COMS W3101-2

Programming Languages: MATLAB

Lecture 4

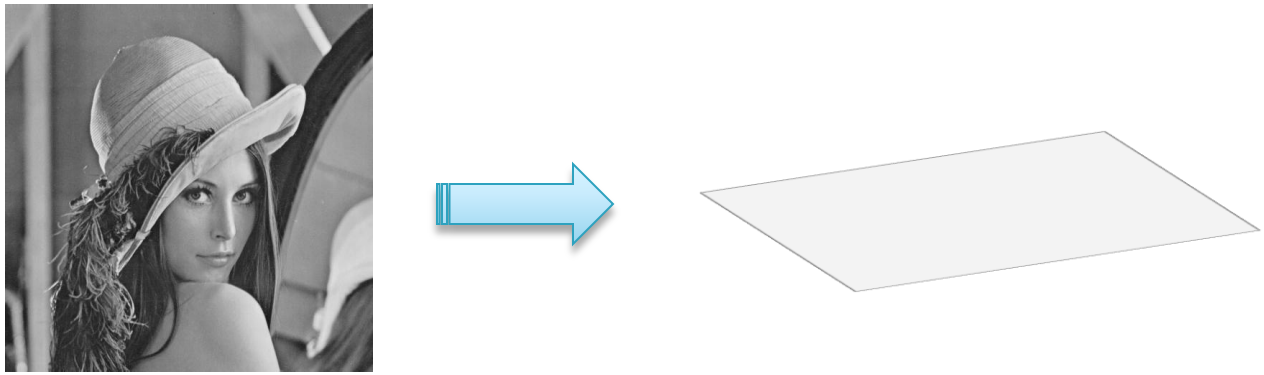
Spring 2010

Instructor: Michele Merler

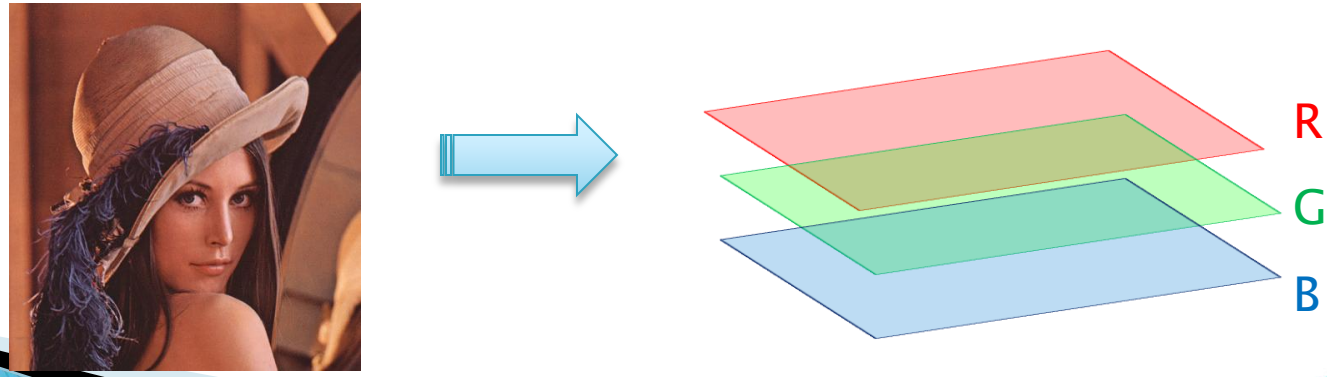
http://www1.cs.columbia.edu/~mmerler/comsw3101-2.html

Images

- Images are matrices (for MATLAB)
  - Grayscale images are [nxm] matrices

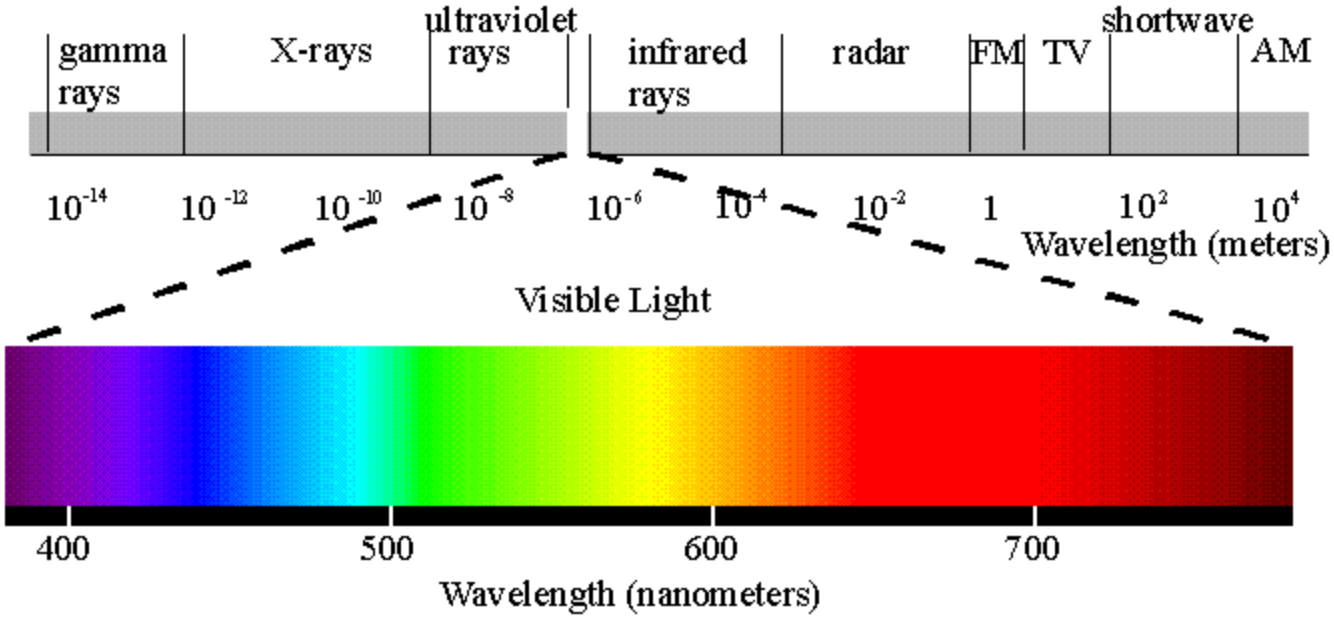


  - Color images are [nxmx3] matrices



Images – Tristimulus Theory

- Why RGB?
- Visible light spectrum



Images – Tristimulus Theory

- Why RGB?
- Human Eye Reception System

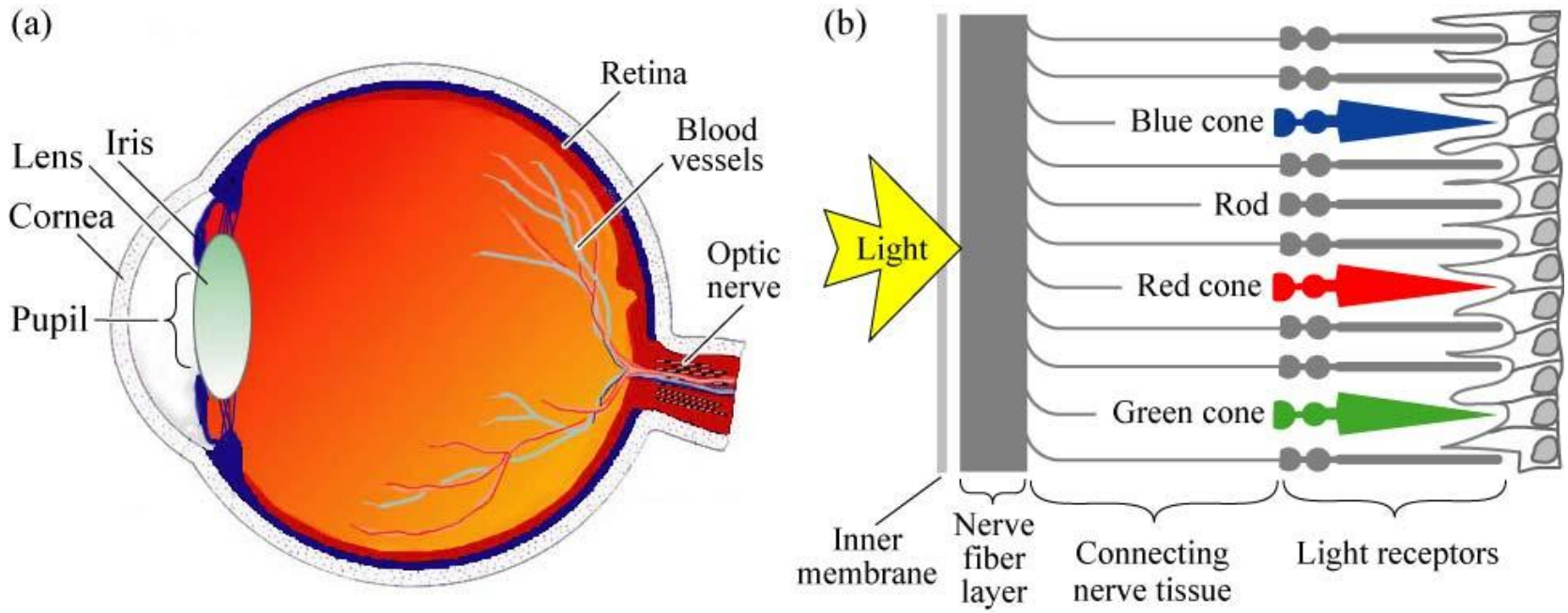


Fig. 16.1. (a) Cross section through a human eye. (b) Schematic view of the retina including rod and cone light receptors (adapted from Encyclopedia Britannica, 1994).

E. F. Schubert
Light-Emitting Diodes (Cambridge Univ. Press)
www.LightEmittingDiodes.org

Images – Tristimulus Theory

- Why RGB?
- Human Eye Reception System

Rods Respond to light intensity (i.e. grayscale) only

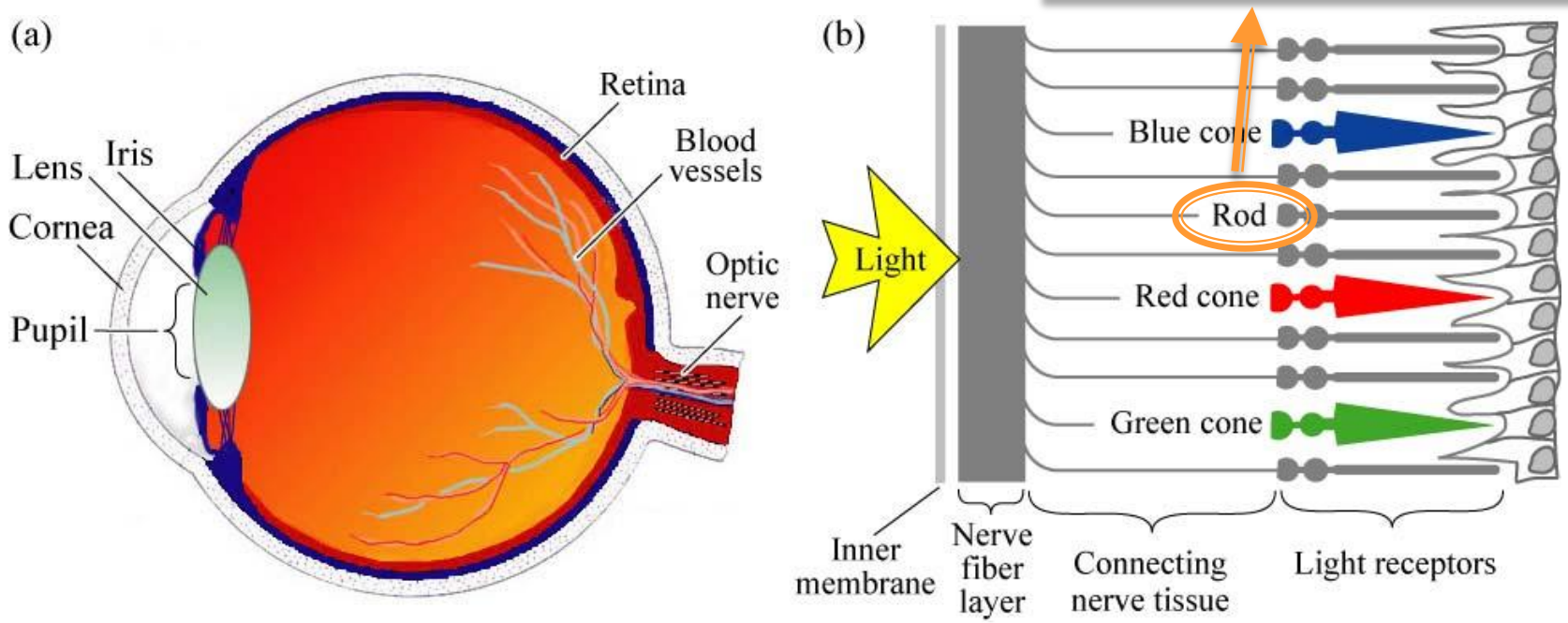
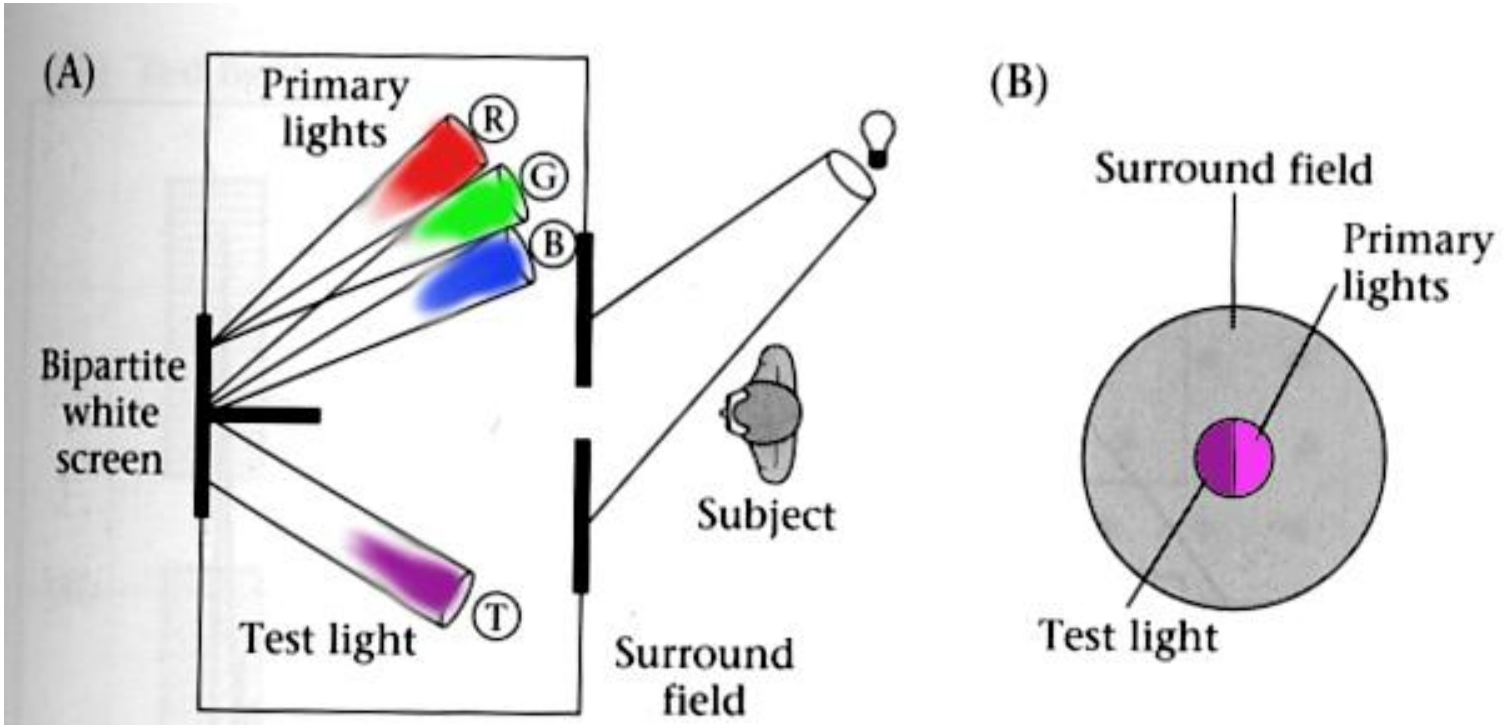


Fig. 16.1. (a) Cross section through a human eye. (b) Schematic view of the retina including rod and cone light receptors (adapted from Encyclopedia Britannica, 1994).

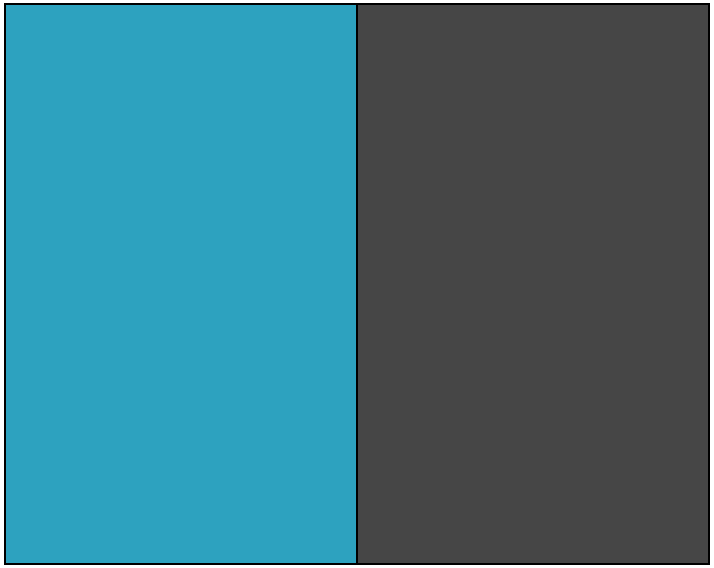
Color matching experiment

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

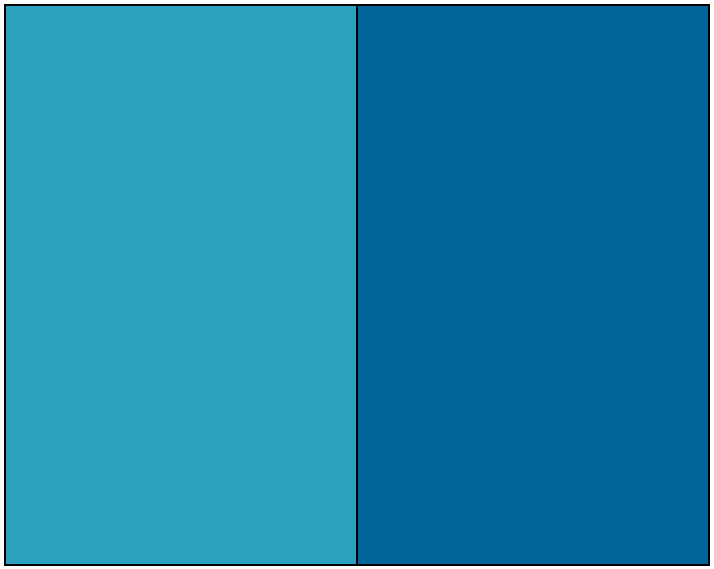


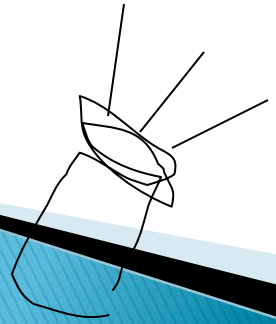
4.10 THE COLOR-MATCHING EXPERIMENT. The observer views a bipartite field and adjusts the intensities of the three primary lights to match the appearance of the test light. (A) A top view of the experimental apparatus. (B) The appearance of the stimuli to the observer. After Judd and Wyszecki, 1975.

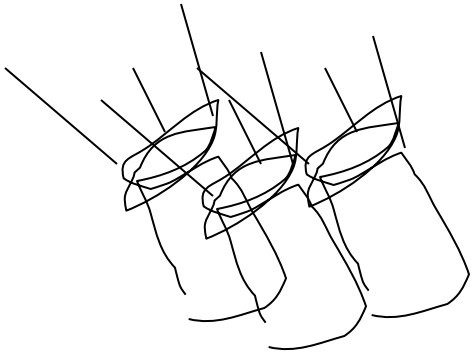
Color matching experiment 1

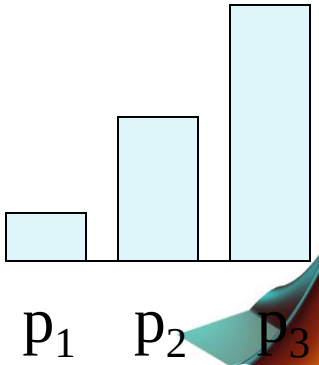


Color matching experiment 1

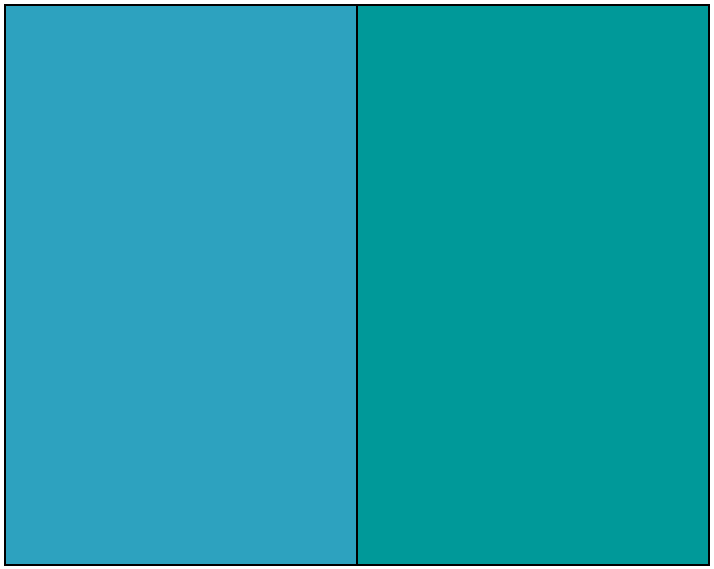


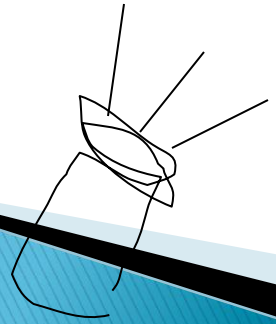


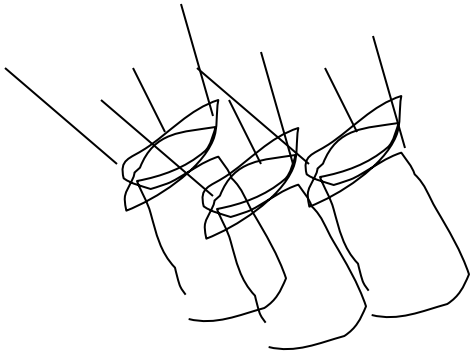


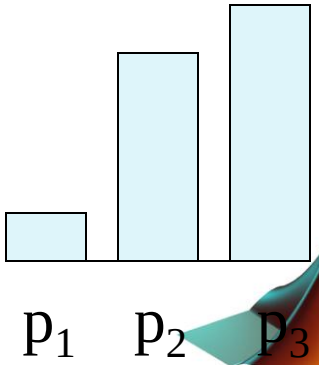


Color matching experiment 1

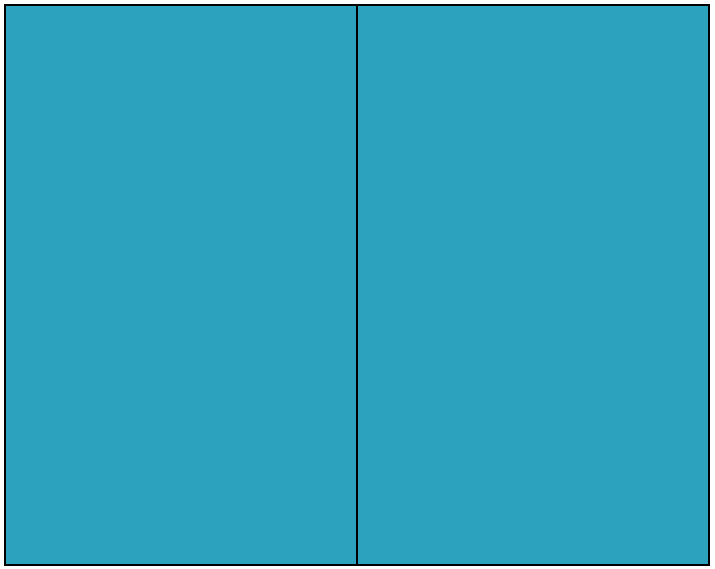




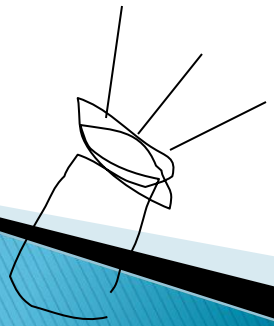


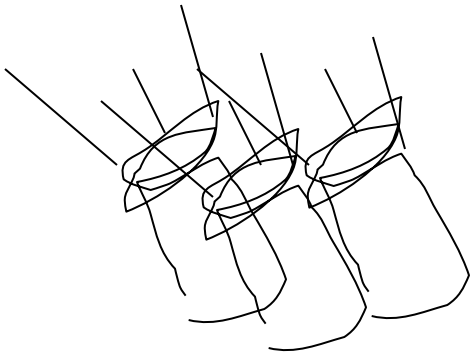


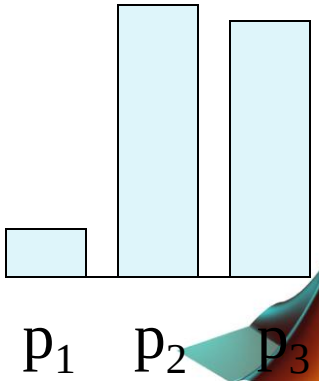
Color matching experiment 1



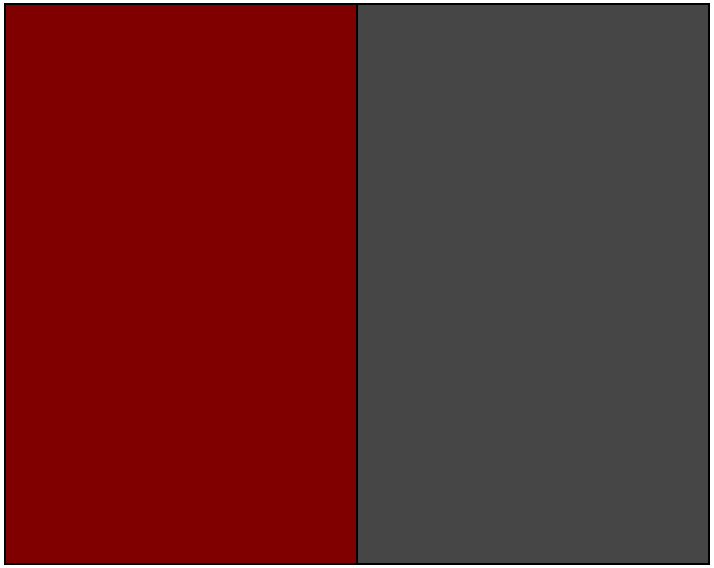
The primary color amounts needed for a match



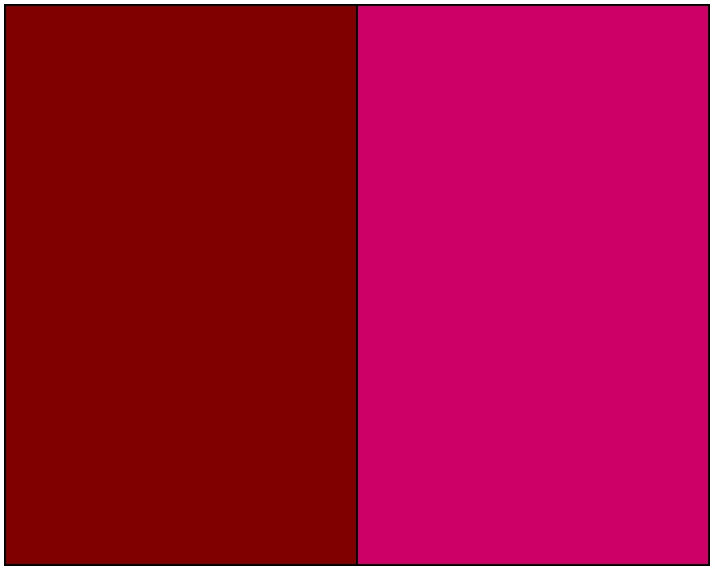


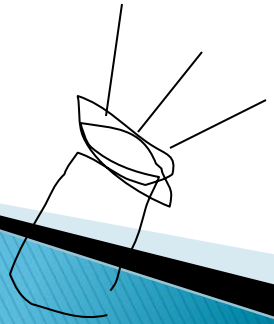


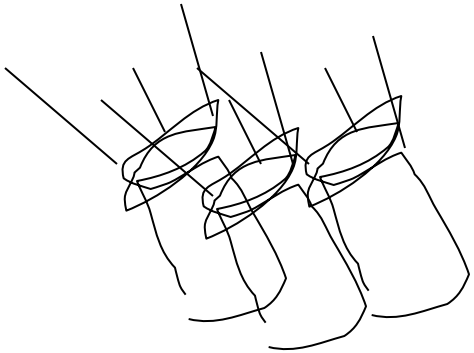
Color matching experiment 2

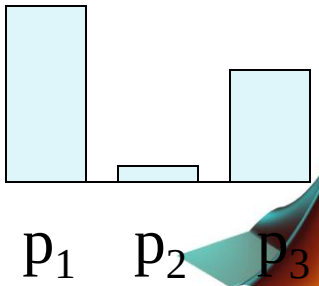


Color matching experiment 2

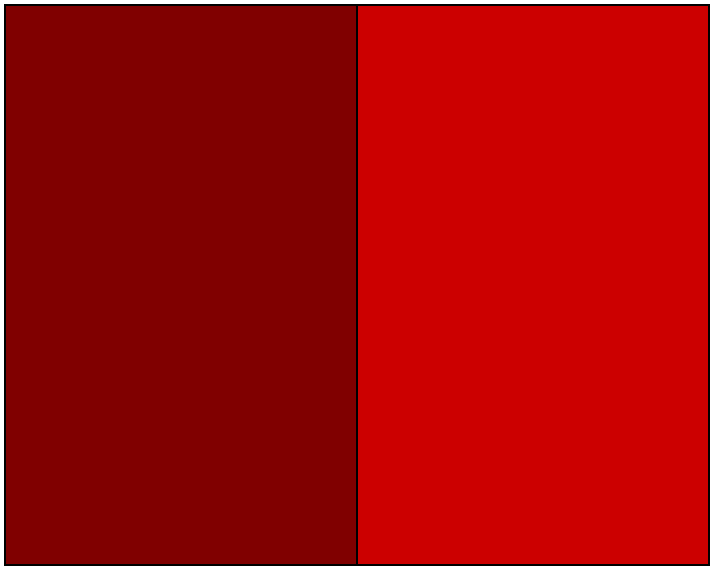


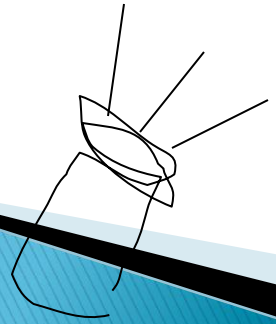


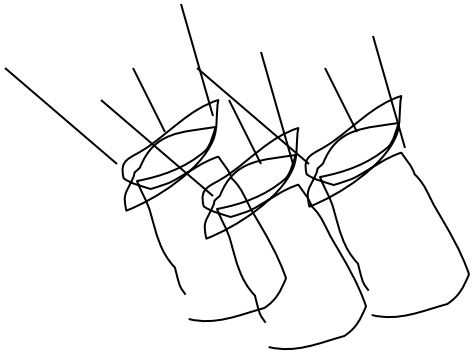


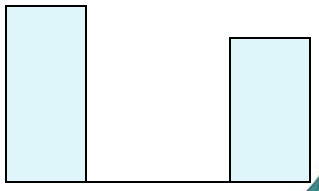


Color matching experiment 2



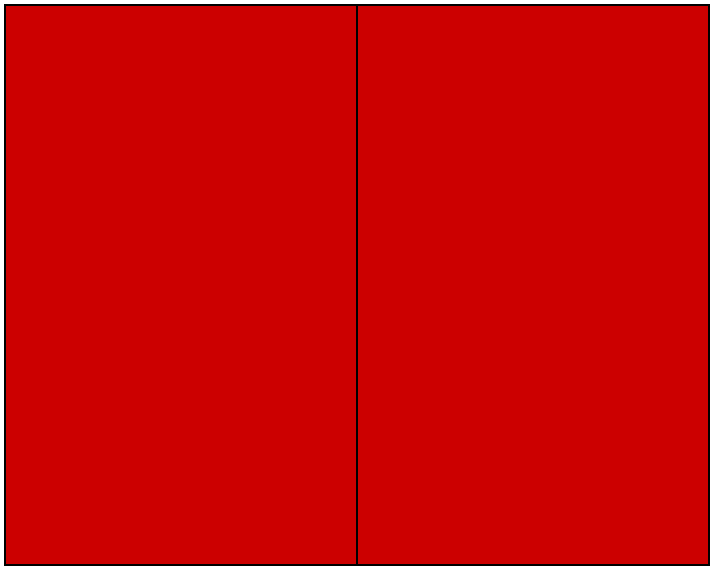




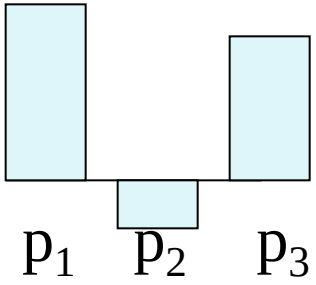


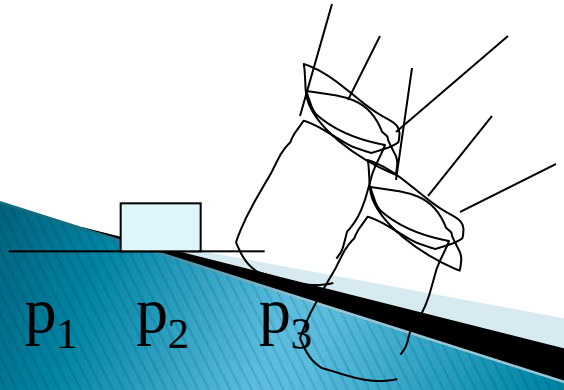
Color matching experiment 2

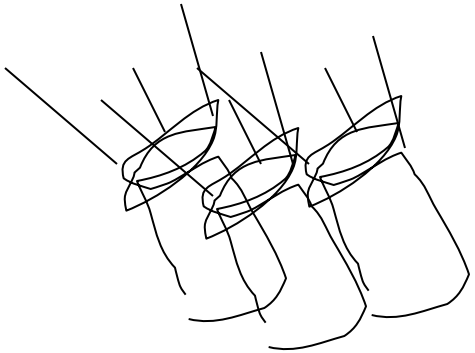
We say a "negative" amount of p_2 was needed to make the match, because we added it to the test color's side.

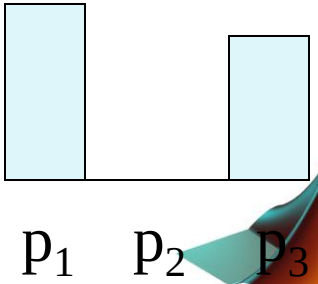


The primary color amounts needed for a match:









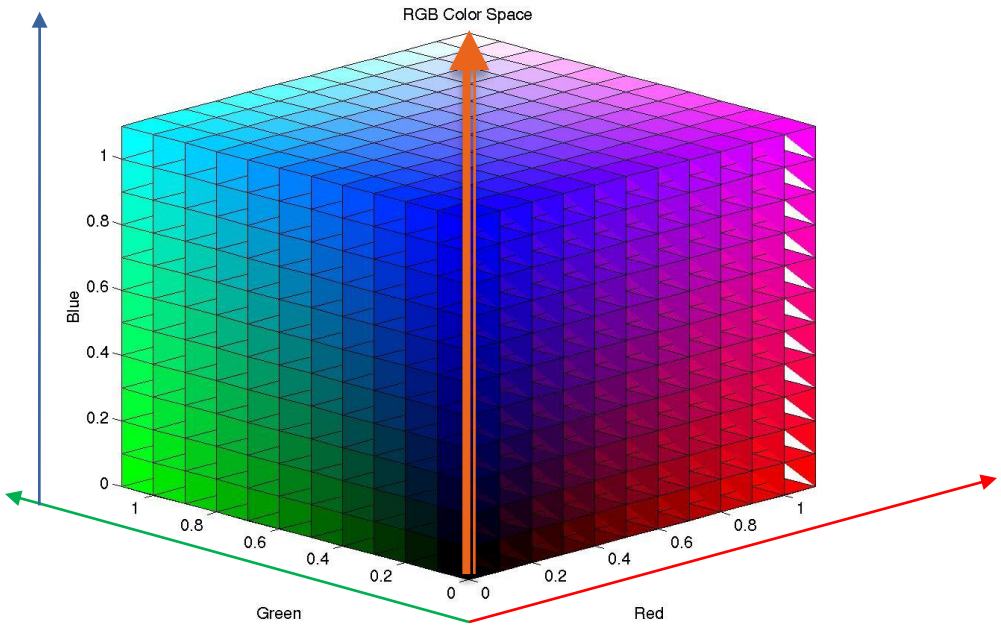
Color Matching Experiment

- Pick a set of 3 primary color lights.
- Find the amounts of each primary, e_1, e_2, e_3, needed to match some spectral signal, t.
- You have a 3 channel representation of any color!
- Since the human eye is most responsive to Red, Green and Blue, we use RGB as standard color space

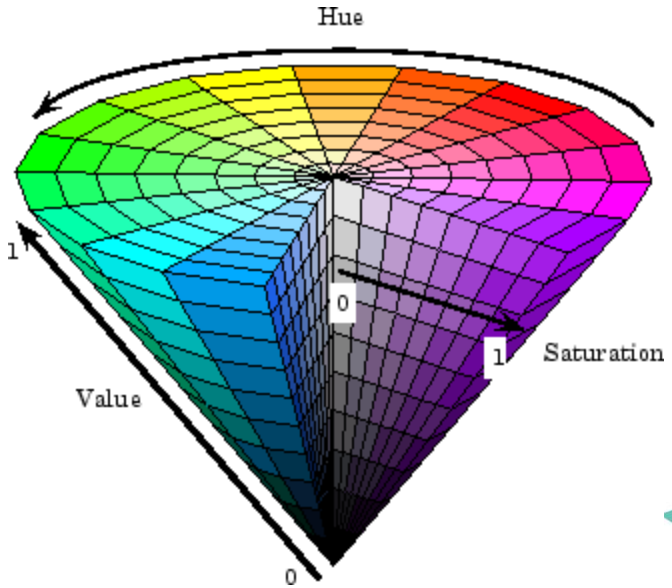
Color Spaces

YCbCr, Lab, XYZ…

- RGB
  - Intensity along the diagonal
  - If R=G=B ↦ Gray Value



- HSV
  - Intensity (V) and Color (HS) are separated



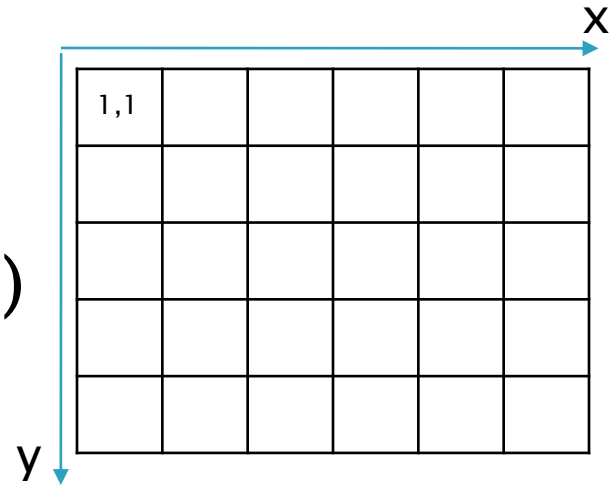
MATLAB Color Representation

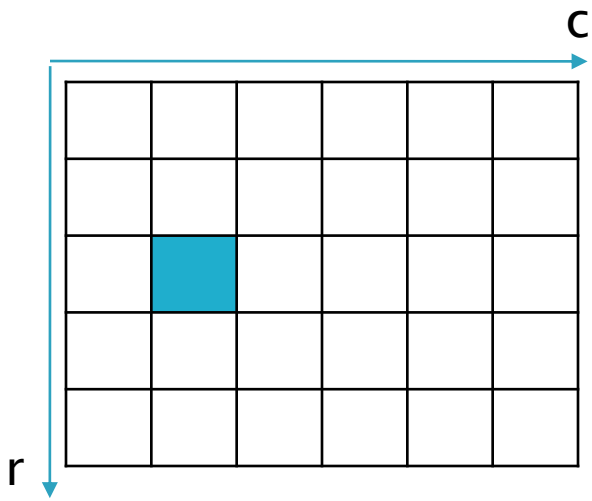
- Each color is represented as a triplet [R G B]
- Usually images use 8 bits to represent each color channel
- [R G B] = 8 + 8 + 8 = 24 bits
- Each channel has a value between 0 and 255
- [0 0 0] → Black
- [255 255 255] → White
- [255 0 0] → Red
- [0 255 0] → Green
- [0 0 255] → Blue
- [255 255 0] → Yellow

Images

- Images are matrices (for MATLAB)
  - Im = [50x50] matrix
  - Im(3,2) element at row 3, column 2

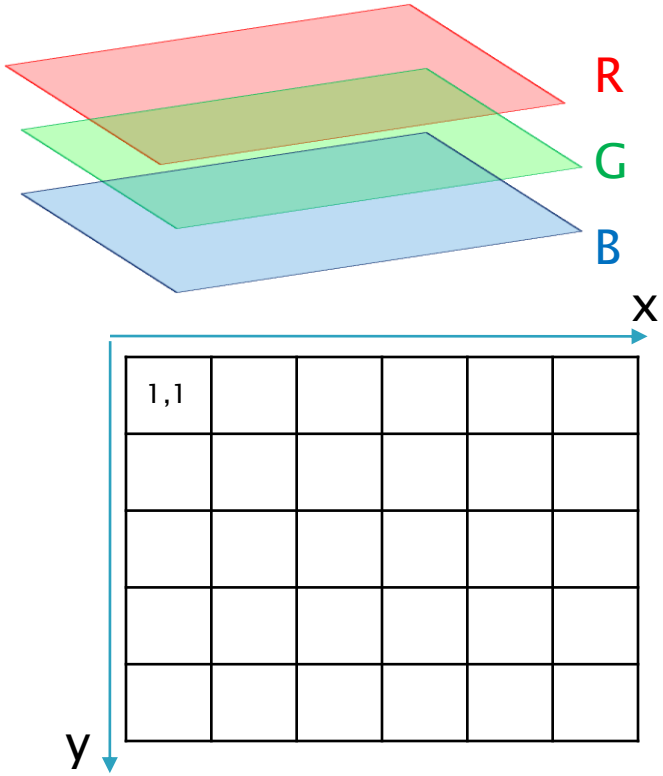
    pixel at coordinate y 3, x 2





I/O – Images

- Images are matrices
  - Color images are [nxmx3] matrices
  - Grayscale images are [nxm] matrices



- Reading Images
- `imread`
  - `Im = imread('mypic.jpg');`

`imread` loads images as uint8 !

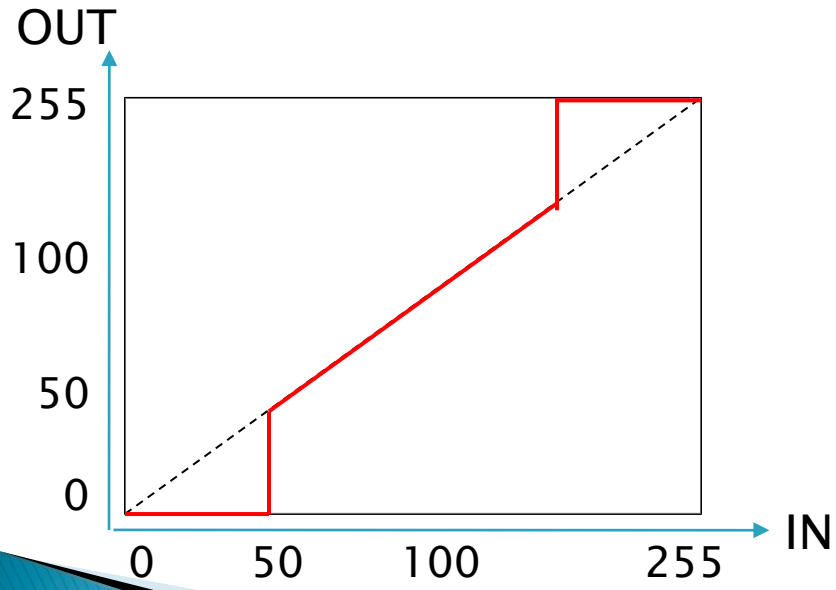
- Saving Images
- `imwrite`
  - `imwrite(Im,'mypic2.png');`

Visualizing Images

- `image()`
  - displays the image
- `pixval`
  - shows pixel values as we hover mouse over image
- `imagesc()`
  - **scales** image data to the full range of the current colormap and displays the image
- `imshow()`
  - displays the image, you can also set the intensity range
  - `imshow(Im, [50 100]);`

Visualizing Images

- `imshow()`
  - displays the image, you can also set the intensity range
  - `imshow(Im, [50 100]);`



Color Conversions

- `rgb2gray()`
  - Converts from RGB to grayscale

- `rgb2hsv()`
  - Converts from RGB to HSV

Color Conversions

- Example

```
Im = imread('BillWarren.jpg');
ImHSV = rgb2hsv(Im);
ImGray = rgb2gray(Im);
figure;
subplot(2,2,1);
imshow(Im);title('RGB');
subplot(2,2,2);
imshow(ImGray);title('Grayscale');
subplot(2,2,3);
imshow(ImHSV);title('HSV');
subplot(2,2,4);
imshow(ImGray,[50 100]);title('Clipped');
```

Color Conversions

- Example

```
Im = imread('BillWarren.jpg');

ImHSV = rgb2hsv(Im);

ImGray = rgb2gray(Im);

figure;
subplot(2,2,1);
imshow(Im);title('RGB');

subplot(2,2,2);
imshow(ImGray);title('Grayscale');

subplot(2,2,3);
imshow(ImHSV);title('HSV');

subplot(2,2,4);
imshow(ImGray,[50 100]);title('Clipped');
```

Some Image Functions

- imresize()
  - Im3 = imresize(image, times);
  - Im3 = imresize(Im, 3, 'method');

    'nearest'
    'bilinear'
    'bicubic'

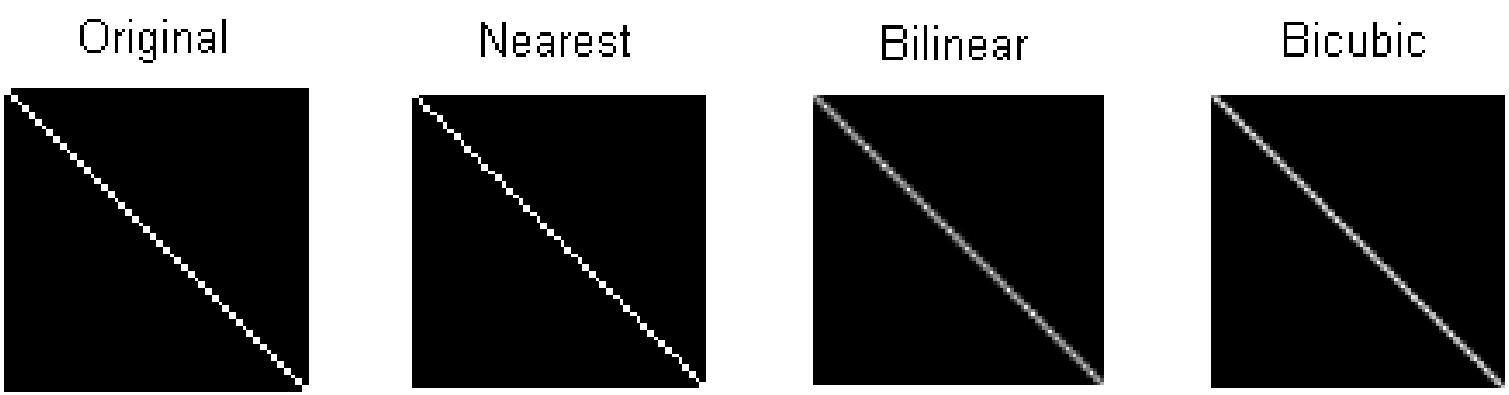
  <u>Example</u>

  - Im = zeros(50,50);
  - for i=1:50
  - Im(i,i)=1;
  - end
  - Im2 = imresize(Im,10,'nearest');
  - Im3 = imresize(Im,10,'bilinear');
  - Im4 = imresize(Im,10,'bicubic');

Some Image Functions

- `imresize()`
  - `Im3 = imresize(image, times);`
  - `Im3 = imresize(Im, 3, 'method');`

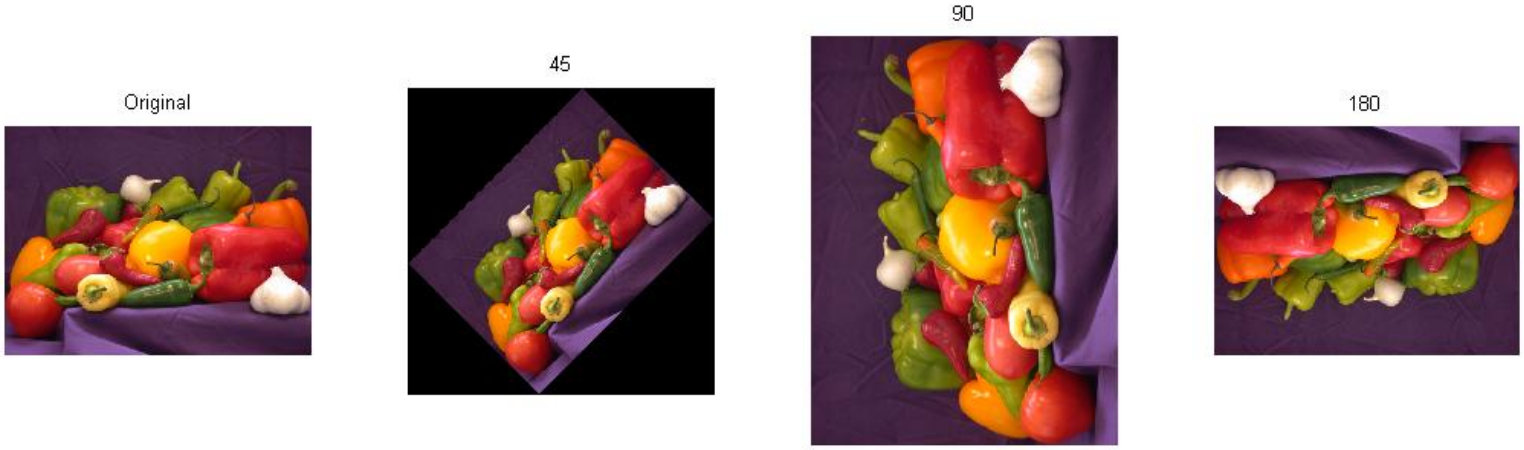
'nearest'
'bilinear'
'bicubic'



Some Image Functions

- imrotate()
  - imrotate(image, angle, 'method')

  - Im = imread('peppers.png');
  - Im2 = imrotate(Im,45,'nearest');
  - Im3 = imrotate(Im,90,'bilinear');
  - Im4 = imrotate(Im,180,'bicubic');



Interacting with images

- `ginput()`
  - gets the points clicked with the mouse over the image
  - `[x y] = ginput;`
    - saves points until we press the 'Return' key
  - `[x y] = ginput(n);`
    - saves the first *n* points clicked

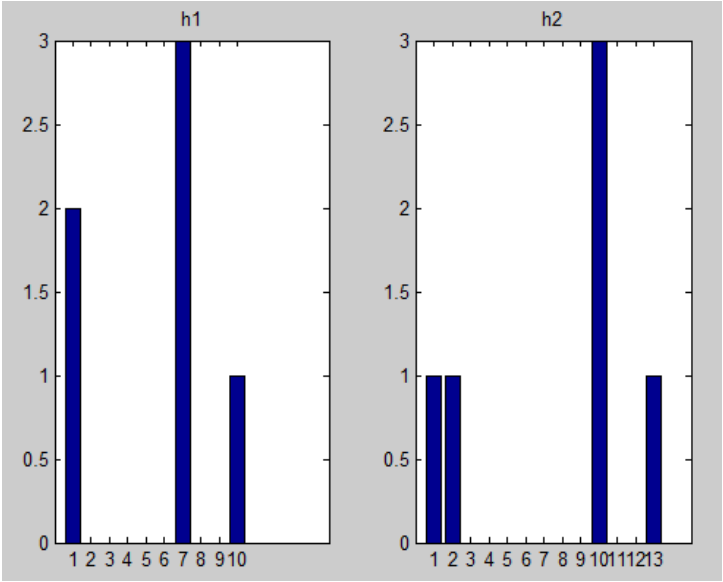
- `imcrop()`
  - crops one part of the image
  - `Imcropped = imcrop(Im);`
  - `[Imcropped, rect] = imcrop(Im);`
  - `Imcroppd = imcrop(Im, rect);`

rect = [xmin ymin width height]

Histogram

- Histogram: the histogram counts the number of elements with a specific value present in a vector (or matrix, or image)

- `hist()`
  - `x = [10 10 10 13 3 4];`
  - `h1 = hist(x);`
  - `h2 = hist(x,13);`



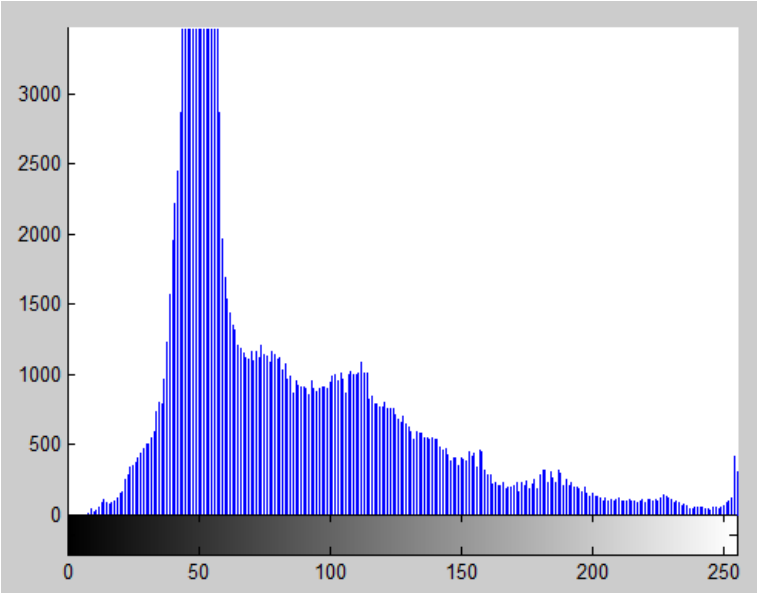
Histogram

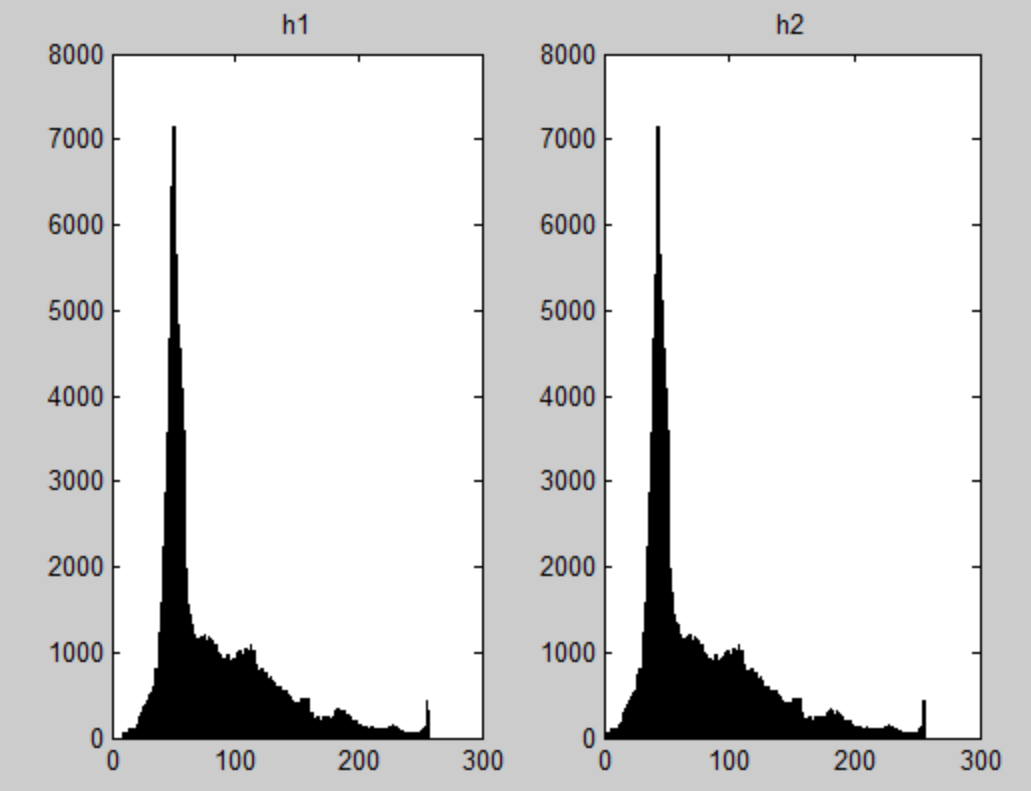
- imhist()
  - histogram for images

```
Im = imread('peppers.png');
ImGray = rgb2gray(Im);
imhist(ImGray);
h1 = imhist(ImGray);
h2 = hist(double(ImGray(:)),256);
figure
subplot(1,2,1)
bar(h1);title('h1')
subplot(1,2,2)
bar(h2);title('h2')
```

Histogram

- `imhist()`
  - histogram for images





CumSum

- `cumsum()`
  - computes the cumulative sum of a vector
  - `x = ones(1,10);`
  - `y = cumsum(x);`
  - `bar(y);`

Animations

- AVI movies or animated GIFs

```
Im = imread('peppers.png');
for t=1:30
  Im = Im - 5;
  imshow(Im);
  pause(.5);
end
```

Animations

- AVI movies or animated GIFs

```
Im = imread('peppers.png');
for t=1:30
  Im = Im - 5;
  imshow(Im);
  M(t) = getframe;
end
```

Save animation in variable M

Animations

- `movie()`
  - `movie(var,times,fps)`

<u>Example</u>

- `movie(M,2,15);`

Movies

Read/save movies

- `aviread()`
- `movie2avi(var,filename,'parameter','value')`
  - `movie2avi(M,'mymovie.avi','Compression','None');`

Image to frame conversions

- `im2frame()`
- `frame2im()`

Homeworks policy

- Due at beginning of class, no exceptions
- Put your code (.m files) and additional files in a single folder, name it *youruni*_hw_*X* and zip it
- Upload the zipped folder to CourseWorks
- Bring a printout of your code to class
- Good luck and have fun !