Programming Languages: MATLAB

Lecture 4

Spring 2010
Instructor: Michele Merler

Images are matrices (for MATLAB)

- Grayscale images are \([n \times m]\) matrices
- Color images are \([n \times m \times 3]\) 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).
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

$P_1$ $P_2$ $P_3$
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:

\[ p_1 \quad p_2 \quad p_3 \]
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

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

- **HSV**
  - Intensity (V) and Color (HS) are separated
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)
  - \( \text{Im} = [50 \times 50] \) matrix
  - \( \text{Im}(3,2) \) 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**
    - \(\text{Im} = \text{imread}('\text{mypic}.jpg');\)
    
    **imread** loads images as uint8!

- Saving Images
  - **imwrite**
    -  \(\text{imwrite}(\text{Im}, '\text{mypic2.png}');\)
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
Example

- \( \text{Im} = \text{imread}(	ext{‘BillWarren.jpg’}); \)
- \( \text{ImHSV} = \text{rgb2hsv}(\text{Im}); \)
- \( \text{ImGray} = \text{rgb2gray}(\text{Im}); \)
- \( \text{figure}; \)
- \( \text{subplot}(2,2,1); \)
- \( \text{imshow}(\text{Im});\text{title(‘RGB’)}; \)
- \( \text{subplot}(2,2,2); \)
- \( \text{imshow}(\text{ImGray});\text{title(‘Grayscale’)}; \)
- \( \text{subplot}(2,2,3); \)
- \( \text{imshow}(\text{ImHSV});\text{title(‘HSV’)}; \)
- \( \text{subplot}(2,2,4); \)
- \( \text{imshow}(\text{ImGray},[50 \hspace{1em} 100]);\text{title(‘Clipped’)}; \)
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');`

Example

- `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');`

Options for `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
  - 
    - \(\text{Imcropped} = \text{imcrop}(\text{Im});\)
  - 
    - \([\text{Imcropped}, \ \text{rect}] = \text{imcrop}(\text{Im});\)
  - 
    - \(\text{Imcroppd} = \text{imcrop}(\text{Im}, \ \text{rect});\)

\(\text{rect} = [\text{xmin} \ \text{ymin} \ \text{width} \ \text{height}]\)
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);`
```plaintext
**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

![Histogram for images](image)
CumSum

- `cumsum()`
  - computes the cumulative sum of a vector

- $x = \text{ones}(1,10)$;
- $y = \text{cumsum}(x)$;
- $\text{bar}(y)$;
AVI movies or animated GIFs

- \( \text{Im} = \text{imread('peppers.png')}; \)
- \( \text{for } t=1:30 \)
  - \( \text{Im} = \text{Im} - 5; \)
  - \( \text{imshow(Im)}; \)
  - \( \text{pause(.5)}; \)
- \( \text{end} \)
Animations

- AVI movies or animated GIFs
  - \( \text{Im} = \text{imread('peppers.png');} \)
  - \( \text{for } t=1:30 \)
  - \( \text{Im} = \text{Im} - 5; \)
  - \( \text{imshow(Im);} \)
  - \( \text{M(t)} = \text{getframe;} \)
  - \( \text{end} \)

Save animation in variable M
Animations

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

Example

- 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!