Introduction to Mathematical Image Processing
Image Processing Toolbox: Part 2
(image quality, histogram, and image sharpening)

Suh-Yuh Yang (楊肅煜)

Department of Mathematics, National Central University
Jhongli District, Taoyuan City 32001, Taiwan

http://www.math.ncu.edu.tw/∼syyang/

Types of images

- **Gray-level (monochrome) image:** encoded as a 2D array, with each pixel having 8 bits; 0 means black and 255 means white; intermediate values indicate varying shades of gray.
- **Binary image:** represented as a 2D array; use 1 bit per pixel; 0 means black and 1 means white.
- **Indexed image:** a matrix of integers, each integer refers to a particular row of RGB values in a secondary matrix *(color map).*
- **RGB image:** each color pixel is represented as (R, G, B); an RGB color image corresponds to a 3D array of dimensions $M \times N \times 3$, M height, N width, and 3 is the color components.

  For RGB images of class *double,* the range of values is $[0.0, 1.0]$; for *uint8*, range is $[0, 255]$; for *uint16,* range is $[0, 65535]$.

Image quality

Image quality is defined in terms of *spatial resolution and quantization.*

- *Spatial resolution is the pixel density over the image, which is expressed qualitatively as pixels/dots per inch (ppi/dpi). The greater the spatial resolution, the more are the pixels used to display the image.*
- `imresize(x,1/2)`: halves image size, choosing even indices.

  `imresize(x,2)`: all the pixels are repeated to produce an image of the double size of the original, but with half the resolution in each direction.

```
>>A=imread('Penguins_RGB.jpg'); % size: 554 × 636 × 3
>>B=imresize(A,1/2);   % size: 277 × 318 × 3
>>C=imresize(A,2);    % size: 1108 × 1272 × 3
```

Image A 

Image B 

Image C 

Change the resolution

The following set of commands reduces the resolution of image:

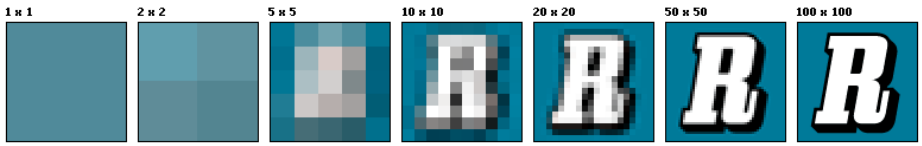
```
>>A1=imresize((imresize(A,1/2)),2);
>>A2=imresize((imresize(A,1/4)),4);
>>A3=imresize((imresize(A,1/8)),8);
```

image A1

image A2

image A3

Image A1: has half the image resolution but the same image size
Image A2: has one-fourth the image resolution but the same image size
Image A3: has one-eighth the image resolution but the same image size



Resolution illustration (wikipedia)

Image quantization

Image quantization can be described as a mapping process by which groups of data points (several pixels within a range of gray values) are mapped to a single point (a single gray level).

```
>>A=imread('Penguins_gray.jpg');
>>B=grayslice(A,64);    % reduces the quantz. levels to 64
>>imshow(B,gray(64))
>>C=grayslice(A,8);     % reduces the quantz. levels to 8
>>imshow(C,gray(8))
```

image A

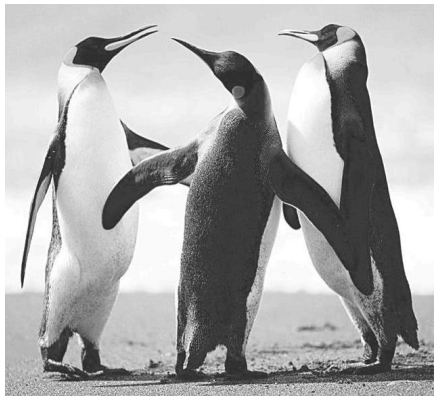


image B

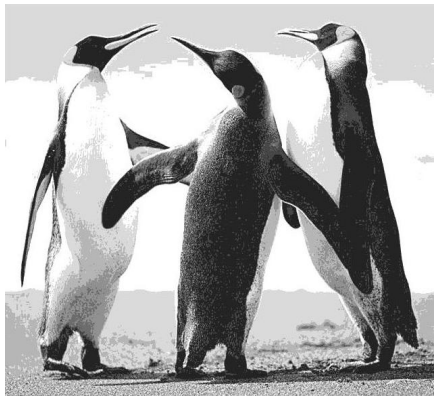
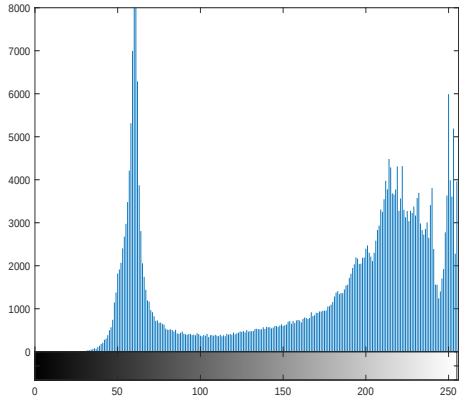


image C

Histogram (直方圖)

- Histogram of a grayscale image represents the frequency of its gray levels occurrence. *It is a graph indicating the number of times each gray level occurs in the image.*

image `A`



histogram: `imhist(A)`

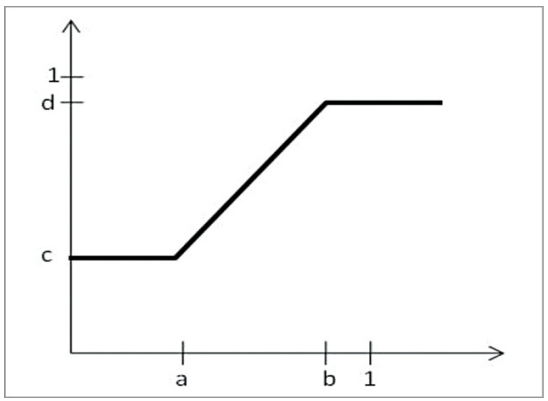
- In a well-contrasted image, gray levels (and hence the histogram) would be well spread out over much of the range.
- The three main operations performed on a histogram include *histogram stretching, histogram shrinking, and histogram sliding.*

Histogram stretching and shrinking

- **Histogram stretching:** This technique *(aka input cropping, 輸入裁剪)* consists of a linear transformation that stretches part of the original histogram so that its non-zero intensity range occupies the full dynamic grayscale.

  The following command stretches the histogram of image `A`:

  ```
  >>B=imadjust(A,[a, b],[c, d]); % a,b,c,d ∈ [0,1]
  ```



`imadjust` *function*

- **Histogram shrinking:** This technique *(aka output cropping, 輸出裁剪)* modifies the original histogram such that its dynamic grayscale range is compressed into a narrower grayscale. The `imadjust` function can be used for histogram shrinking.

Histogram stretching (continued)

```
>>A=imread('Penguins_gray.jpg');
>>B=imadjust(A,[0.2, 0.8],[0, 1]);
```

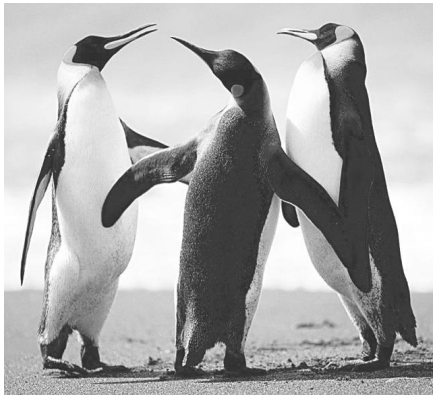


image A

adjusted image B

```
>>C=imread('Penguins_RGB.jpg');
>>D=imadjust(C,[0.2, 0.8],[0, 1]);
```

image C

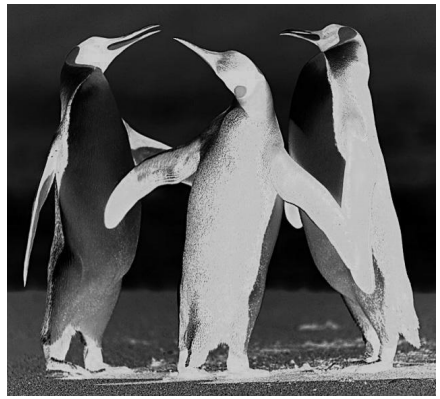
adjusted image D

Histogram stretching (continued)

`imadjust(A,[],[1,0])` inverts the gray value of the image, to produce a result similar to *photographic negative (攝影底片)*

```
>>A=imread('Penguins_gray.jpg');
>>B=imadjust(A,[],[1,0]);
```

image A



inverted image B

Histogram equalization (HE)

An alternative approach is to use *histogram equalization* (直方圖均衡化) command, which is entirely an automatic procedure.

```
>>A=imread('Penguins_gray.jpg');
>>HE=histeq(A);    % histogram equalization
>>imshow(HE)
>>imhist(HE)
```

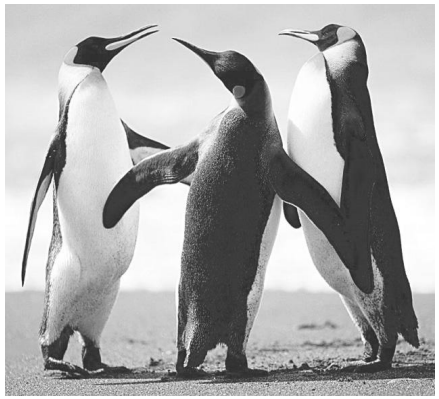
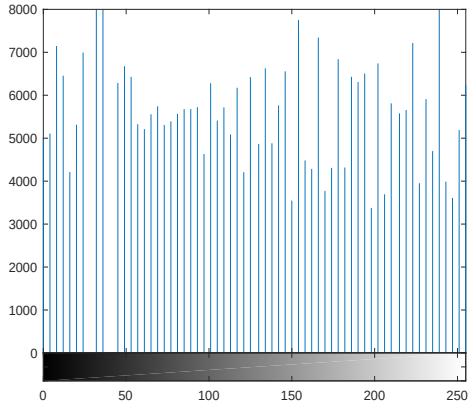


image A

image HE



`imhist(HE)`

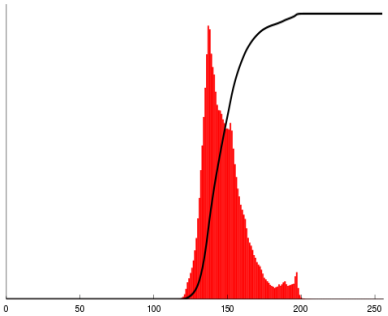
Histogram equalization

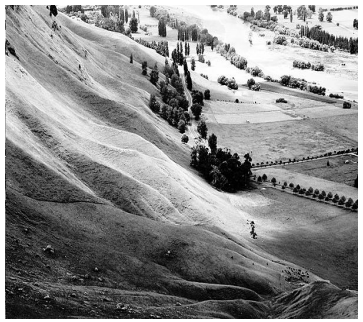
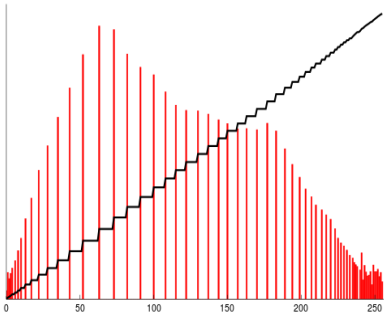
- We are given a grayscale image $f : \Omega_D \to V := \{0, 1, \cdots, 255\}$, where domain $\Omega_D := \{(1,1), (1,2), \cdots, (m,n)\}$. *The cumulative histogram (累增直方圖)* F is defined by

$$F(\eta) := \#\{\boldsymbol{x} \in \Omega_D : f(\boldsymbol{x}) \le \eta\}, \quad \forall\, \eta \in V.$$

- *The histogram equalized image* $u : \Omega_D \to V$ is defined by

$$u(\boldsymbol{x}) := 255 \times \left(\frac{1}{mn} F\big(f(\boldsymbol{x})\big)\right).$$

Histogram equalization: (left) before; (right) after; (red) corresponding histogram; (black) cumulative histogram F (quoted from Wikipedia)

Histogram sliding

Simply adding or subtracting a constant brightness value to all pixels in the image. An image with comparable contrast properties, but higher or lower average brightness.

`imadd`, `imsubtract` functions can be used for histogram sliding.

When implementing histogram sliding, you must make sure that pixel values do not go outside the grayscale boundaries.

```
>>A=imread('Penguins_gray.jpg');
>>B=im2double(A);    % entries in matrix B are in [0,1]
>>bright_add=0.2;
>>C=B+bright_add;    %C=imadd(B,0.2);
```

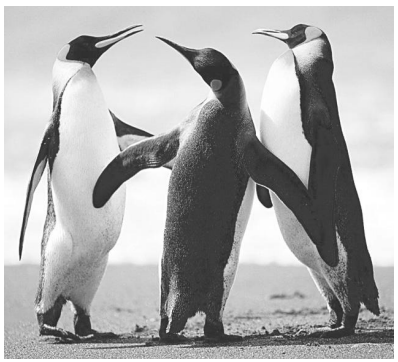
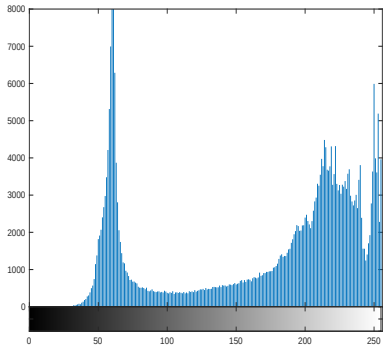
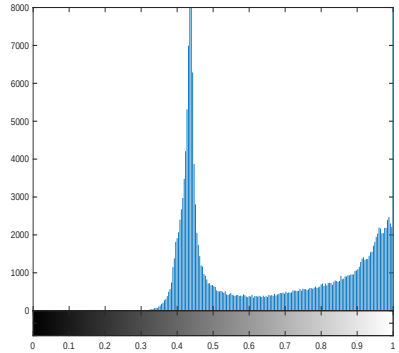


image A



`imhist(A)`

image C



`imhist(C)`

Thresholding

Thresholding is used to remove unnecessary details from an image and concentrate on essentials, also used to bring out hidden details, in case the object of interest and background have similar gray levels.

To convert an image into *black-and-white image* with threshold of 70:

```
>>A=imread('Penguins_gray.jpg');
>>B=imshow(A>70);
```

image A

image A > 70

Image sharpening

- Sharpening enhances the edges and fine details of an image for viewing by human beings. It increases the contrast between bright and dark regions to bring out image features.

Image sharpening: original and resultant (Wikipedia)

- Image sharpening is a powerful tool for emphasizing texture. The tool works by *exaggerating the brightness difference along the edges within an image.*
- Note that the sharpening process is *not able to reconstruct the ideal image, but it creates the appearance of a more pronounced edge.*

Image sharpening: MATLAB command

Most image sharpening software tools work by applying something called an *'unsharp mask,'* which actually acts to sharpen an image.

The command used for sharpening an image:

```
>>A=imread('image_original.jpg');
>>B=imsharpen(A);
>>imwrite(B,'image_resultant.jpg')
>>C=imsharpen(A,'Radius',4,'Amount',2);
% 'Threshold',0.8);
>>imwrite(C,'image_resultant2.jpg')
```

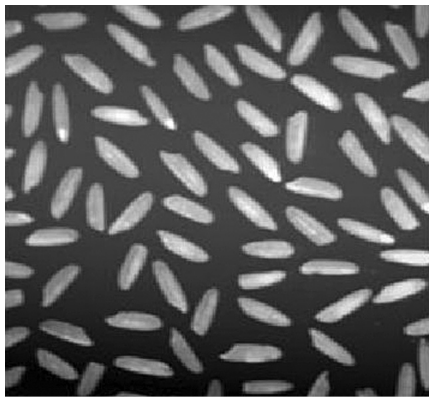
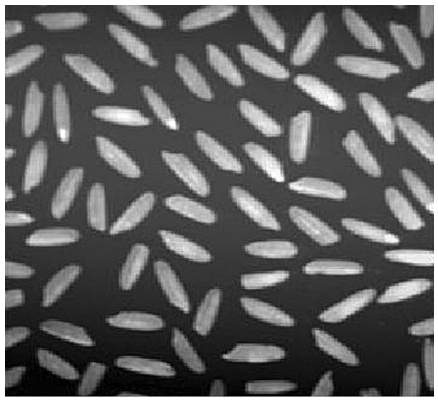
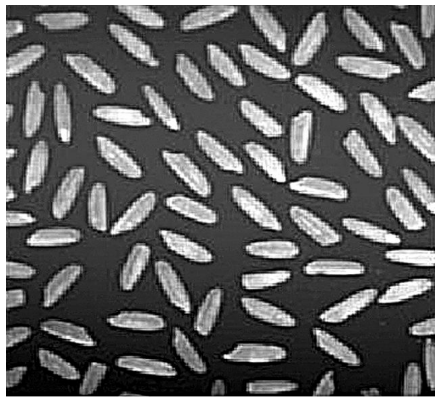




Image sharpening: original, resultant 1, resultant 2

Basically, sharpening involves application of a high-pass filter to an image.