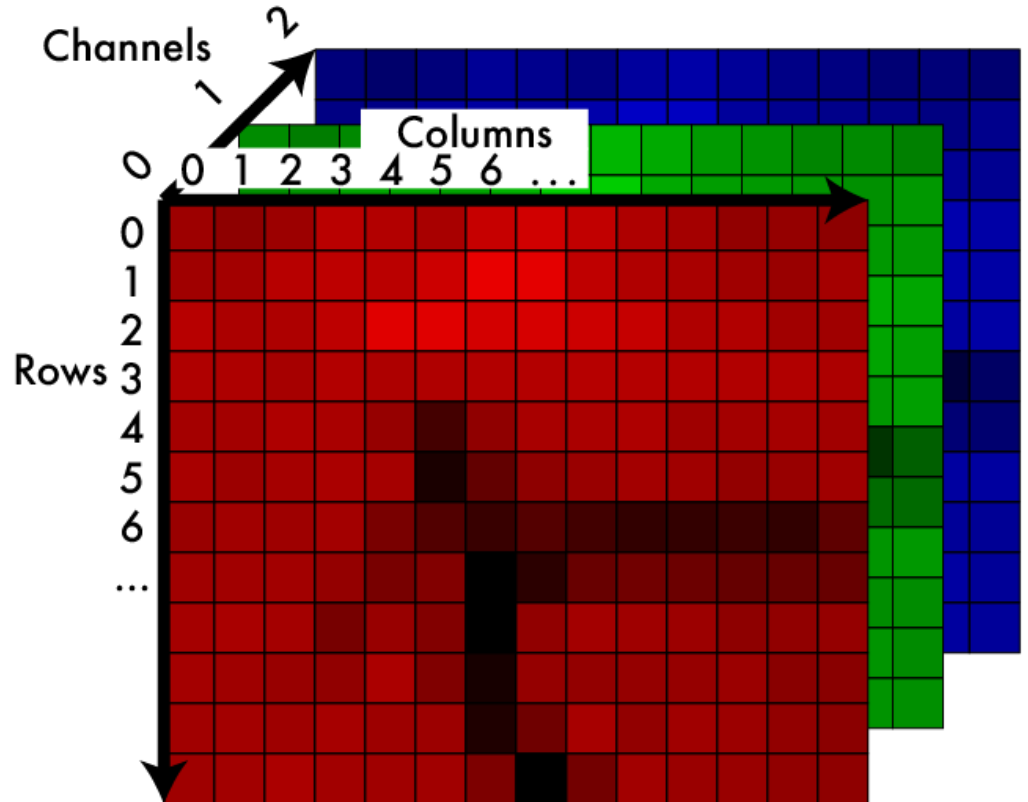
FF00AA



Time	Monday	Tuesday	Wednesday	Thursday	Friday
9:30-10:30					
10:00-11:20		CSE 455 online		CSE 455 online	
11:00-12:00			Office Hour	ECE Faculty Meeting	
12:30-2:00	Lunch	Multimedia Meeting	Lunch	Lunch	Lunch
1:30-3:30				CSE Faculty Meeting	
3:30-4:30				CSE Colloquium	
4:30-6:00				FERG Meeting	

Image: 2d array of color

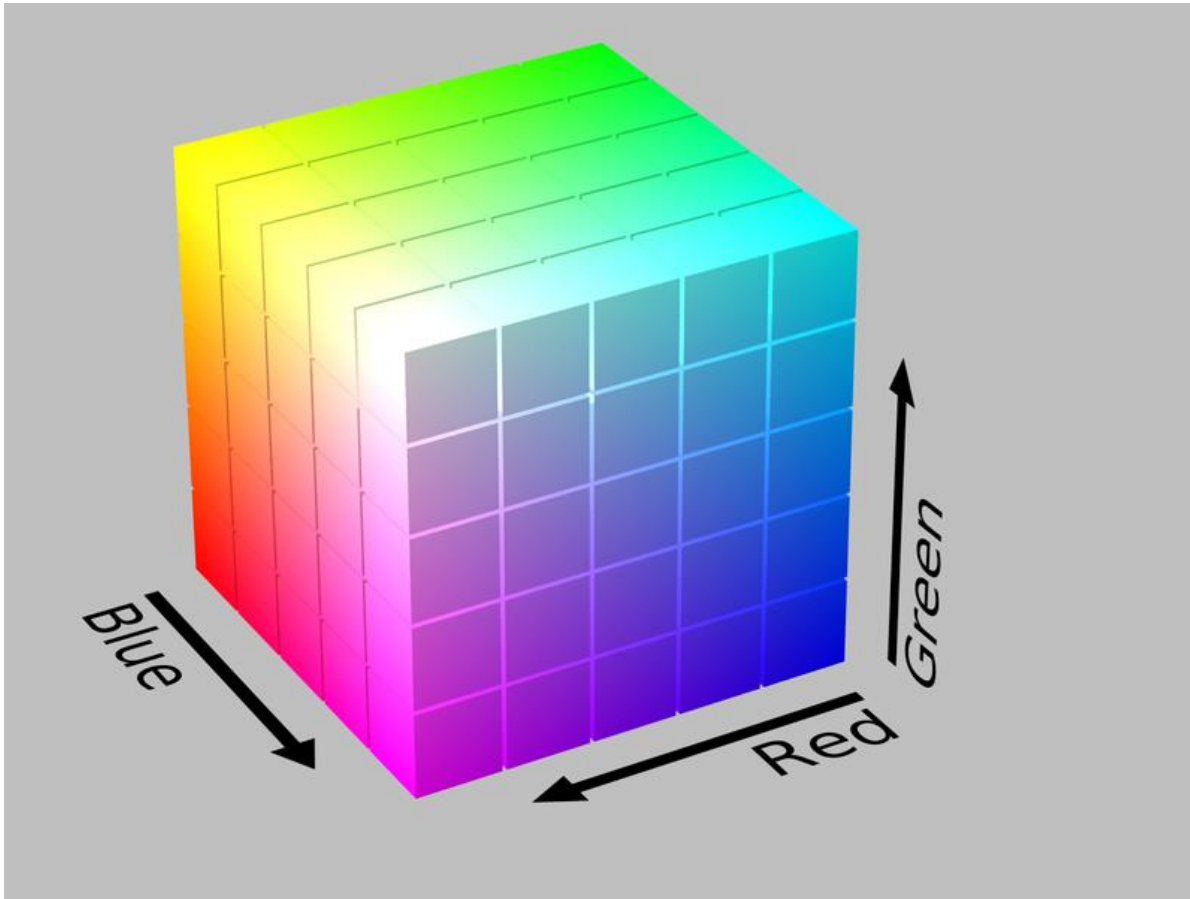
- Some range
 - [0,255]
 - 0.0 - 1.0
- We'll talk more about this later.



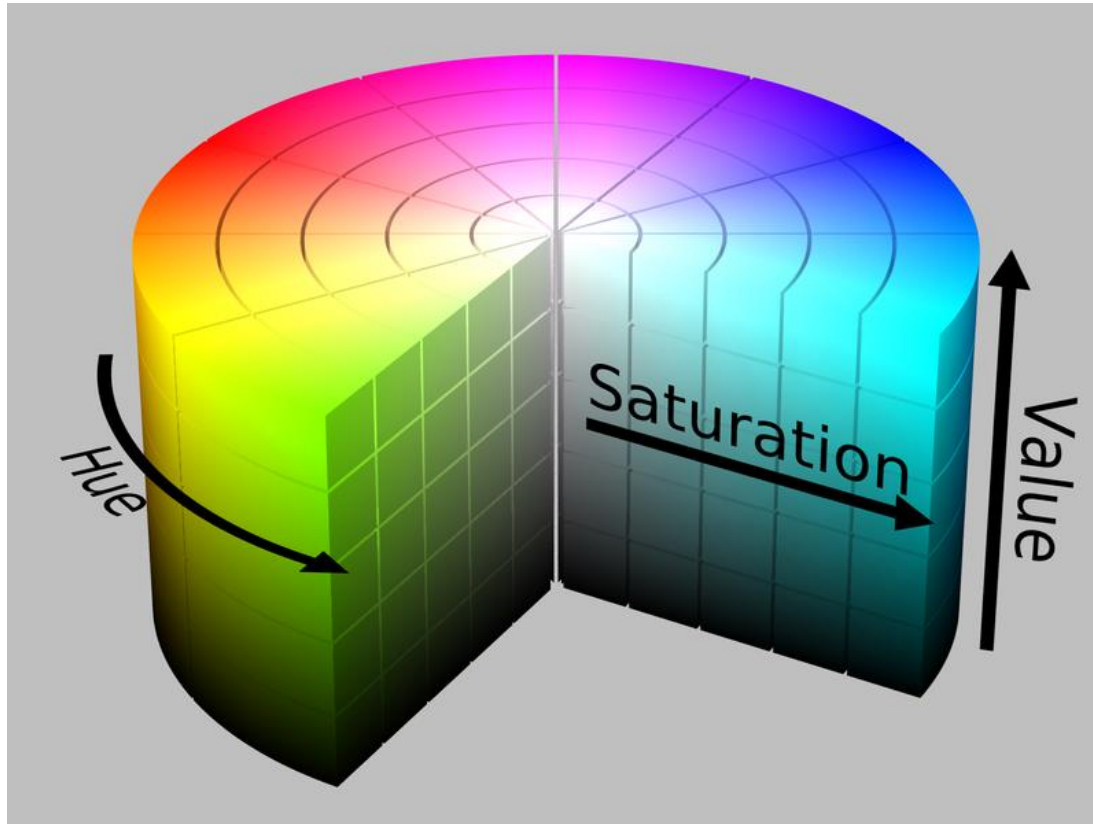
Grayscale - making color images not

- We can simulate monochromatic images from RGB
- Want a good approximation of how “bright” the image is without color information
- $(R+B+G/3)$ - looks weird
- *We should*
 - Gamma decompress
 - Calculate lightness
 - Gamma compress
- We can just operate on sRGB
 - **Typically $\sim .30R + .59G + .11B$**

RGB is a cube...

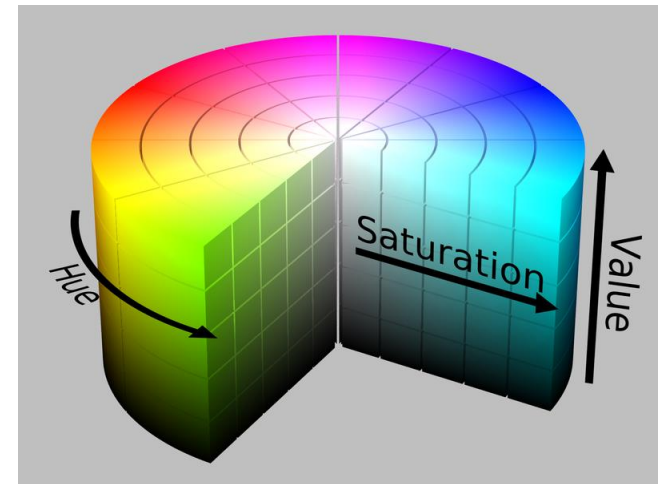


Hue, Saturation, Value: cylinder!



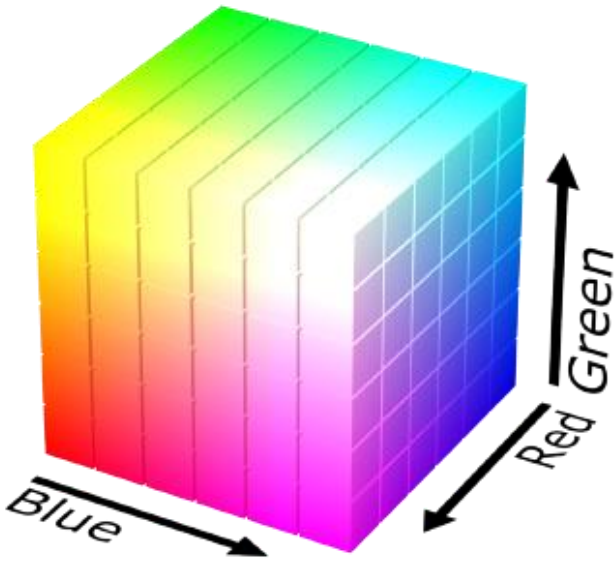
Hue, Saturation, Value

- Different model based on perception of light
- **Hue**: what color
- **Saturation**: how much color
- **Value**: how bright
- Allows easy image transforms
 - Shift the hue
 - Increase saturation

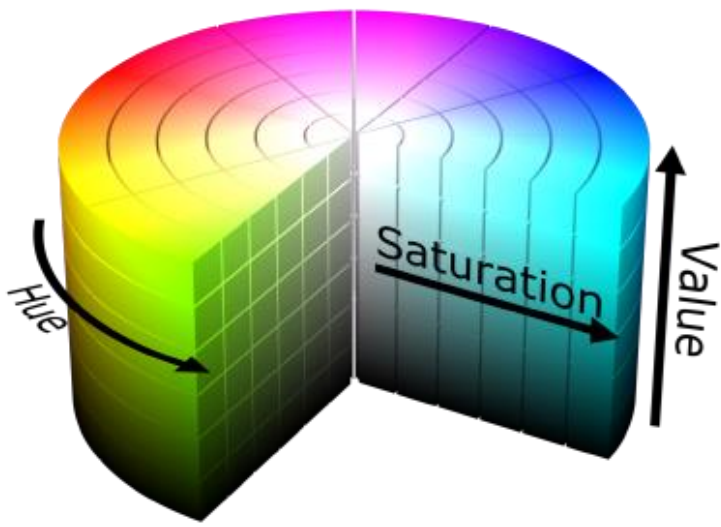


Other colorspace are fun!

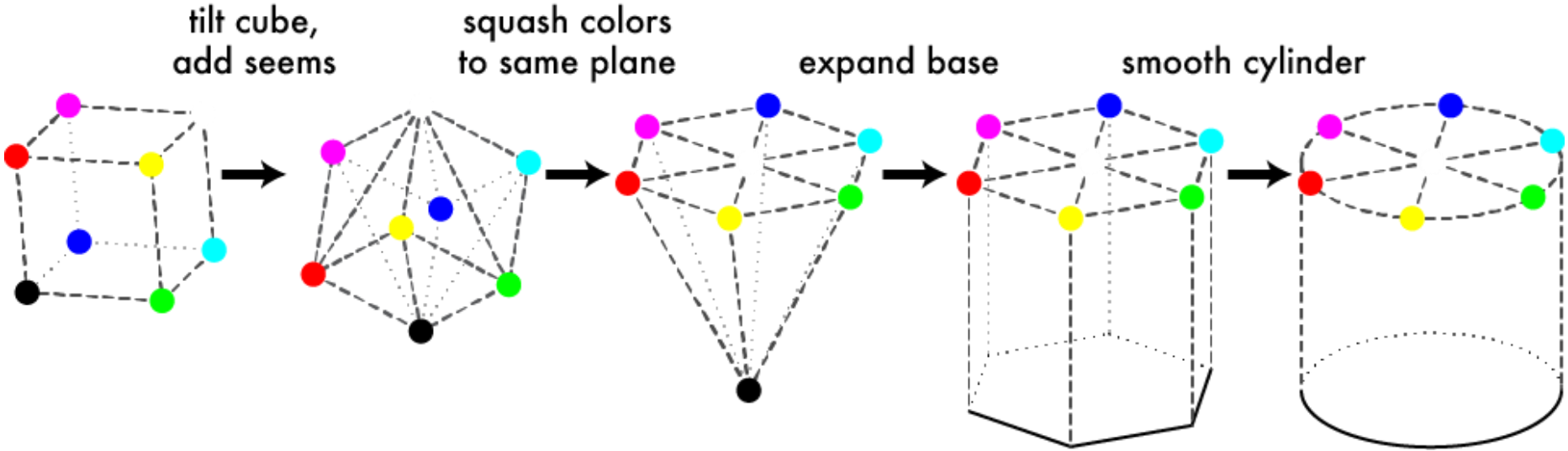
RGB



HSV



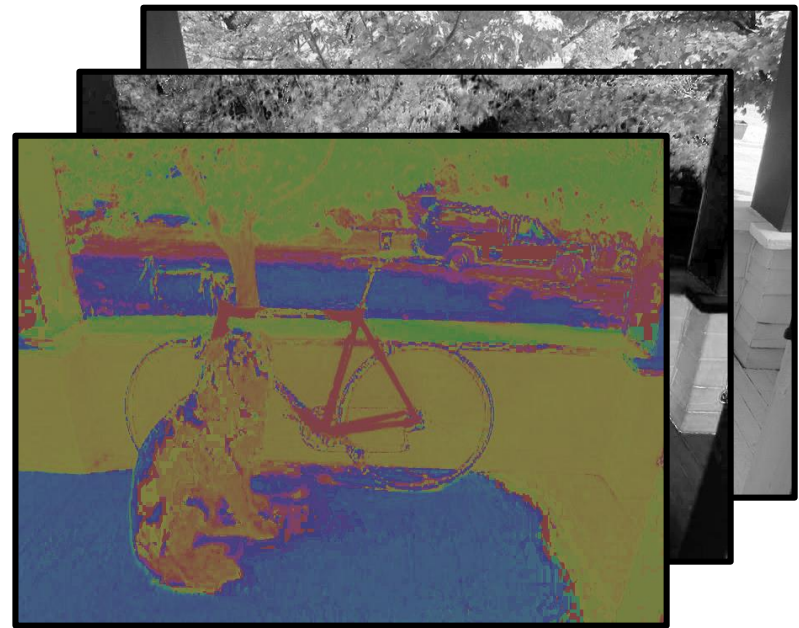
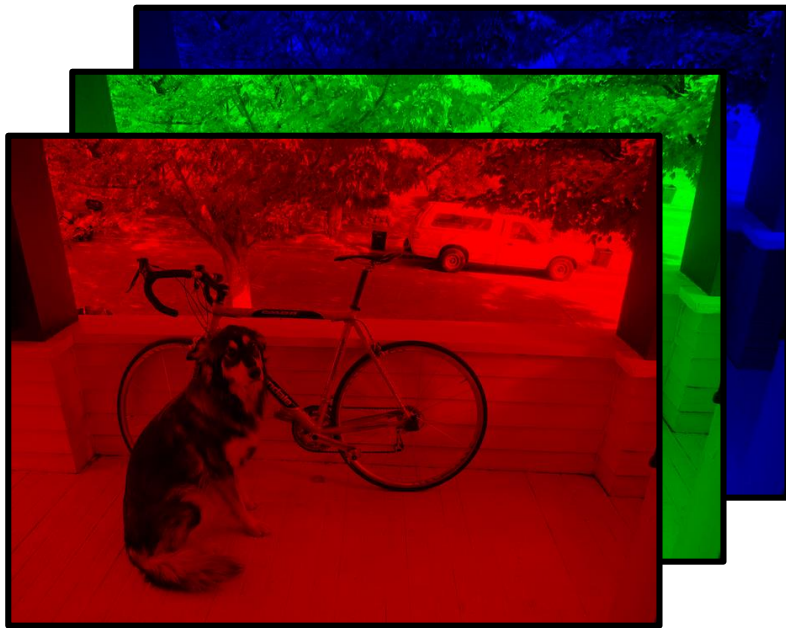
Geometric HSV to RGB:



An RGB Image



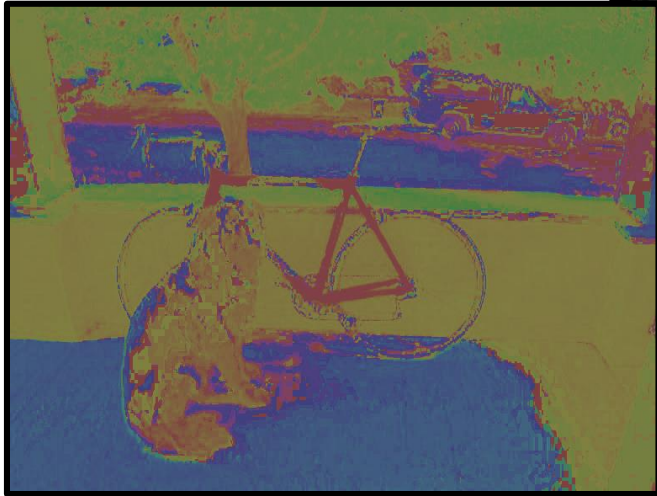
Still 3d tensor, different info



Hue

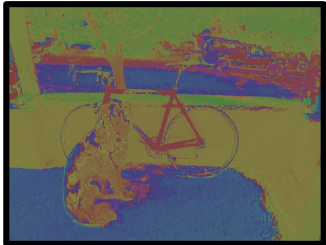
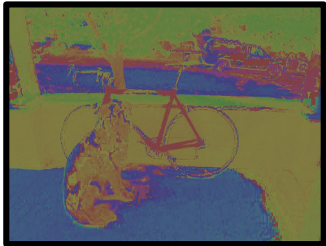
Saturation

Value



More saturation = intense colors

H Channel



2x

S Channel

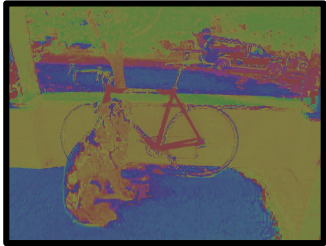


V Channel



More value = lighter image

H Channel



S Channel



V Channel

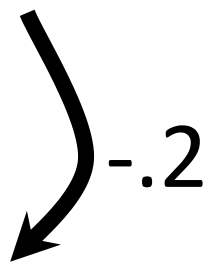
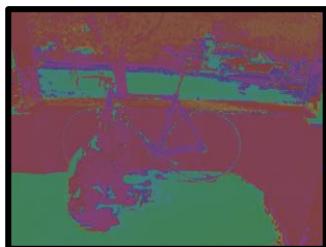


2x

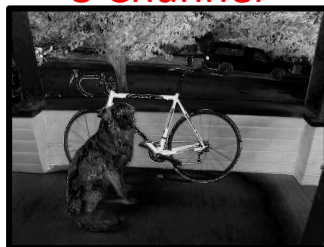


Shift hue = shift colors

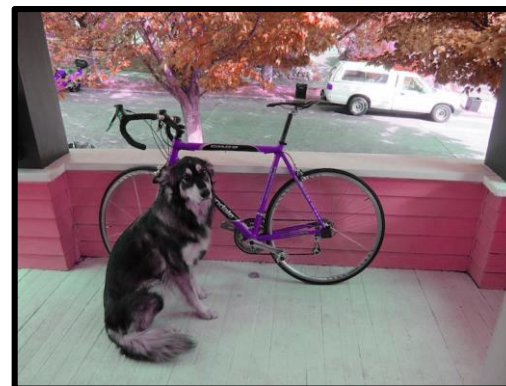
H Channel



S Channel

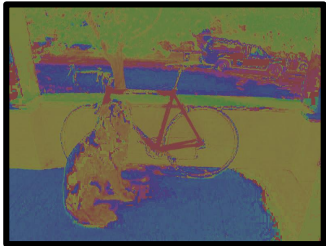


V Channel



Set hue to your favorite color!

H Channel



S Channel

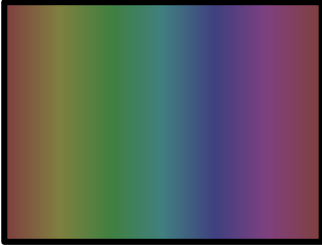
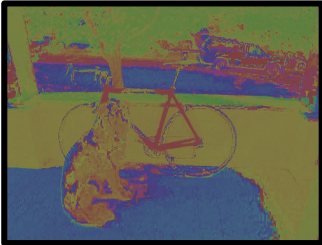


V Channel



Or pattern...

H Channel



S Channel

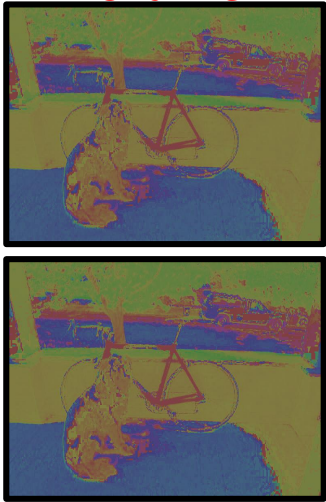


V Channel



Increase and threshold saturation

H Channel



S Channel



V Channel



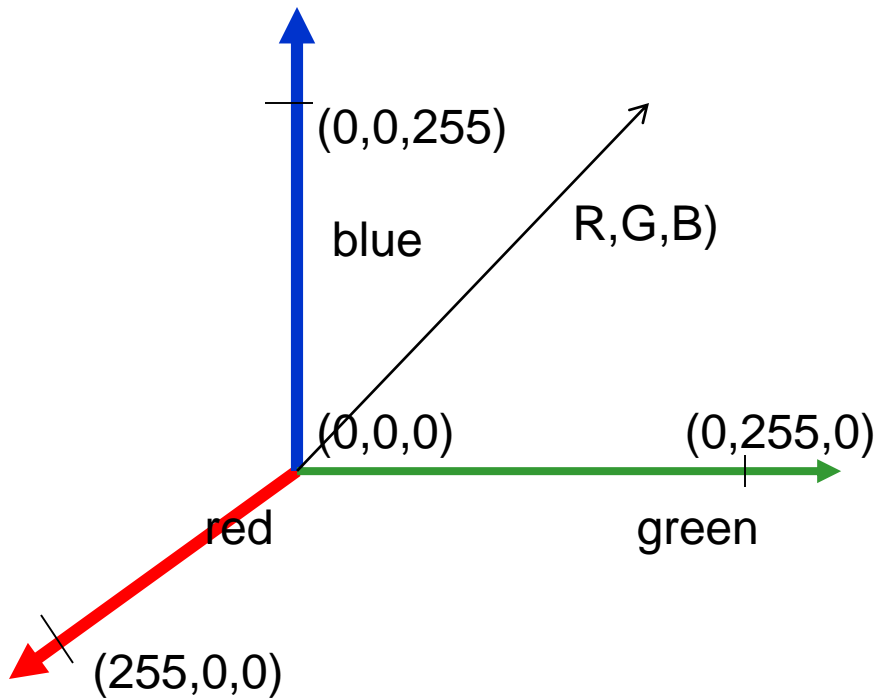


More Details on Color Spaces

- RGB standard for cameras
- HSI/HSV allows us to separate
- CIE L^*a^*b intensity plus 2 color channels
- YIQ color TVs, Y is intensity
- Opponent used in Swain & Ballard work

RGB Color Space

Absolute



Normalized

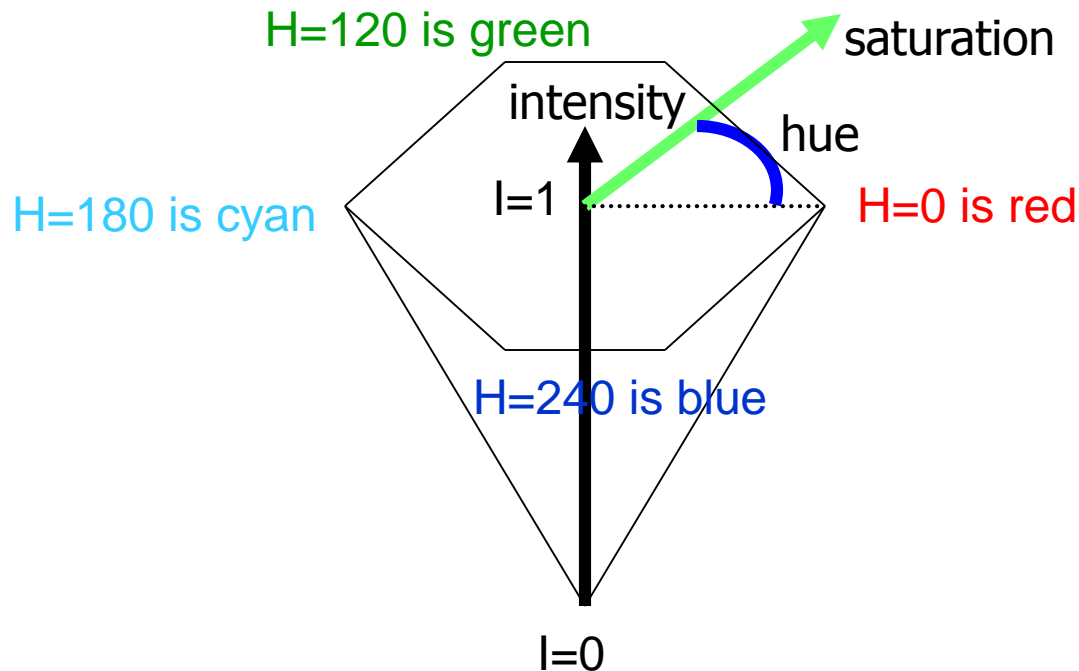
Normalized red $r = R/(R+G+B)$

Normalized green $g = G/(R+G+B)$

Normalized blue $b = B/(R+G+B)$

Color hexagon for HSI (HSV)

- Hue is encoded as an angle (0 to 2π).
- Saturation is the distance to the vertical axis (0 to 1).
- Intensity is the height along the vertical axis (0 to 1).



Conversion from RGB to YIQ

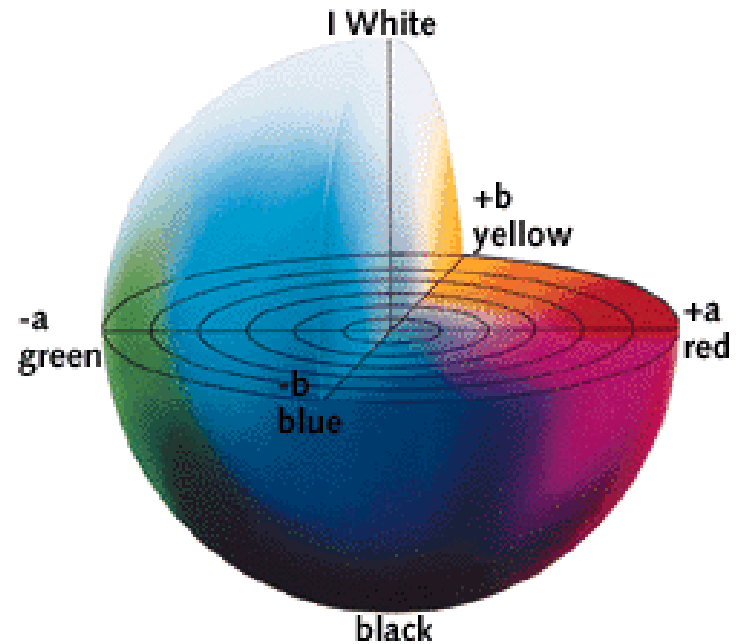
An approximate linear transformation from RGB to YIQ:

$$\begin{aligned} \text{luminance } Y &= 0.30R + 0.59G + 0.11B \\ \text{R - cyan } I &= 0.60R - 0.28G - 0.32B \\ \text{magenta - green } Q &= 0.21R - 0.52G + 0.31B \end{aligned}$$

We often use this for **color to gray-tone conversion**.

CIELAB, Lab, L^*a^*b

- One luminance channel (L) and two color channels (a and b).
- In this model, the color differences which you perceive correspond to Euclidian distances in CIELab.
- The a axis extends from green (-a) to red (+a) and the b axis from blue (-b) to yellow (+b). The brightness (L) increases from the bottom to the top of the three-dimensional model.



CIE $L^*a^*b^*$ (CIELAB) is a [color space](#) specified by the [International Commission on Illumination](#) (French *Commission internationale de l'éclairage*, hence its CIE [initialism](#)).

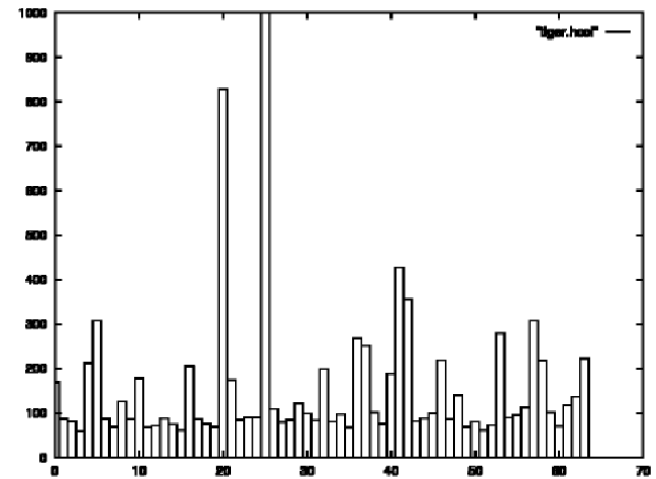
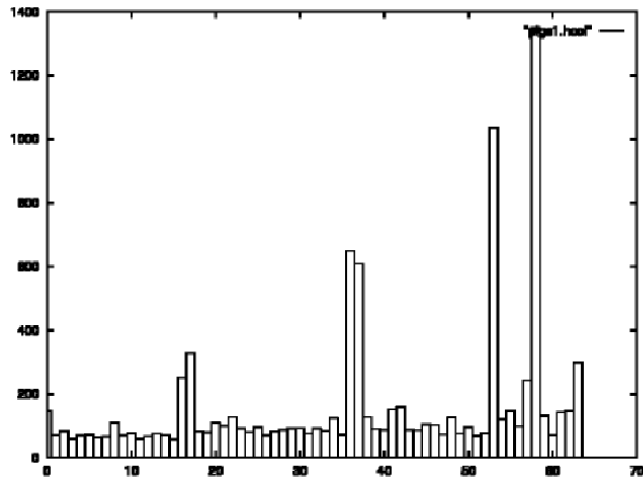
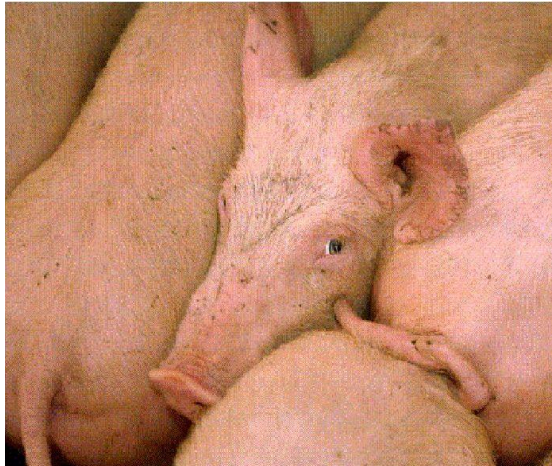
Histograms

- A histogram of a gray-tone image is an array $H[*]$ of bins, one for each gray tone.
- $H[i]$ gives the count of how many pixels of an image have gray tone i .
- $P[i]$ (the normalized histogram) gives the percentage of pixels that have gray tone i .

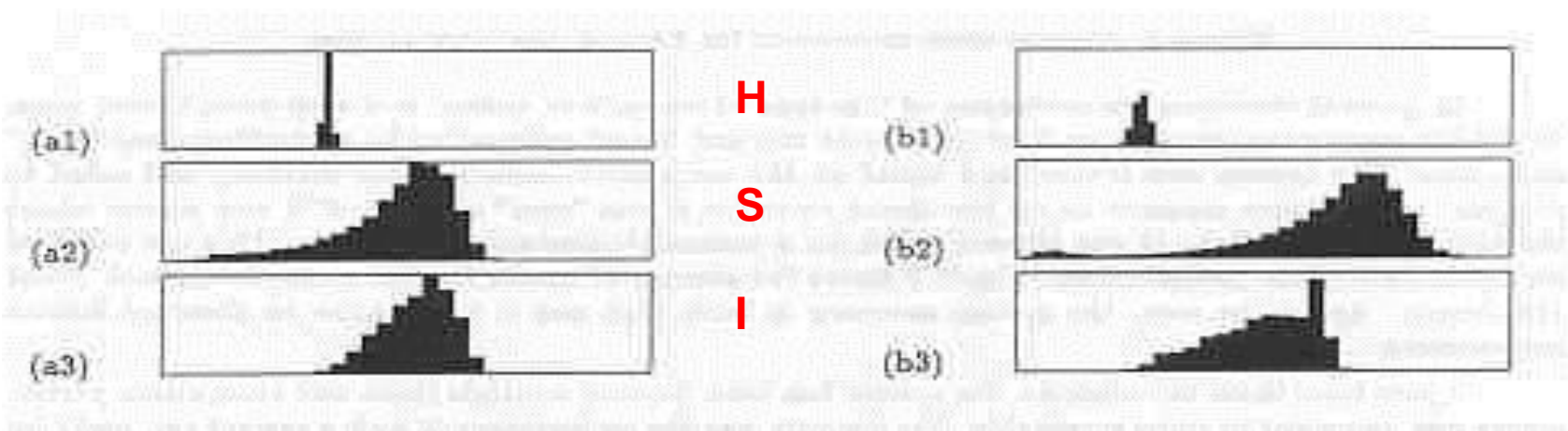
Color histograms can represent an image

- Histogram is fast and easy to compute.
- Size can easily be normalized so that different image histograms can be compared.
- Can match color histograms for database query or classification.

Histograms of two color images



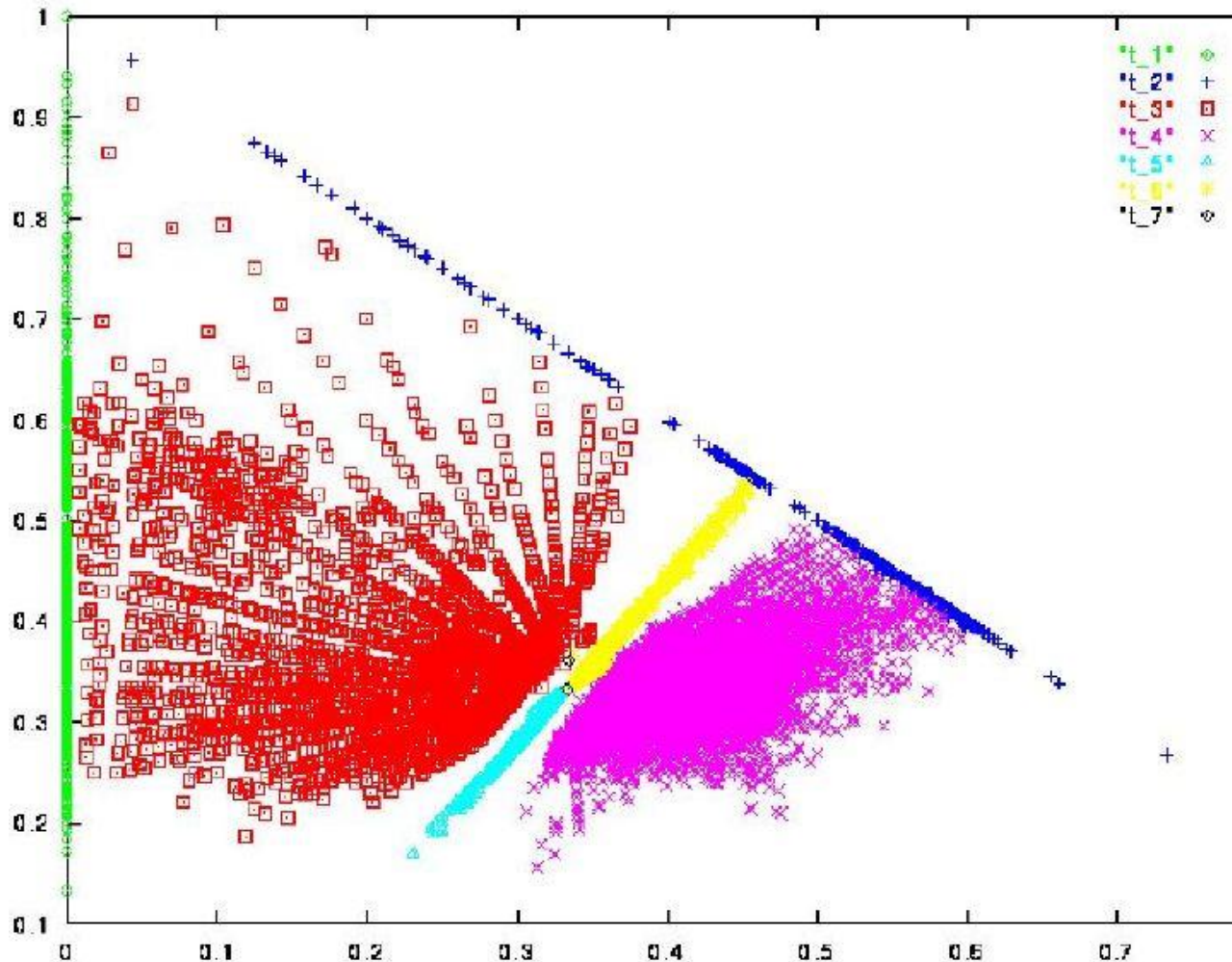
Apples versus Oranges



Separate HSI histograms for apples (left) and oranges (right) used by IBM's VeggieVision for recognizing produce at the grocery store checkout station (see Ch 16).

Skin color in Normalized R-G Space

G normalized



Purple region shows skin color samples from several people. Blue and yellow regions show skin in shadow or behind a beard.

R normalized

Finding a face in video frame



- (left) input video frame
- (center) pixels classified according to RGB space
- (right) largest connected component with aspect similar to a face (all work contributed by Vera Bakic)

Swain and Ballard's Histogram Matching for Color Object Recognition (IJCV Vol 7, No. 1, 1991)

Opponent Encoding:

- $wb = R + G + B$
- $rg = R - G$
- $by = 2B - R - G$

Histograms: $8 \times 16 \times 16 = 2048$ bins

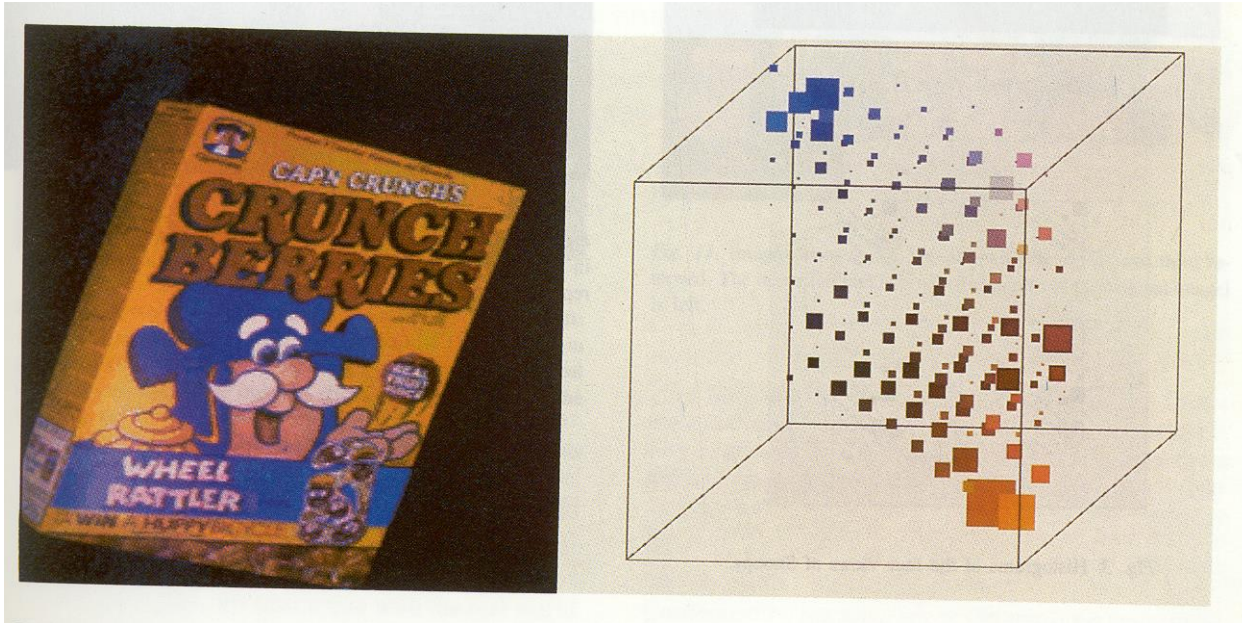
Intersection of image histogram and model histogram:

$$\text{intersection}(h(I), h(M)) = \sum_{j=1}^{\text{numbins}} \min\{h(I)[j], h(M)[j]\}$$

Match score is the **normalized** intersection:

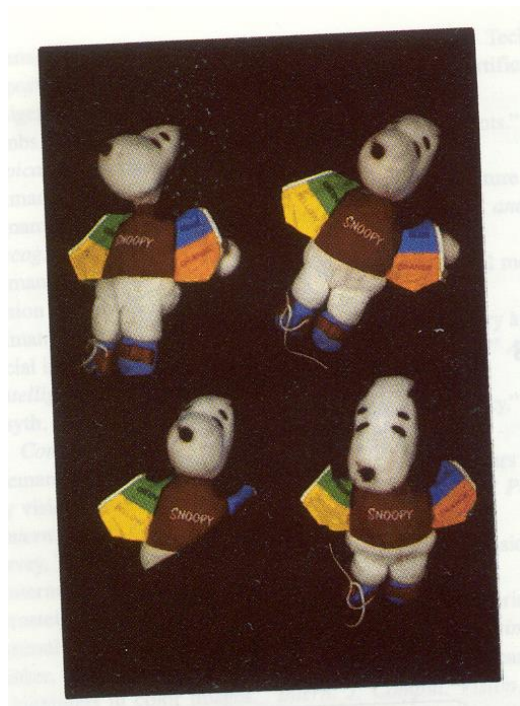
$$\text{match}(h(I), h(M)) = \text{intersection}(h(I), h(M)) / \sum_{j=1}^{\text{numbins}} h(M)[j]$$

(from Swain and Ballard)

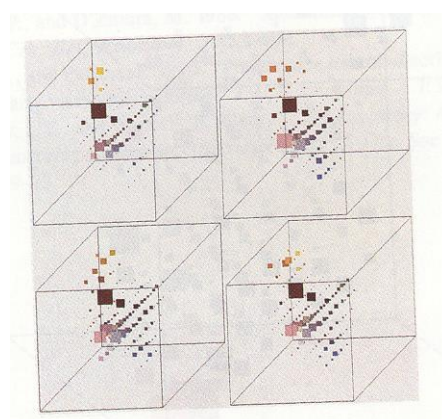


cereal box image

3D color histogram



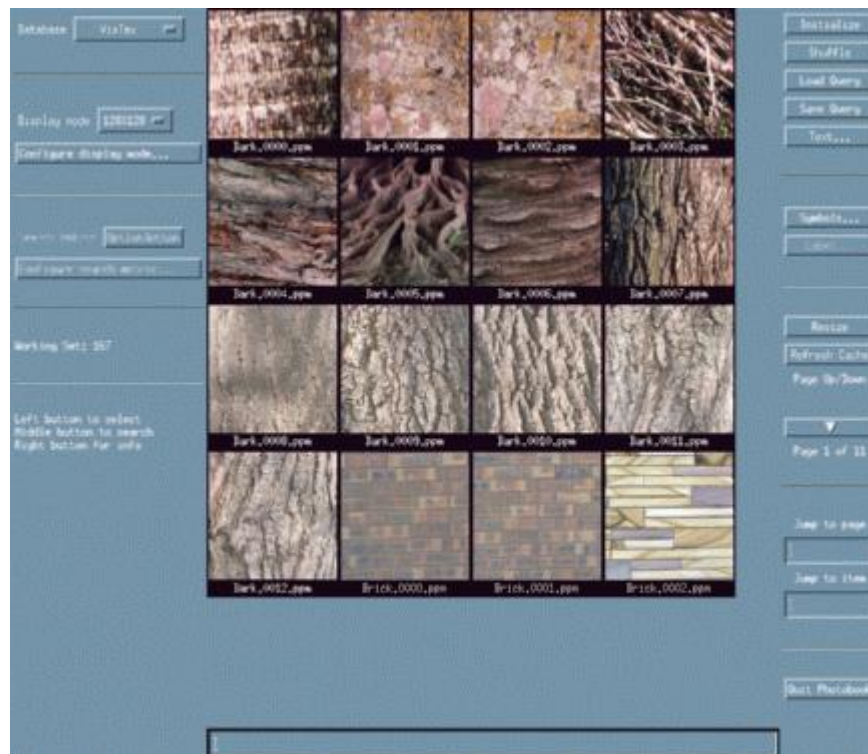
Four views of Snoopy



Histograms

Texture

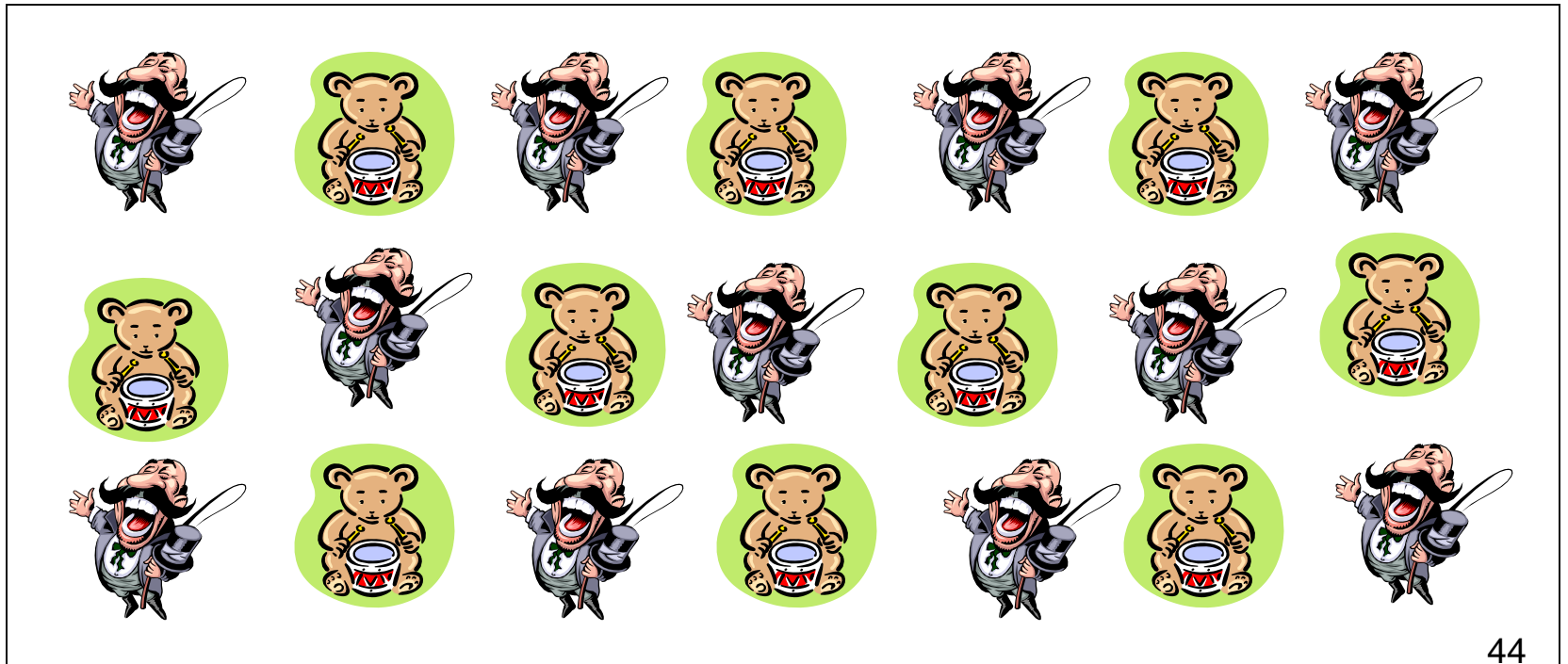
- Color is well defined.
- But what *is* texture?



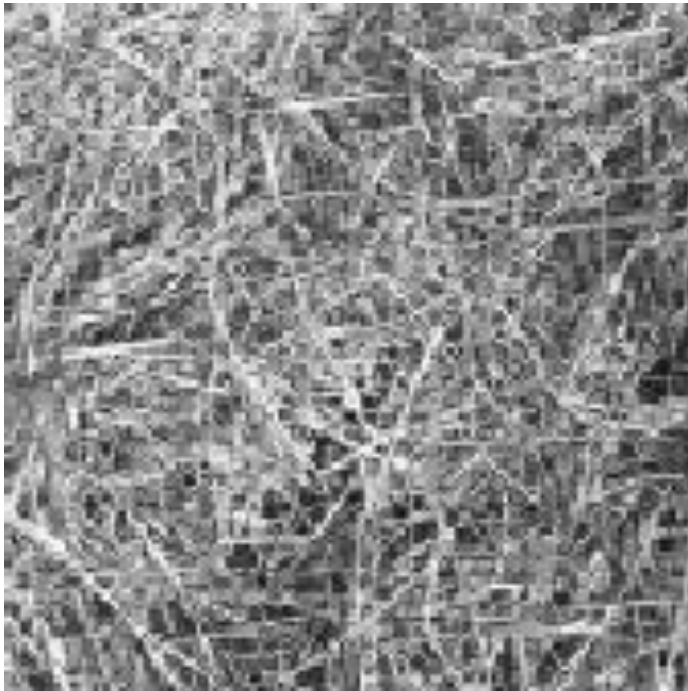
Structural Texture

Texture is a description of the spatial arrangement of color or intensities in an image or a selected region of an image.

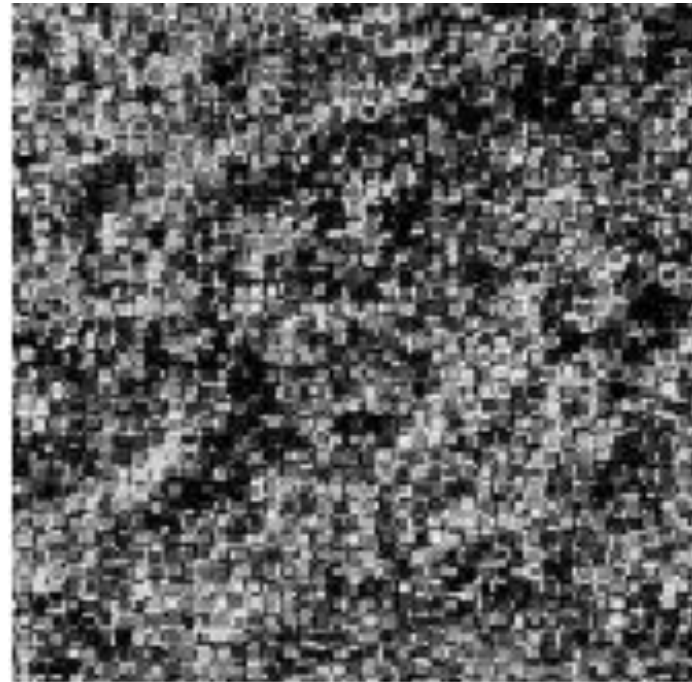
Structural approach: a set of texels in some regular or repeated pattern



Natural Textures from VisTex



grass



leaves

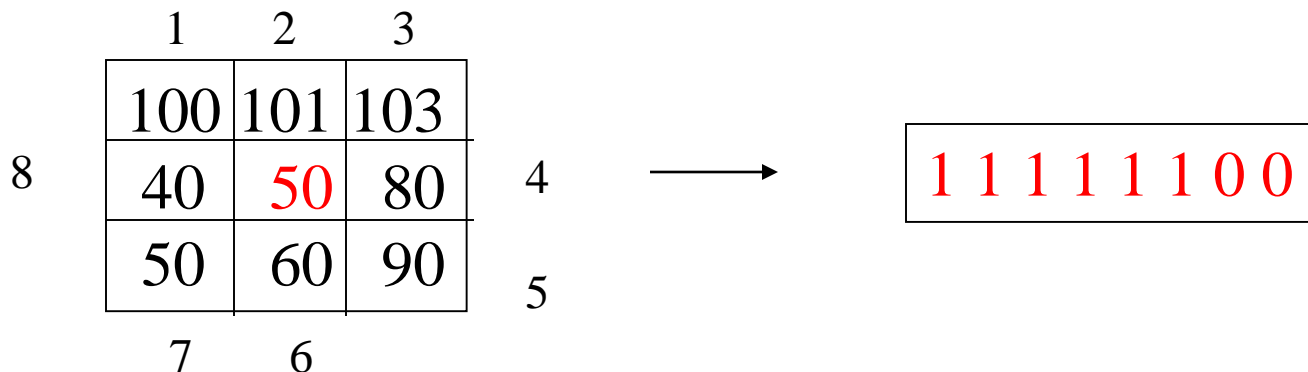
What/Where are the texels?

The Case for Statistical Texture

- Segmenting out texels is difficult or impossible in real images.
- Numeric quantities or statistics that describe a texture can be computed from the gray tones (or colors) alone.
- This approach is less intuitive, but is computationally efficient.
- It can be used for both classification and segmentation.

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.



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

Fids demo



The screenshot shows the Fids demo interface. At the top, a query image of a garden path is highlighted with a red box. Below it, six similar images are displayed in a 2x3 grid. The interface includes a control bar with buttons for 'Random', 'Go', 'ZoomIn', and 'ZoomOut'. To the right of these buttons, it says 'Found 191 matches. Displaying 1 - 6'. Below the control bar, there are two columns of distance measures. The first column has checkboxes for 'ColorHistL14x4x4', 'ColorHist8x8x8', 'SobelEdgeHist', 'LBPHist', 'fleshiness', and 'Wavelets'. The second column has radio buttons for 'loose' and 'strict', and a numerical value '5' next to each. To the right of these columns is a search operator selection box with radio buttons for 'And', 'Or', and 'Sum'. At the bottom, it says 'Server Connected'.

Found 191 matches. Displaying 1 - 6

distance measures	loose ... strict	
<input type="checkbox"/> ColorHistL14x4x4	<input type="radio"/> loose <input type="radio"/> strict	5
<input type="checkbox"/> ColorHist8x8x8	<input type="radio"/> loose <input type="radio"/> strict	5
<input type="checkbox"/> SobelEdgeHist	<input type="radio"/> loose <input type="radio"/> strict	5
<input checked="" type="checkbox"/> LBPHist	<input type="radio"/> loose <input type="radio"/> strict	5
<input type="checkbox"/> fleshiness	<input type="radio"/> loose <input type="radio"/> strict	5
<input type="checkbox"/> Wavelets	<input type="radio"/> loose <input type="radio"/> strict	5

Search operators: And, Or, Sum

Server Connected

Example

Fids demo

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

The screenshot shows the Fids demo interface. At the top, there's a grid of six images. The first image in the top row is highlighted with a red border. Below the grid are navigation buttons: Random, Go, ZoomIn, and a right arrow. To the right of these buttons, it says "Found 119 matches. Displaying 1 - 6". Below the navigation buttons are two columns of controls. The first column is labeled "distance measures" and includes checkboxes for ColorHistL14x4x4, ColorHist8x8x8, SobelEdgeHist, LBPHist (checked), fleshiness, and Wavelets. The second column is labeled "loose ... strict" and has a slider for each measure, with a "5" at the end of each. To the right of these sliders are three radio buttons: And (selected), Or, and Sum. Below these controls are two buttons: "Put In Cart" and "Check Out". At the bottom right, there's a box with the text "A double click on an image means:" and two radio buttons: "Set query / Go" (selected) and "Zoom in". At the bottom left, it says "Server Connected".

What else is LBP good for?

- We found it in a paper for classifying deciduous trees.
- We used it in a real system for finding cancer in Pap smears.
- We are using it to look for regions of interest in breast and melanoma biopsy slides.

Co-occurrence Matrix Features

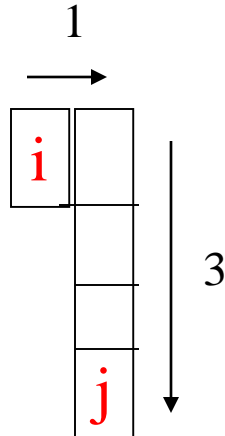
A co-occurrence matrix is a 2D array C in which

- Both the rows and columns represent a set of possible image values.
- $C_d(i,j)$ indicates how many times value i co-occurs with value j in a particular spatial relationship d .
- The spatial relationship is specified by a vector $d = (dr,dc)$.

Co-occurrence Example

1	1	0	0
1	1	0	0
0	0	2	2
0	0	2	2
0	0	2	2
0	0	2	2

gray-tone
image



$$d = (3,1)$$

	0	1	2
0	1	0	3
1	2	0	2
2	0	0	1

co-occurrence
matrix

C_d

From C_d we can compute N_d , the normalized co-occurrence matrix, where each value is divided by the sum of all the values.

Co-occurrence Features

What do these measure?

$$\text{Energy} = \sum_i \sum_j N_d^2(i, j) \quad (7.7)$$

$$\text{Entropy} = - \sum_i \sum_j N_d(i, j) \log_2 N_d(i, j) \quad (7.8)$$

$$\text{Contrast} = \sum_i \sum_j (i - j)^2 N_d(i, j) \quad (7.9)$$

$$\text{Homogeneity} = \sum_i \sum_j \frac{N_d(i, j)}{1 + |i - j|} \quad (7.10)$$

$$\text{Correlation} = \frac{\sum_i \sum_j (i - \mu_i)(j - \mu_j) N_d(i, j)}{\sigma_i \sigma_j} \quad (7.11)$$

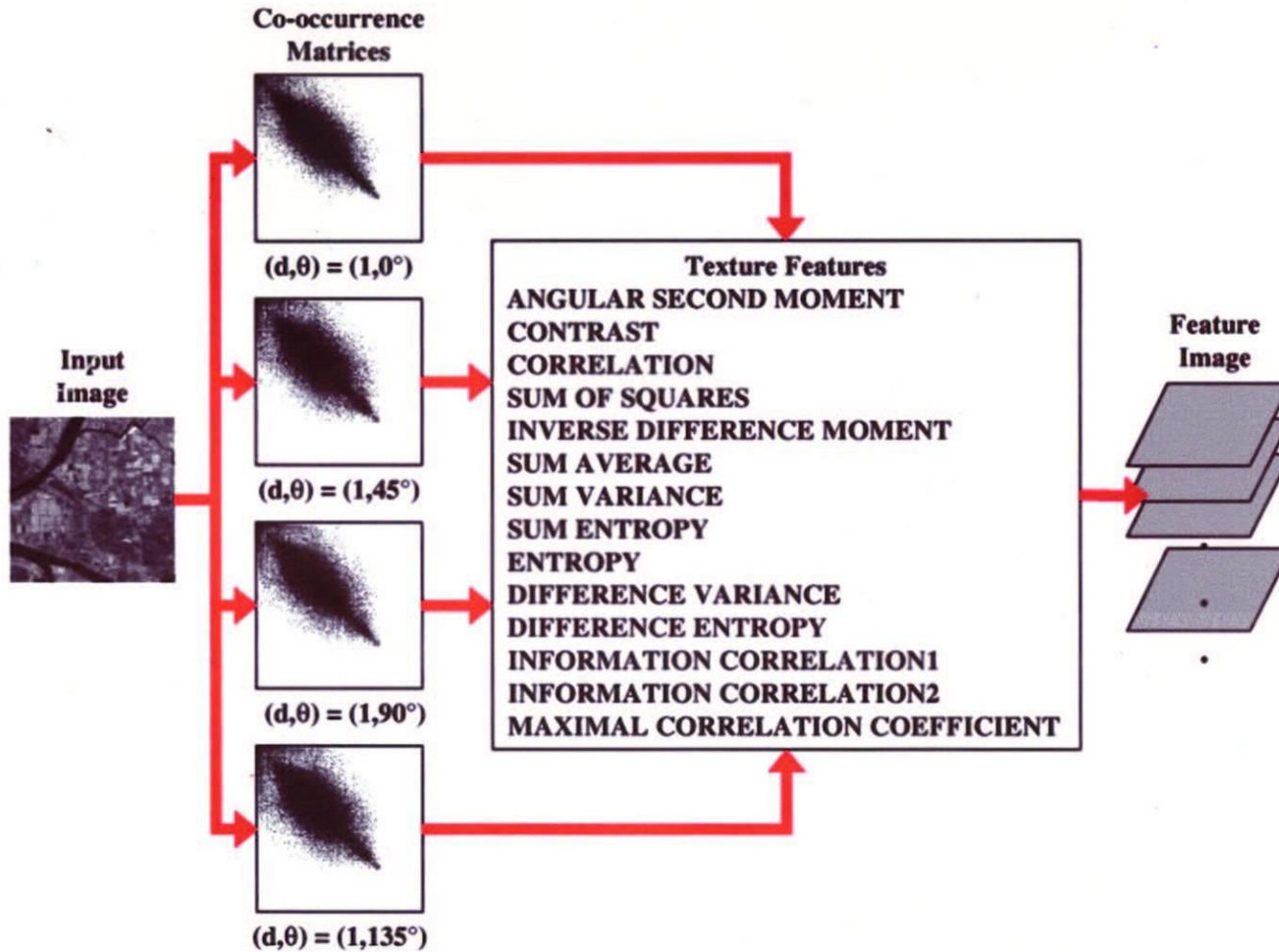
where μ_i, μ_j are the means and σ_i, σ_j are the standard deviations of the row and column sums.

Energy measures uniformity of the normalized matrix.

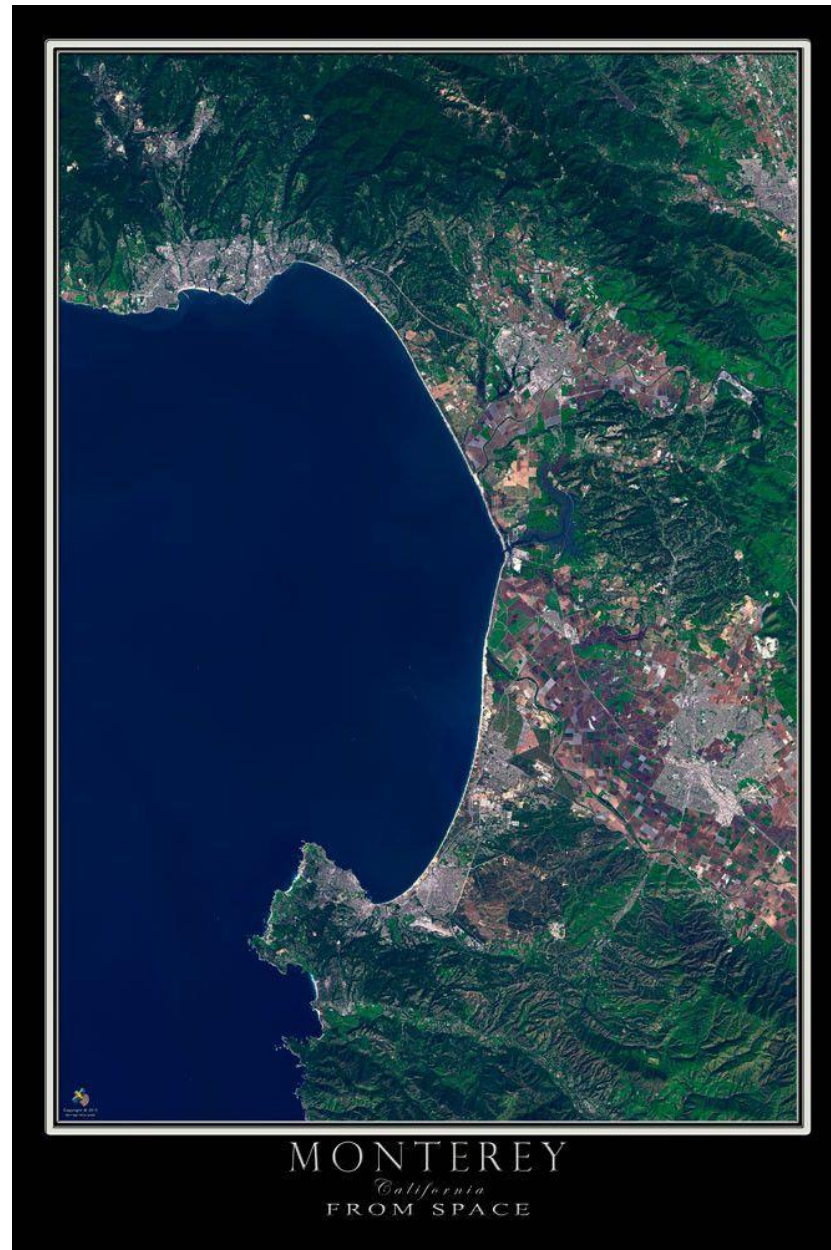
What are Co-occurrence Features used for?

- They were designed for recognizing different kinds of **land uses in satellite images**.
- They are still used heavily in geospatial domains, but they can be added on to any other calculated features.

Example



Satellite Image of Monterey Bay



Assignment 0

FUN WITH COLOR!

Image basics

- Data structure for an image

```
typedef struct{  
    int h, w, c;  
    float *data;  
} image;
```

Image basics

- Data structure for an image

```
typedef struct{  
    int h, w, c;  
    float *data;  
} image;
```

– h = height

– w = width

– c = number of channels

- c = 3 for RGB image; c = 1 for grayscale image

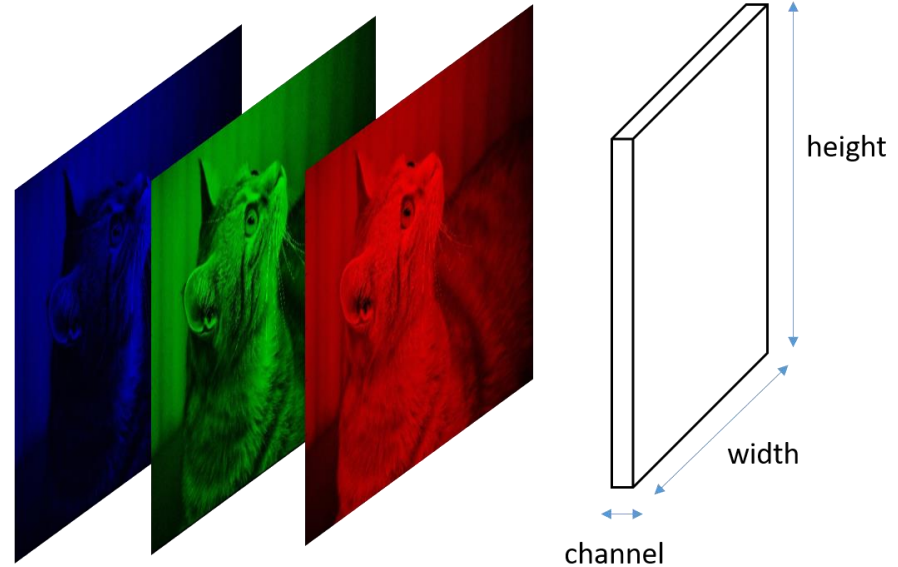


Image basics

- Data structure for an image

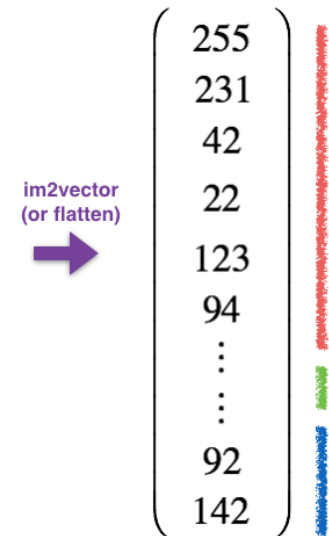
```
typedef struct{  
    int h, w, c;  
    float *data;  
} image;
```

- data = array of floats
- floats in the range [0,1]
- 3D image matrix linearized into 1D array

3-channel matrix

	Blue				
Green	255	134	93	22	
Red	255	134	202	22	
255	231	42	22	4	30
123	94	83	2	92	124
34	44	187	92	14	142
34	76	232	124	14	
67	83	194	202		

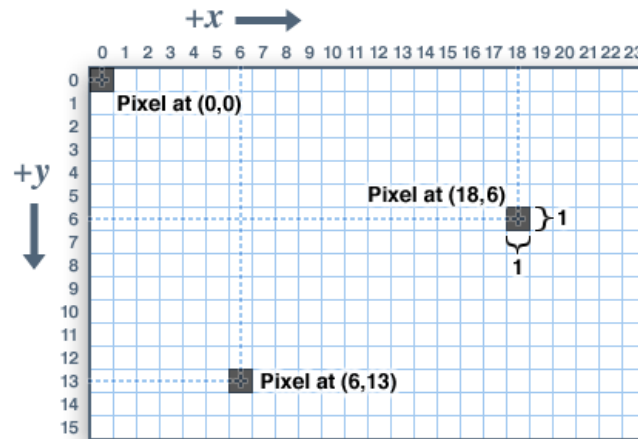
reshaped image vector



To Do #1

- Fill in:

- `float get_pixel(image im, int x, int y, int c)`



- `void set_pixel(image im, int x, int y, int c, float v)`

- assign value `v` to `im.data[coord(x,y,c)]`

To Do #2

- Fill in:
 - `image copy_image(image im)`
 - create a separate image that is a copy of the input image `im` and return the copy
 - Useful functions:
 - `make_image()` in `load_image.c`
 - `memcpy()`
 - You can use `get_pixel()` and `set_pixel()` functions you implemented from now on

To Do #3

- Fill in:
 - `image rgb_to_grayscale(image im)`
 - create a new image with 1 channel
 - Get R,G,B values from input image `im`
 - Compute $Y = 0.299 R + 0.587 G + .114 B$
 - Assign Y to new image and return that image

To Do #4

- Fill in:
 - `void shift_image(image im, int c, float v)`
 - add value `v` to each pixel of `im` in channel `c`
 - change the input image in-place (do not create a separate image) wherever the return type of the function is `void`

To Do #5

- Fill in:
 - `void clamp_image(image im)`
 - clamp the pixel values of input image `im` to be in the range `[0,1]`

To Do #6

- Fill in:
 - `void rgb_to_hsv(image im)`
 - Convert R,G,B values of image `im` pixelwise into H,S,V values using the given formulas
 - You can use the `three_way_max()` and `three_way_min()` functions provided

To Do #7

- Fill in:
 - `void hsv_to_rgb(image im)`
 - Convert H,S,V values of image `im` pixelwise into R,G,B values using the given table

To Do #8 (extra credit)

- Fill in:
 - `void scale_image(image im, int c, float v)`
 - multiply each pixel of `im` in channel `c` with value `v`
 - add to the lines in `image.h` and other files as necessary to test the function. No need to submit the other edited files; write a comment in CAPS at the top of `process_image.c` if you attempt any extra credit

To Do #9 (super extra credit)

- Fill in:
 - `void rgb_to_lch(image im)`
 - Convert R,G,B values of image `im` pixelwise into L,C,H values using the formulas in the given link

Have Fun