The small green squares are the same color. Hold your hand to the screen and block out the adjacent blocks, showing only the two green squares, and you'll see how dramatic the chromatic induction effect is.

These two brown shades are different colors—yet the square blocks appear to be the same shade of brown. It is only when your eye follows the line down that you see how different the shades are. Cover the bottom part of the illustration with your hand and look only at the brown boxes to see the way the chromatic induction effect
Image Processing 101
Pset 0 is out and due Wednesday
Domain, range
Domain vs. range
Domain vs. range
Domain vs. range

domain : 2D plane $x, y$

range : RGB [0 - 1] float
Domain vs. range

- 2D plane: domain of images
  - x, y Cartesian coordinates
- color value: range (R³ for us)
  - stores the red, green and bleu components in \( \text{im}(y, x, 0) \), \( \text{im}(y, x, 1) \), \( \text{im}(y, x, 2) \), respectively
Basic types of operations

Point operations: range only
Basic types of operations

Point operations: range only

Each pixel is treated the same, doesn’t depend on other pixels, only on the input color
Basic types of operations

Point operations:
range only

Domain operations
Basic types of operations

Point operations:
range only

Domain operations

Only moves pixels around, doesn’t change colors, just where they are
Basic types of operations

Point operations: range only

Domain operations

Neighborhood operations: domain and range
Basic types of operations

Point operations: range only
- Pset 1

Domain operations
- Pset 2

Neighborhood operations: domain and range
- Pset 3
Images in C++
Digital Images

- Can be encoded as 2D arrays of RGB triplets
- Formally:
  - The domain is the 2D plane
  - The range is the RGB space
- Other color spaces are possible and will be used
- Values often encoded as 8 or 16 bit integers
  - But we will use floats in [0...1] to make life easier
Arrays

• C++ vector
  - dynamically sized
  - templatized by type, float in our case

• Created with

• data = std::vector<float>(size,initialValue);
  - for a float array with an initial value

• References:
1D to 3D

• vectors only have one 1D index

2D index → 1D index

\[ y \times width + x \]

3D images

\[ y \times width \times nChannels + x \times nChannels + c \]
1D to 3D

• vectors only have one 1D index
• turn 2D index into 1D one through strides
  - pixel at $x, y$, stored at $y \times \text{width} + x$
  - width is called the “stride”

| 0 1 2 3 |
| 4 5 6 7 |
| 8 9 10 11 |
| 12 13 14 15 |

2D array

| 0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 |

1D vector encoding

• For 3D index $x, y, c$ where $c$ is the color channel:
  - $y \times \text{width} \times \text{numChannels} + x \times \text{numChannels} + c$
Our Image class
Our Image class

- Stores a vector of pixel values
  - size width*height*channels
  - float

- number of dimensions
  - could be used to represent 2D images with a single channel, or even 1D arrays

- size of various dimensions

- gamma (more later)

- name string
Our Image class

- Access data using `im(x, y, c)`
  - weird for C++, but more coherent with Halide
File formats

• We’ll use PNG
  - simpler to read (no need for libjpeg)
  - can be converted to and from with online tools
  - unfortunately a little slow to read
  - unfortunately a little big

• We’ll talk about how JPEG works later
Light & perception
Light matter, eyes

• Light from sources is reflected by objects and reaches the eye.
• The amount of light from the source gets multiplied by the object reflectance
  - on a per-wavelength basis
Human perception

- Our eyes have an uncanny ability to discount the illumination
  - Only objects really matter for survival
  - Light is only useful to understand if you’re a photographer or to choose your sun lotion
Illusion by Adelson

• A & B have exactly the same tone
Illusion by Adelson

- A & B have exactly the same tone
Illusion by Adelson

- A & B have exactly the same tone
Illusion by Adelson

- A & B have exactly the same tone
Mechanism to discount light

- **Light adaptation**
  - We recenter our neural response around the current average brightness
  - neural + chemical + pupil

- **Chromatic adaptation**
  - eliminate color cast due to light sources
  - e.g. Daylight is white but tungsten is yellowish
  - Related to white balance - more soon
  - and Spanish Castle illusion
Contrast is about ratios

- Contrast between 1 & 2 is the same as between 100 & 200
- Useful to discount the multiplicative effect of light

![Contrast examples](image)

- 0.1 to 0.2
- 0.4 to 0.8
Exposure

• On cameras, exposure (shutter speed, aperture, ISO) has a multiplicative effect on the values recorded by the sensor.

• Changes the “brightness”, not contrast

http://photographystepbystep.com/exposure-2/auto-bracketing/
Take-home message

• Multiplication and ratios matters
  - Lot of perceptual aspects of images are invariant to a multiplication by a scalar.
  - Contrast is about ratios
White balance
White balance & Chromatic adaptation

• Different illuminants have different color temperature

• Our eyes adapt to this: Chromatic adaptation
  – We actually adapt better in brighter scenes
  – This is why candlelit scenes still look yellow

White balance problem

• When watching a picture on screen or print, we adapt to the illuminant of the room, not that of the scene in the picture.
• The eye cares more about objects’ intrinsic color, not the color of the light leaving the objects.
• We need to discount the color of the light source.

Same object, different illuminants
White balance & Film
White balance & Film

• Different types of film for fluorescent, tungsten, daylight
• Need to change film!

• Electronic & Digital imaging are more flexible
Von Kries adaptation

• Multiply each channel by a gain factor
  – $R’ = R \ast k_r$
  – $G’ = G \ast k_g$
  – $B’ = B \ast k_b$

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

• Multiply each channel by a gain factor
• Note that the light source could have a more complex effect
  – Arbitrary 3x3 matrix
  – More complex spectrum transformation

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

- How do we find the scaling factors for r, g, and b?
Best way to do white balance

\[ r_g \cdot g_g \cdot b_g = \frac{1}{r_g} \cdot \frac{1}{g_g} \cdot \frac{1}{b_g} \]

18%
Best way to do white balance

• Grey 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 $k/r_w$, $k/g_w$, $k/b_w$
where $k$ controls the exposure
Lightroom demo

• Most photo editing software lets you click on a neutral object to achieve white balance
  • In “Levels” in Photoshop
  • In “basic” in Lightroom
  • In Adjustments in Aperture
• You also often have presets such as daylight, tungsten
Party name tags

• Provide excellent white references!
Without grey cards

• We need to “guess” which pixels correspond to white objects
Grey world assumption

• Assume average color in the image is grey
• Use weights proportional to

\[
\frac{1}{\int_{\text{image}} r}, \frac{1}{\int_{\text{image}} g}, \frac{1}{\int_{\text{image}} b}
\]

• Usually assumes 18% grey to set exposure
Brightest pixel assumption

- Highlights usually have the color of the light source
  - At least for dielectric materials
- White balance by using the brightest pixels
  - Plus potentially a bunch of heuristics
  - In particular use a pixel that is not saturated/clipped
Light with color control

- Gel on flash
Refs

• Recent work on color constancy
  – http://gvi.seas.harvard.edu/paper/color-subspaces-photometric-invariants

• Still an open problem!
INSTEAD OF DRIVING ALL THIS WAY, WE COULDN'T HAVE JUST TAKEN OUR SUMMER PICTURES AND MESS IN THE "HUE" SLIDER IN PHOTOSHOP.

HUSH.
Take home messages

- Discounting the illumination is useful
- Ratios matter
- Optical illusions are not optical but fun
Gamma
Problem with linear encoding

- e.g., 8-bit values: 0..255
Linearity and gamma

- Images are usually gamma encoded
- Instead of storing the light intensity $x$, they store $x^\gamma$
- To get more precision in dark areas for 8-bit encoding
Linearity and gamma

- Images are usually gamma encoded
- Instead of storing the light intensity $x$, they store $x^\gamma$
- to get more precision in dark areas

6 bit encoding for emphasis:

Linear

Gamma2.2
Gamma demo

- thank you Lili & Charlotte
Linearity and gamma

• Images are usually gamma encoded
• Instead of storing the light intensity $x$, they store $x^\gamma$

• Half of image processing algorithms work better in linear space
  - If linearity is important
  - To deal with ratios and multiplicative factors better

• Half work better in gamma space
  - closer to logarithmic scale
How to capture linear images

- http://www.mit.edu/~kimo/blog/linear.html
Take home message

• Images are usually gamma-encoded
  - gamma 2.2
  - provides better quantization
  - sometimes good for algorithms
  - sometimes bad
    - convert to linear values!