Tutorial 2
Image Enhancement in the Spatial and the Basic Pipeline for Image Processing in Matlab

COMP 4421: Image Processing

September 16, 2019

Teaching Asistant

- TA: WANG, Jierong
- Email: jwangdh@connect.ust.hk
- Office: Room 4208, Lo Kwee-Seong Medical Image Analysis Laboratory
- Office Hours: By appointment

Outline

- Octave
- Image Enhancement in the Spatial Domain
  - Histogram Equalization
  - Smoothing
  - Sharpening
- Image Processing in Matlab
  - Basic Introduction
  - Vectorization
  - Implement your own filter

Outline

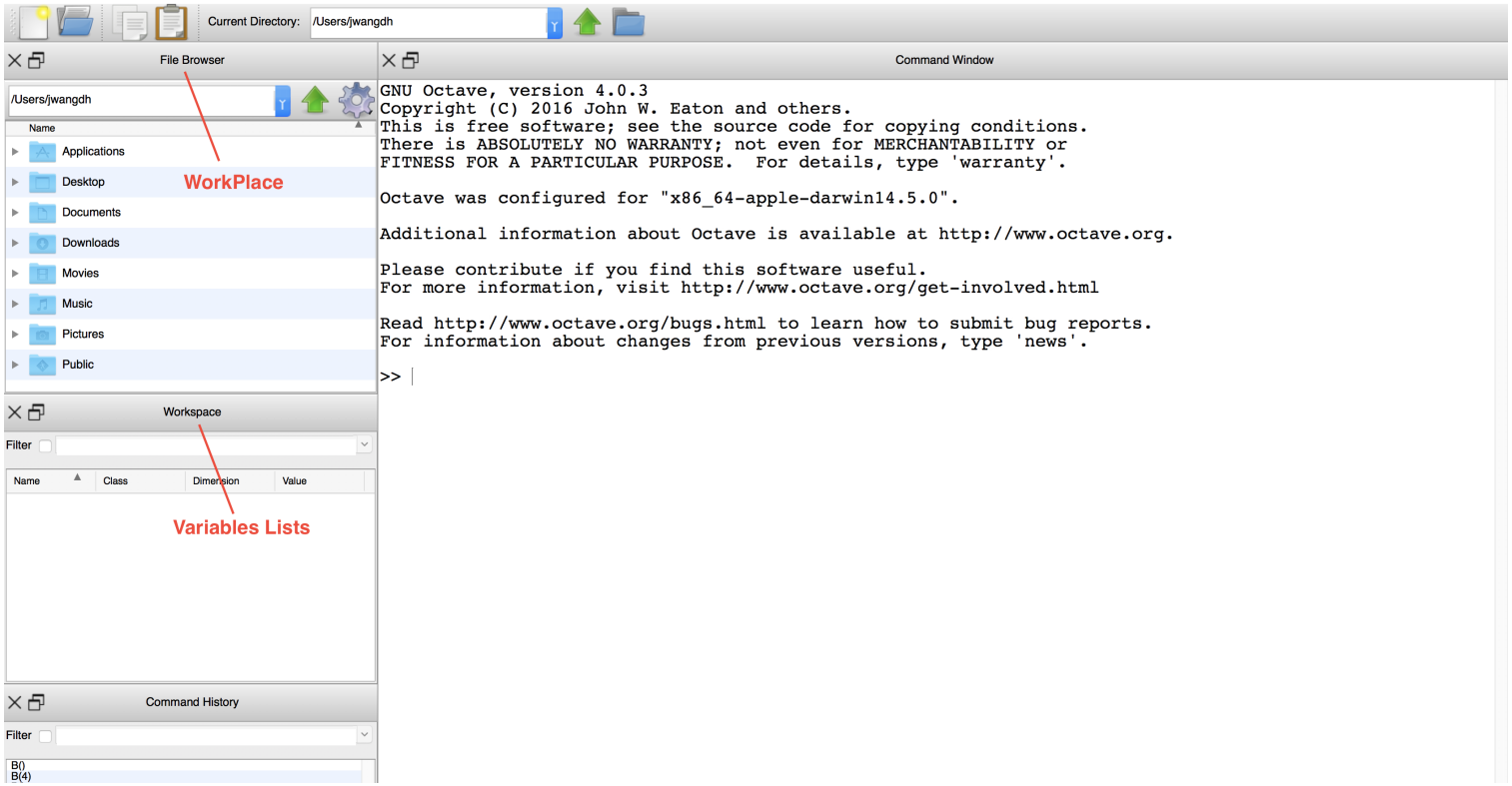
- Octave
- Image Enhancement in the Spatial Domain
  - Histogram Equalization
  - Smoothing
  - Sharpening
- Image Processing in Matlab
  - Basic Introduction
  - Vectorization
  - Implement your own filter

Octave - Description(1)

- Scientific Programming Language
  - Powerful mathematics-oriented syntax with built-in plotting and visualization tools
  - Free software, runs on GNU/Linux, macOS, BSD, and Windows
  - Drop-in compatible with many Matlab scripts
- Link to mainpage: https://www.gnu.org/software/octave/
- Link to Octave Forge: https://octave.sourceforge.io (Download specific packages you may need)

Octave - Description(2)

- interfaces



- Download packages: - pkg install -forge package_name

Octave - Difference

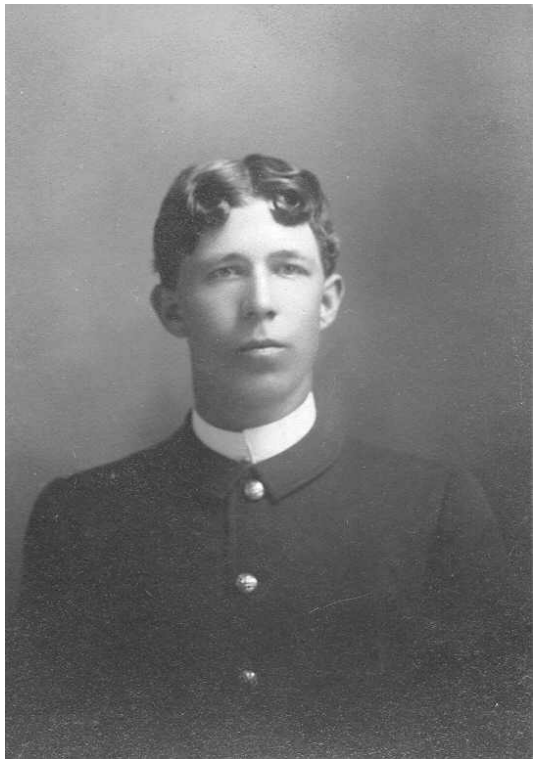
- Difference between Matlab and Octave
  - function definition:
    - function f(a=3)
      if a == 4 a
      else a
      end
      end
  - +=, -= operations:
    - a += 3;
  - Efficiency
  - ...
- Octave has C++ programming style and lower efficiency for computation

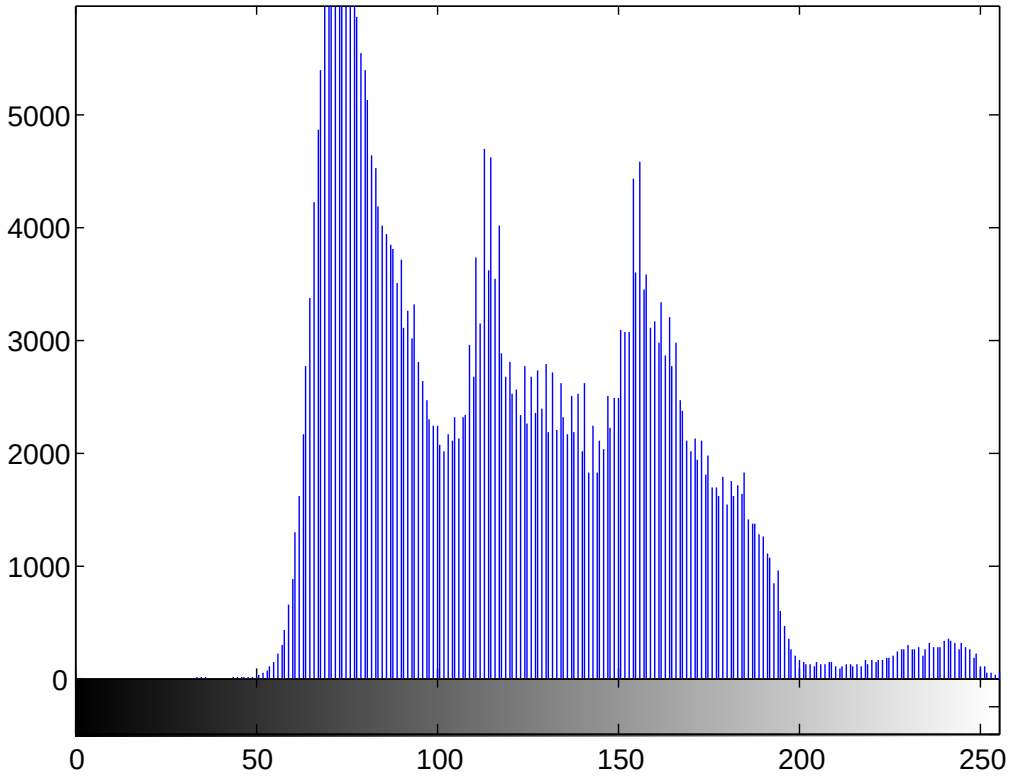
Outline

- Octave
- Image Enhancement in the Spatial Domain
  - Histogram Equalization
  - Smoothing
  - Sharpening
- Image Processing in Matlab
  - Basic Introduction
  - Vectorization
  - Implement your own filter

How to obtain a histogram? (imhist)

- f=imread('*charles_butter_2.jpg*'); imhist(f)

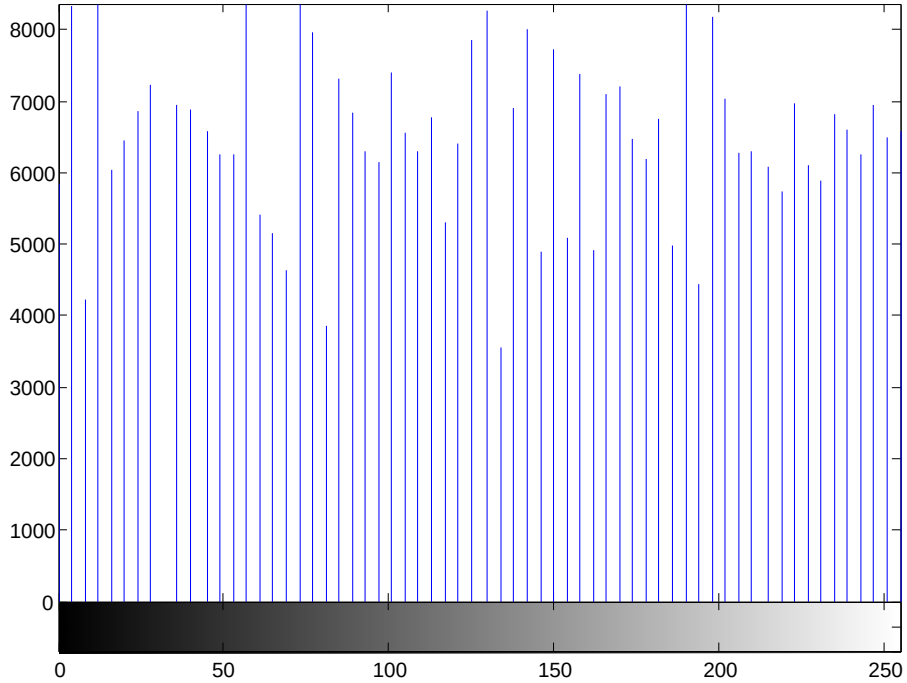




Global Equalization (histeq)

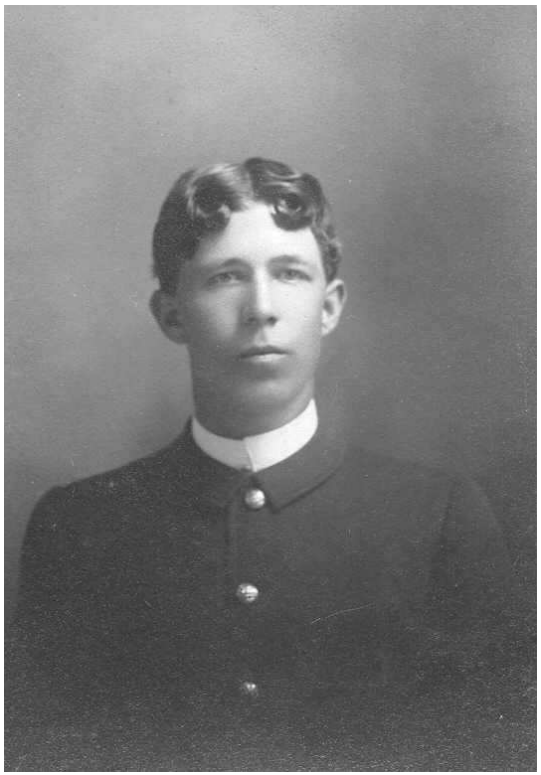
- g = histeq(f); imshow(g)

- imhist(g)



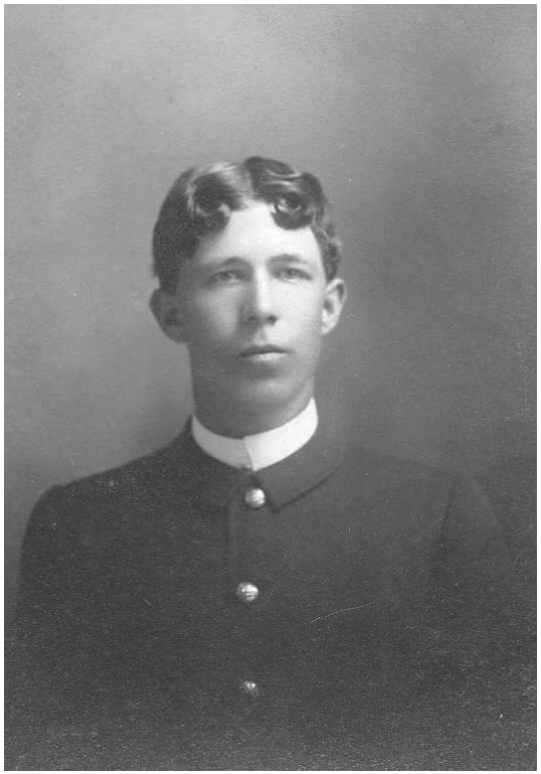
Smoothing via an Average Mask (imfilter, fspecial)

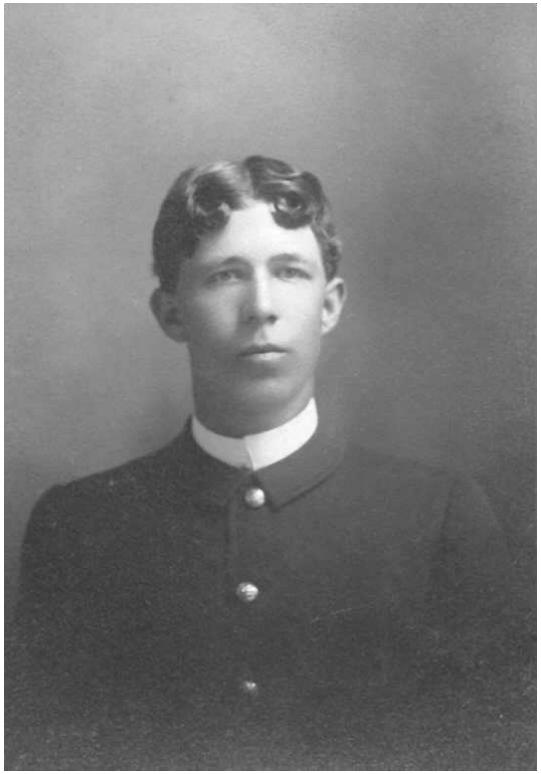
- '>> image_mask.m'
- 3×3: fspecial
- 5×5: $\frac{1}{25} * ones(5, 5)$



Smoothing via a Median Filter (medfilt2)

- '>> image_mask.m'
- 3×3
- 5×5

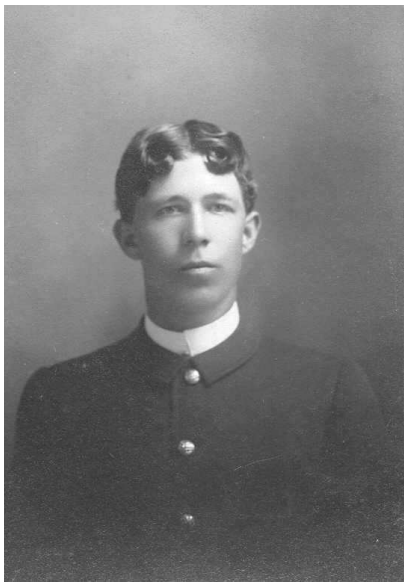
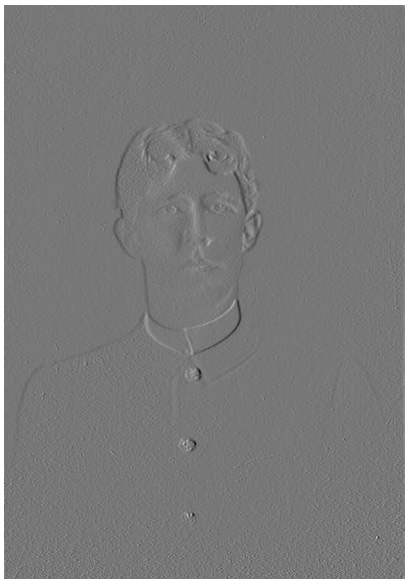
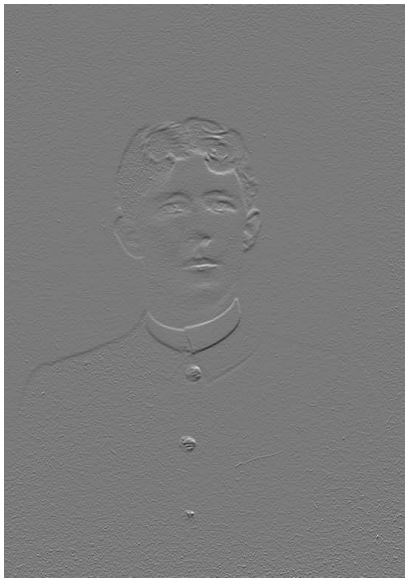
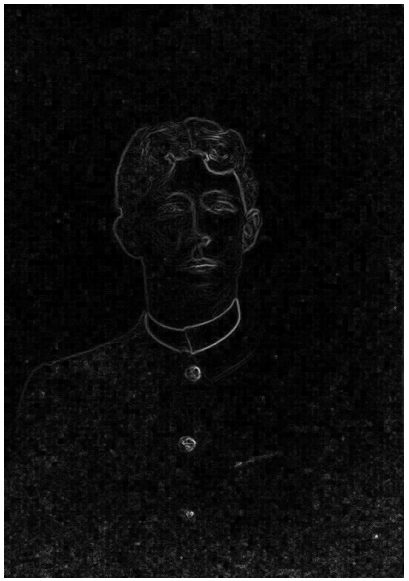




Gradients (gradient)

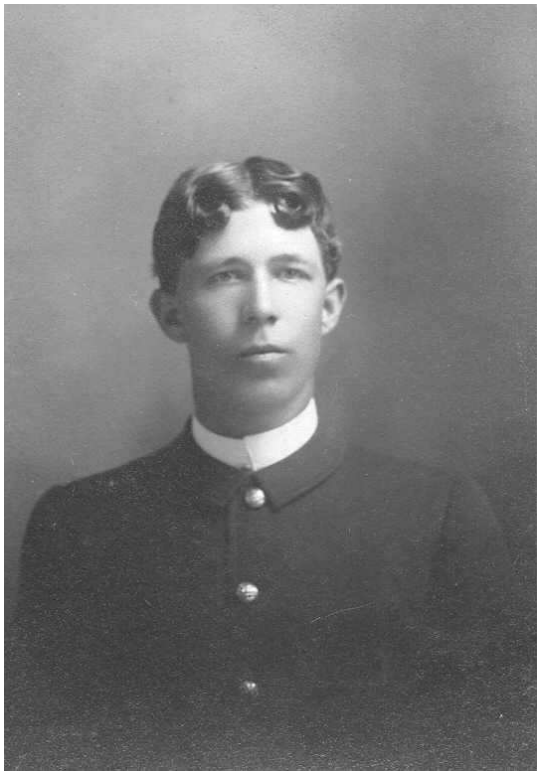
‘>> image_gradient.m‘

- Original
- df/dx
- df/dy
- magnitude

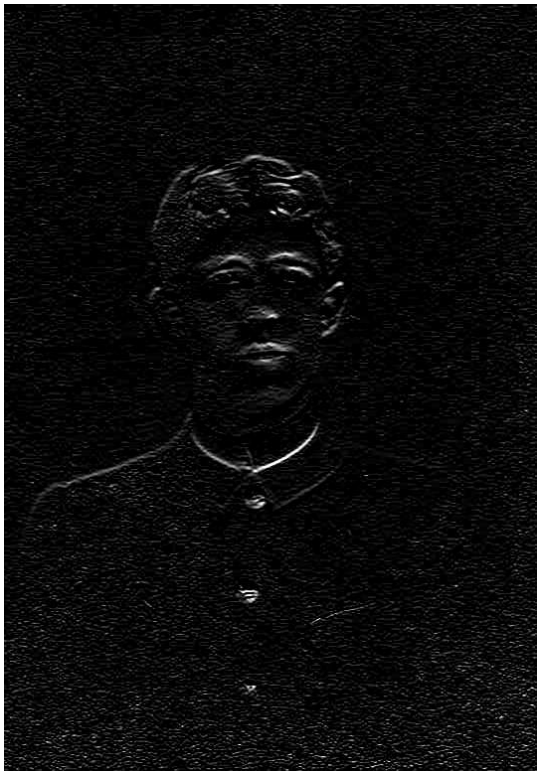





Sharpening via Approximated Derivative Filters (fspecial or define by ourself)

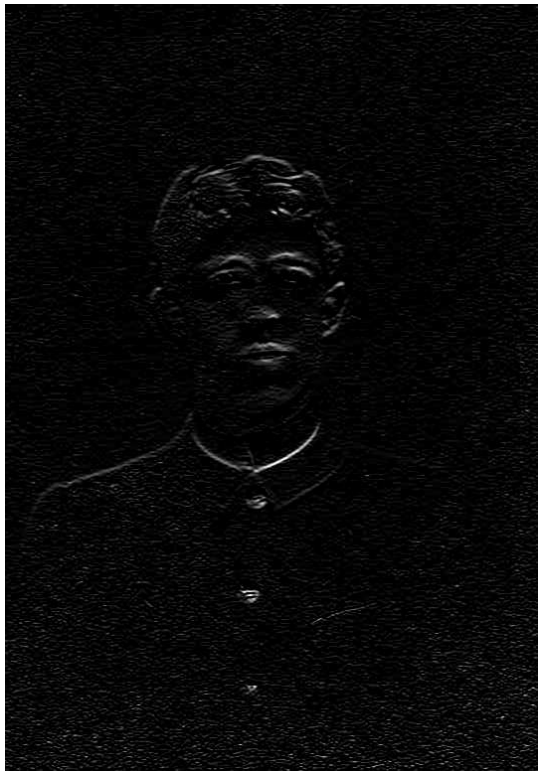
- '>> image_mask.m'



- sobel



- prewitt



Recap

- The following built-in functions are important:
  - imshow
  - imhist
  - histeq
  - imfilter
  - medfilt2
  - gradient
  - fspecial
- Please explore other interesting functions!

Outline

- Octave
- Image Enhancement in the Spatial Domain
  - Histogram Equalization
  - Smoothing
  - Sharpening
- Image Processing in Matlab
  - Basic Introduction
  - Vectorization
  - Implement your own filter

Image Processing in Matlab

- Image Representation: 2D/3D Matrix
  - [0, 255] for uint8 (unsigned int)
  - [0, 1] for double
- Example
  - img = imread('*example_1.png*');
  - imshow(img)
  - img = double(img);
  - img_inv = 255 - img; % inverse
  - img_log = 30*log(1 + img); % log
  - img_pow = 0.1*img.^1.5); % power law
  - img_con = (img>100)*255; % Contrast Stretching
  - subplot(141), imshow(uint8(img_inv)), title('Inverse')
  - subplot(142), imshow(uint8(img_log)), title('Log')
  - subplot(143), imshow(uint8(img_pow)), title('Power')
  - subplot(144), imshow(uint8(img_con)), title('Contrast Stretching')

Image Processing in Matlab

- In conclusion
  - Read an image: imread - [0, 255], uint8
  - Processing: double ([0, 255.0]) or imdouble ([0, 1])
  - Show/Write an image: imshow/imwrite
    - [0, 255], uint8
    - [0, 1], double
  - mat2gray(): rescaling to [0, 1] (double)

Vector Computation

- MATLAB is optimized for operations involving matrices and vectors
- The process of revising loop-based, scalar-oriented code to use MATLAB matrix and vector operations is called vectorization
- Vectoring your code will save time and make you program easy to read.
- Example: Compute 1*1 + 2*2 + 3*3
  - a = [1,2,3], b = [1,2,3], s = 0;
  - for i = 1:3
  - s = s + a[i]*b[i];
  - end
- Much easier way: s = a * b';
- Practice: Histogram Equliazation

More Vectoring

- https://www.mathworks.com/help/matlab/matlab_prog/vectorization.html

Implement your own filter

1. Read an image
2. Convert the type of the matrix
3. Boundary problem
4. Two-loop or one-loop image retrieval
5. Convert the type of the matrix

Thank you!