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 commission 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
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

<table>
<thead>
<tr>
<th>Time</th>
<th>Monday</th>
<th>Tuesday</th>
<th>Wednesday</th>
<th>Thursday</th>
<th>Friday</th>
</tr>
</thead>
<tbody>
<tr>
<td>9:30-10:30</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10:00-11:20</td>
<td>CSE 455 online</td>
<td></td>
<td></td>
<td>CSE 455 online</td>
<td></td>
</tr>
<tr>
<td>11:00-12:00</td>
<td>Office Hour</td>
<td>Multimedia Meeting</td>
<td>Lunch</td>
<td>BCE Faculty Meeting</td>
<td>Lunch</td>
</tr>
<tr>
<td>12:30-2:00</td>
<td>Lunch</td>
<td></td>
<td>Lunch</td>
<td>Lunch</td>
<td></td>
</tr>
<tr>
<td>1:30-3:30</td>
<td>CSE Faculty Meeting</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3:30-4:30</td>
<td>CSE Colloquium</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>4:30-6:00</td>
<td>FERG Meeting</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
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
  - Calculatight lightness
  - Gamma compress
- We can just operate on sRGB
  - Typically \(~ .30R + .59G + .11B\)
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!
Geometric HSV to RGB:

tilt cube, add seems
squash colors to same plane
expand base
smooth cylinder
An RGB Image
Still 3d tensor, different info
<table>
<thead>
<tr>
<th>Hue</th>
<th>Saturation</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Image" /></td>
<td><img src="image2.png" alt="Image" /></td>
<td><img src="image3.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image4.png" alt="Image" /></td>
<td><img src="image5.png" alt="Image" /></td>
<td><img src="image6.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="image7.png" alt="Image" /></td>
<td><img src="image8.png" alt="Image" /></td>
<td><img src="image9.png" alt="Image" /></td>
</tr>
</tbody>
</table>
More saturation = intense colors

H Channel

S Channel

V Channel

2x
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

Red bicycle and dog images in different color channels.
Or pattern...

H Channel

S Channel

V Channel

[Images of patterns and channels]
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 intensity plus 2 color channels
- **CIE L*a*b** color TVs, Y is intensity
- **YIQ** used in Swain & Ballard work
- **Opponent**
RGB Color Space

Absolute

Normalized

Normalized red \( r = \frac{R}{R+G+B} \)

Normalized green \( g = \frac{G}{R+G+B} \)

Normalized blue \( b = \frac{B}{R+G+B} \)
Color hexagon for HSI (HSV)

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

H=120 is green
H=180 is cyan
H=240 is blue
H=0 is red
Conversion from RGB to YIQ

An approximate linear transformation from RGB to YIQ:

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

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

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

\[
\text{match}(h(I), h(M)) = \frac{\text{intersection}(h(I), h(M))}{\sum_{j=1}^{\text{numbins}} h(M)[j]}
\]
(from Swain and Ballard)

cereal box image  3D color histogram
Four views of Snoopy

Histograms
Texture

- Color is well defined.
- But what is texture?
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.
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.
Low-level measures don’t always find semantically similar images.
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

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?

\[
\text{Energy} = \sum_{i} \sum_{j} N_d(i, j)^2
\]
(7.7)

\[
\text{Entropy} = -\sum_{i} \sum_{j} N_d(i, j) \log_2 N_d(i, j)
\]
(7.8)

\[
\text{Contrast} = \sum_{i} \sum_{j} (i - j)^2 N_d(i, j)
\]
(7.9)

\[
\text{Homogeneity} = \sum_{i} \sum_{j} \frac{N_d(i, j)}{1 + |i - j|}
\]
(7.10)

\[
\text{Correlation} = \frac{\sum_{i} \sum_{j} (i - \mu_i)(j - \mu_j) N_d(i, j)}{\sigma_i \sigma_j}
\]
(7.11)

where \(\mu_i, \mu_j\) are the means and \(\sigma_i, \sigma_j\) 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
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 $Y = 0.299 \, R + 0.587 \, G + 0.114 \, B$
    • 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