CSIT 5410 (Spring 2021)

Assignment 1

Jierong WANG
jwangdh@connect.ust.hk

Task 1 – Image Quantization

- *myquantizer.m:* takes a gray image and a target number of gray levels as input, and generates the quantized image as output ;
- Note that, computers always represent “white” via the pixel value of 255(for uint8). For example, when “level” == 4, the resulting image should contain pixels of {0, 85, 170, 255}, instead of {0, 1, 2, 3}.
- Sample: *out_img = myquantizer(img, 4);*

Task 1 – Image Quantization

2

4

8

32

128

Original: 256

Task 2 – Image Rescaling

- *myscaler.m:* takes a gray image and a target size as input, and generates the scaled image as output;
- The new size may not be strictly larger or smaller than the original size. That is, the case that 200x300 -> 100x400 will exist;
- Interpolation: nearest, bilinear, bicubic, …
- Sample: *out_img = myscaler(img, [100, 400]);*
- *out_img = myscaler(img, [250, 255]);*

Task 2 – Image Rescaling

Original: 384x256

300x450

200x500

200x300

128x192

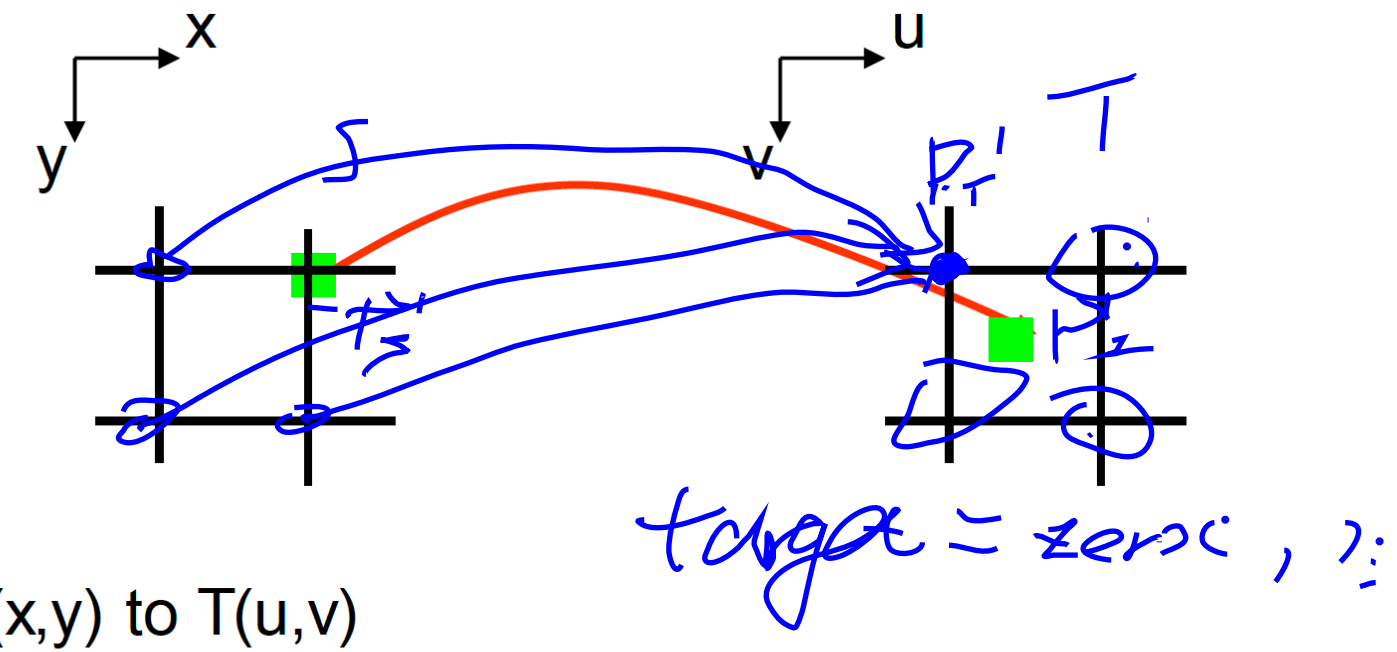
64x96

32x48

16x24

8x12

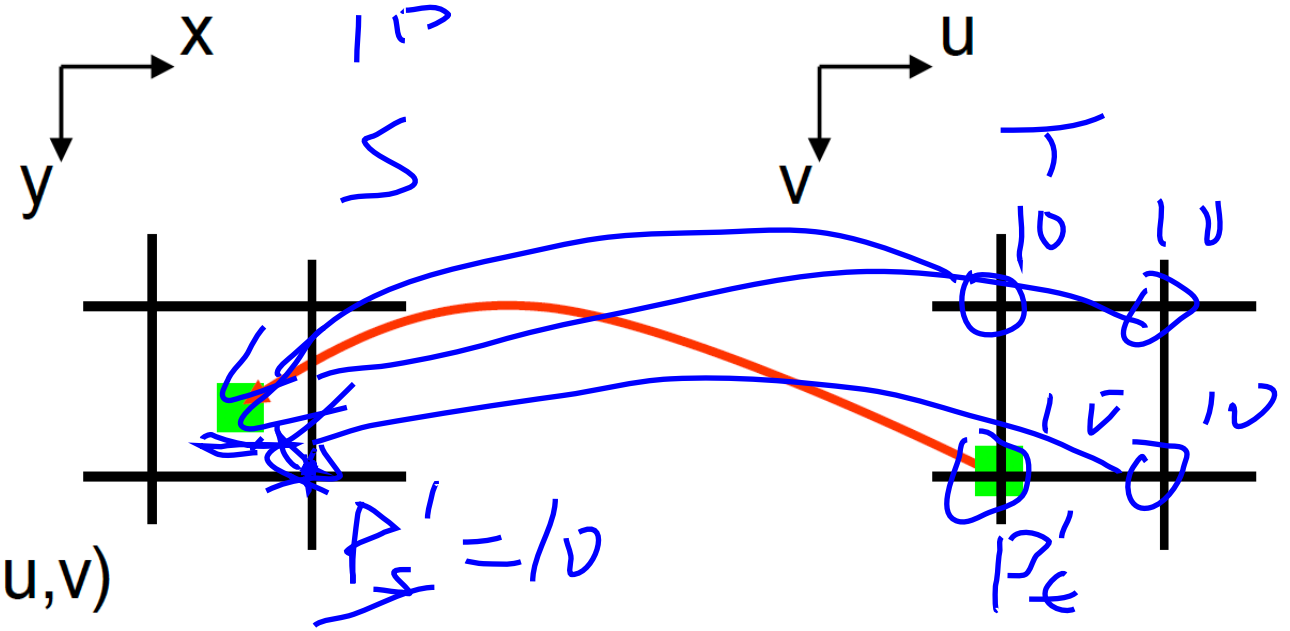
```
for y = ymin to ymax
    for x = xmin to xmax
        u = f(x,y); v = g(x,y)
        copy pixel at source S(x,y) to T(u,v)
```



Task 2 – Image Rescaling

- Forward mapping
  - Iterate over source, sending pixels to destination
  - Holes may be generated during construction since some points in the target image may not be hit.

```
for v = v_min to v_max
    for u = u_min to u_max
        x = F(u,v);  y = G(u,v)
        copy pixel at source S(x,y) to T(u,v)
```

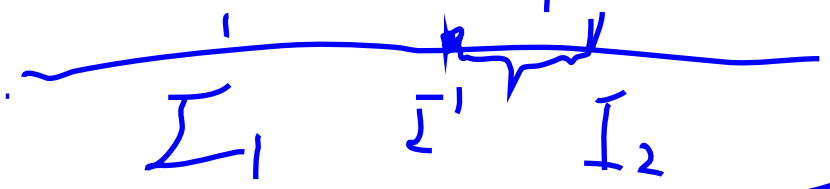


Task 2 – Image Rescaling

- Reverse mapping
  - Iterate over destination, finding pixels from source.
  - Non-integer coordinates: interpolation (Nearest, Bilinear, Bicubic…)
  - No holes, better than forward mapping

bilinear

1D. $p_1=3$ $p'=3.7$ $p_2=4$



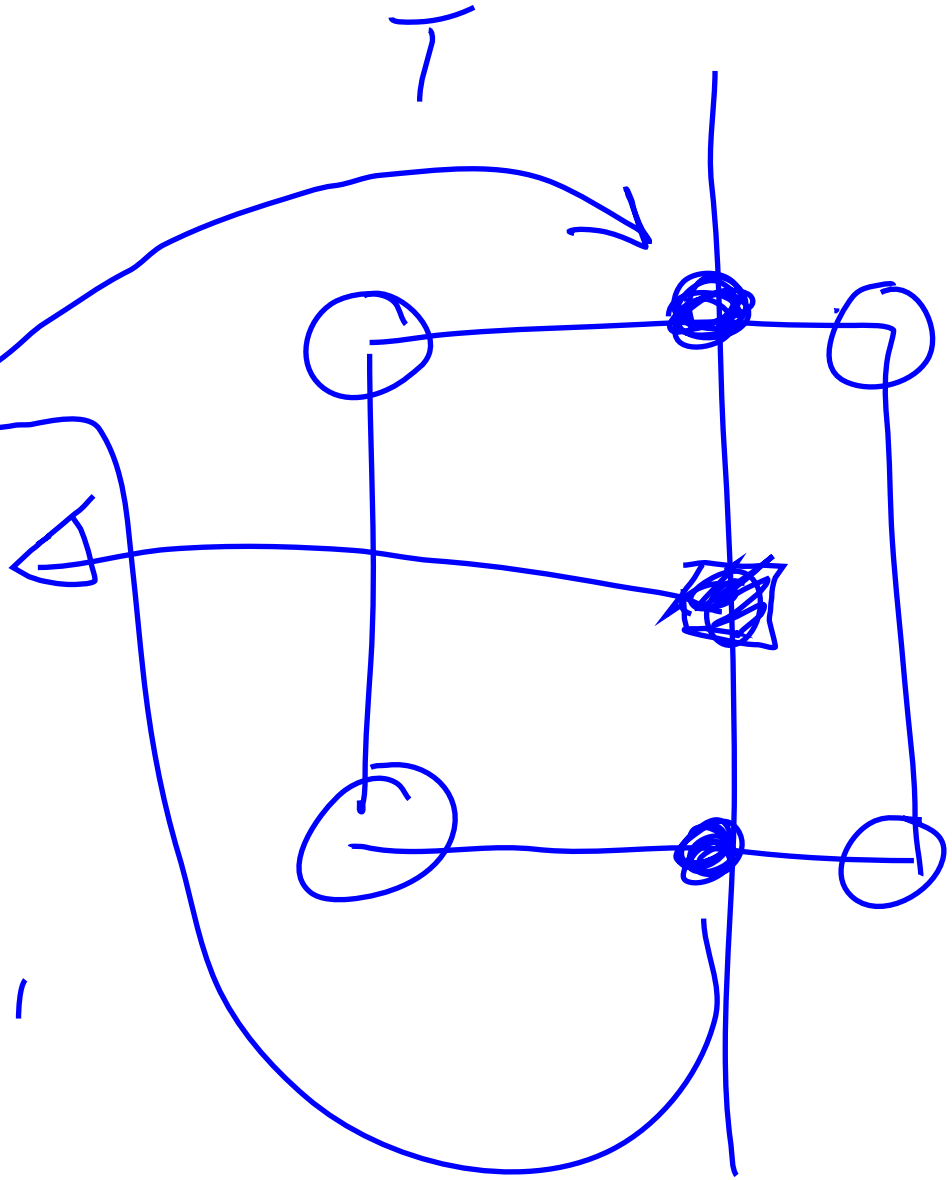
$I' = w_1 I_1 + w_2 I_2$

$1-(3.7-3) = 1-0.3 = 0.7$

$= v_1$

$1-(4-3.7) = 1-0.3 = 0.7$

$\therefore$



Task 3 – Edge Detection in the Spatial Domain

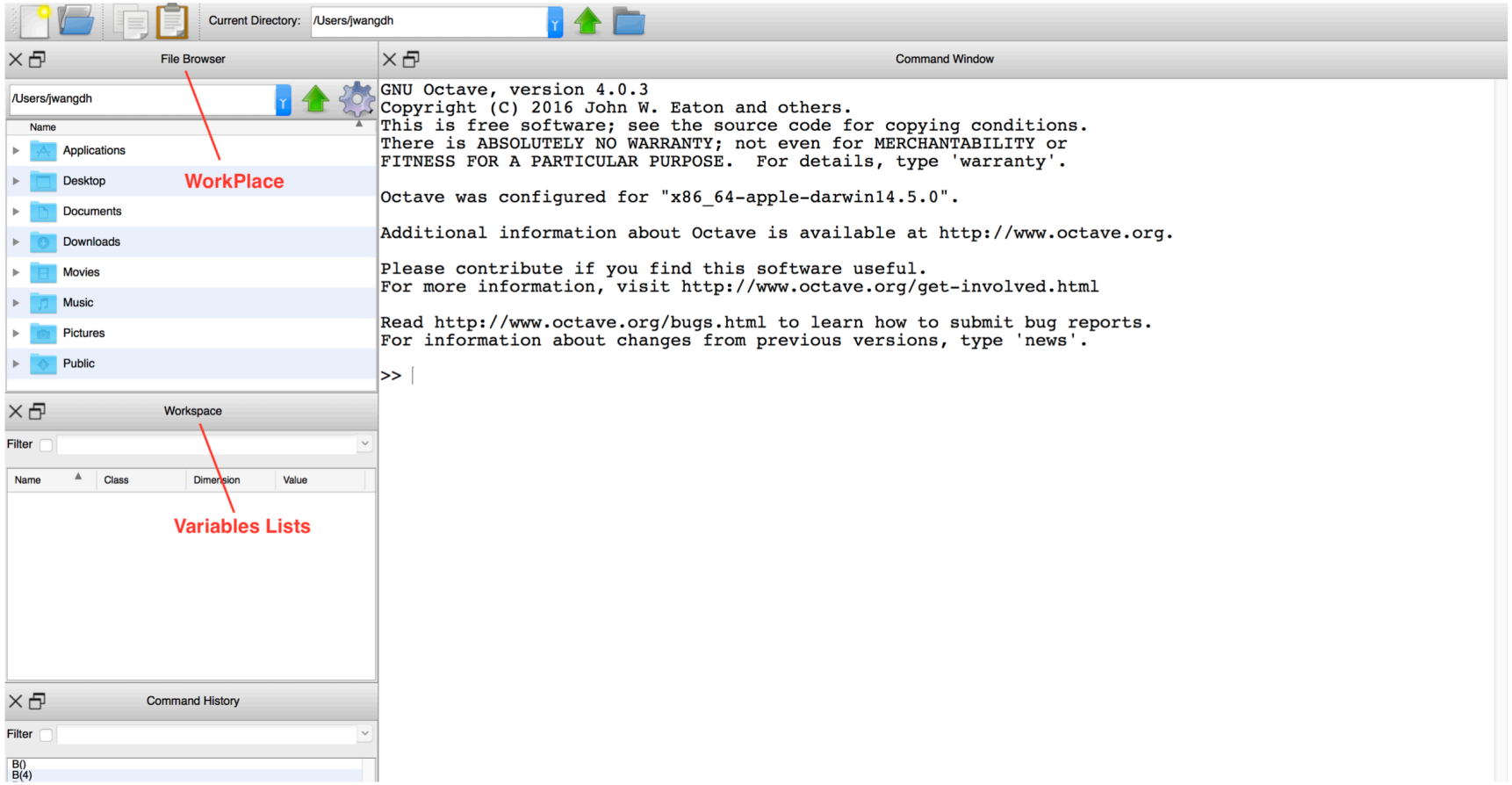
- *mysobeledge.m:* computes the binary edge image from the input image;
- *T* = Threshold for generating the binary output image. If you do not specify *T*, or if *T* is empty ([ ]), *mysobeledge(img,[],direction)* chooses the value automatically according to **Algorithm 1** described in the assignment handout.
- *direction* = A string for specifying whether to look for *'horizontal'* edges, *'vertical'* edges, positive 45 degree *'pos45'* edges, negative 45 degree *'neg45'* edges, or *'all'* edges (maximum of all responses).

Octave – Description

- 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

- 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

Image Processing in Matlab/Octave

- 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/Octave

- In conclusion
  - Read an image: imread - [0, 255], uint8
  - Processing: double ([0, 255.0]) or im2double ([0, 1])
  - Show/Write an image: imshow/imwrite
    - [0, 255], uint8
    - [0, 1], double
  - mat2gray(): rescaling to [0, 1] (double)

A : double [0, 255.0]

imshow(A)

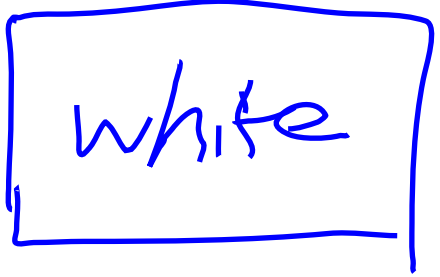


Image Processing in Matlab/Octave

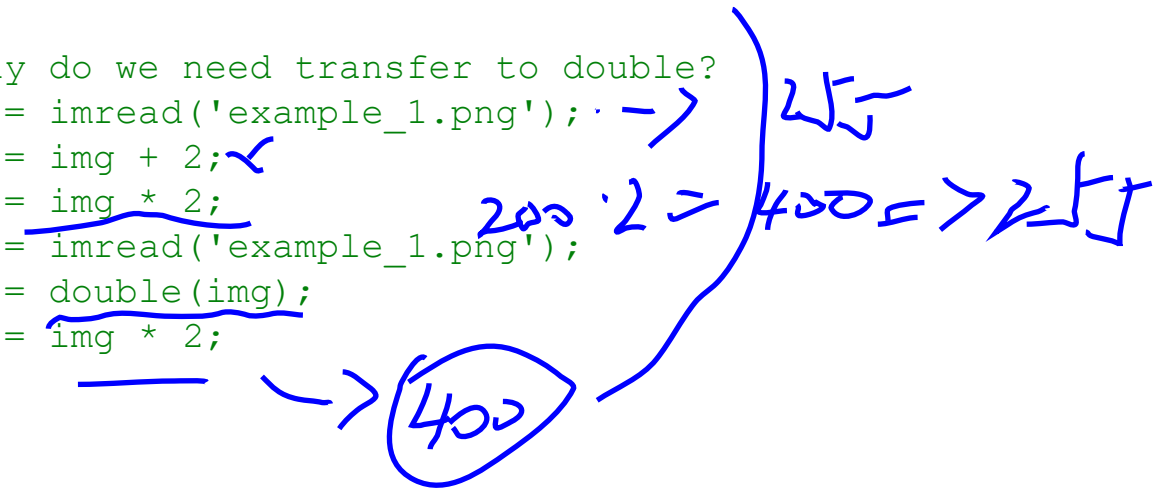
```
clc
clear


img = imread('example_1.png');

% Pay attention that the data type of matrix img is
uint8
imshow(img)

% Now we change it to double(Why?)
img = double(img);
%img = im2double(img);

%{
% Why do we need transfer to double?
img = imread('example_1.png');
img = img + 2;
img = img * 2;
img = imread('example_1.png');
img = double(img);
img = img * 2;
%}
```



```
img_inv = 255 - img;
img_log = 30 * log(1 + img);
img_pow = 0.1 * img.^1.5; % A * B ~= A .* B (*, /, ^,
...) c * img^n
img_con = (img > 100) * 255; % img > 100: logical matrix

%{
% Why do we need transfer to uint8?
img_inv = uint8(img_inv);
img_log = uint8(img_log);
img_pow = uint8(img_pow);
img_con = uint8(img_con);
%}
subplot(221),imshow(img_inv,[]),title('Inverse')
subplot(222),imshow(img_log,[]),title('Log')
subplot(223),imshow(img_pow,[]),title('Power')
subplot(224),imshow(img_con,[]),title('Contrast
Stretching')
%{
% mat2gray
img = imread('example_1.png');
img = double(img);
img = img * 2;
img = mat2gray(img);
%}
```

Vectorization

- 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';

Assignment Submission and Marking

1. Your submitted programs should be *myquantizer*, *myscaler*, and *mysobeledge,* plus all the other related M-files (plus README.txt file).

2. You must include a **README.txt file** indicating the programming software (**Octave or MATLAB**) that you are using for this assignment. By default, we will grade your assignment with Octave if the README.txt file is missing.

3. Please also attach your stitching results for different groups of images when you submit the codes. If there are more than two images in the group, then your result image should be the stitching results for **all** the images within the group.

Assignment Submission and Marking

4. You must compress all your files with the following filename format: [your 8-digit student ID]_assign1.rar (or zip), e.g. 09654321_assign1.rar, into one file.

5. If your assignment compressed file has been submitted multiple times before the due date (including late submission date), the new version will replace the old version in marking.

6. Note that we take plagiarism seriously. You are allowed to discuss or share your idea to your classmate, but you are not allowed to share your code/pseudocode of your assignment. Please also follow the referencing skills at https://libguides.ust.hk/referencing/plagiarism to avoid plagiarism.

7. Marks would be deducted if there are any violations of the above requirements.

Octave package installation

If you are using Octave, you need to install the image package.

>> pkg install -forge image

And load the image package by

>> pkg load image