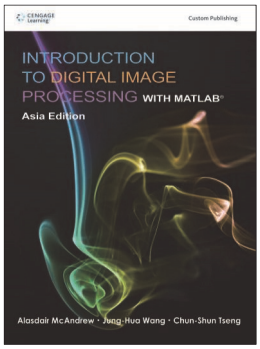


Introduction to Digital Image Processing with MATLAB® Asia Edition
McAndrew·Wang·Tseng

Chapter 1: Introduction

CENGAGE Learning

1.2 What Is Image Processing?

- **Image processing** involves changing the nature of an image in order to either
  1. improve its pictorial information for human interpretation (enhancement) → Category 1
  2. render it more suitable for autonomous machine perception (recognition) → Category 2

CENGAGE Learning

1.2 What Is Image Processing?

Category 1.

- Enhancing the edges of an image to make it appear sharper
- Note how the second image appears cleaner; it is a more pleasant image.
- Sharpening edges is a vital component of printing.

(a) (b)

FIGURE 1.1 *Image sharpening. (a) The original image. (b) Result after sharpening.*

CENGAGE Learning

1.2 What Is Image Processing?

- Removing noise from an image
- Noise in the image comes from random errors.

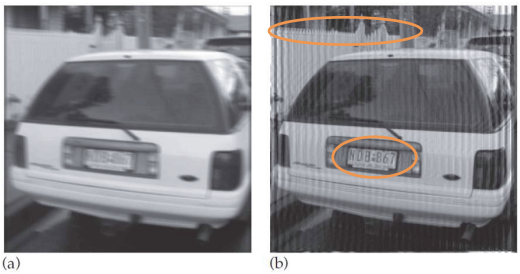
(a) (b)

FIGURE 1.2 *Removing noise from an image. (a) The original image. (b) After removing noise.*

CENGAGE Learning

1.2 What Is Image Processing?

- Removing motion blur from an image. An example is given in Figure 1.3
- In Figure 1.3(b), it is easier to read the number plate and to see the spikes on the fence behind the car, as well as other details not at all clear in the original image Figure 1.3(a).



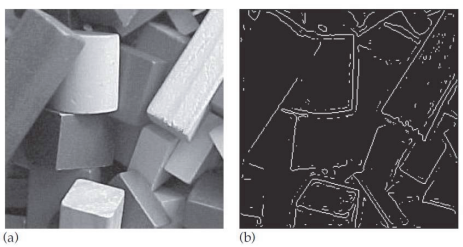
(a) (b)

FIGURE 1.3 *Image deblurring. (a) The original image. (b) After removing the blur.*

CENGAGE Learning

1.2 What Is Image Processing? Category 2

- Obtaining the edges of an image
- Once we have the edges we can measure their spread and the area contained within them.
- We can also use edge-detection algorithms as a first step in edge enhancement



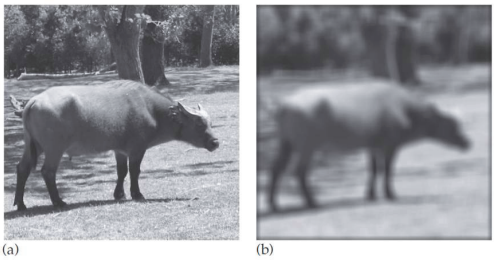
(a) (b)

FIGURE 1.4 *Finding edges in an image. (a) The original image. (b) Its edge image.*

CENGAGE Learning

1.2 What Is Image Processing?

- For measurement or counting purposes, we may not be interested in all the detail in an image (Figure 1.5).
- We could, for example, measure the size and shape of the animal without being distracted by unnecessary detail.

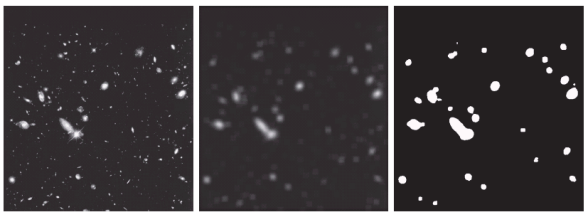


(a) (b)

FIGURE 1.5 *Blurring an image. (a) The original image. (b) Blurring to remove detail.*

CENGAGE Learning

Example of Measuring Size and Shape by Noise Reduction



a b c

FIGURE 3.36 (a) Image from the Hubble Space Telescope. (b) Image processed by a 15 × 15 averaging mask. (c) Result of thresholding (b). (Original image courtesy of NASA.)

CENGAGE Learning

1.3 Image Sampling and Acquisition

- **Sampling** refers to the process of digitizing a continuous function
  - e.g., $y = \sin(x) + \frac{1}{3}\sin(3x)$
  - ✓ sample it at 10 evenly spaced values of *x* only
  - ✓ sample it at 100 points

FIGURE 1.6 *Sampling a function—undersampling.*

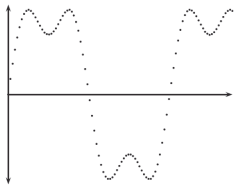


FIGURE 1.7 *Sampling a function with more points.*

CENGAGE Learning

How many samples ?

- Sampling rate : samples/sec
- Nyquist sampling theorem (= Shannon's sampling theorem)
  - gives us an optimal number of samples which also indicates optimal trade-offs between memory space and loss of information.
  - If a continuous signal is band-limited, i.e. if a signal has a maximum frequency f_{max}, we have to choose the sampling frequency $f_s > 2f_{max}$ → Nyquist rate

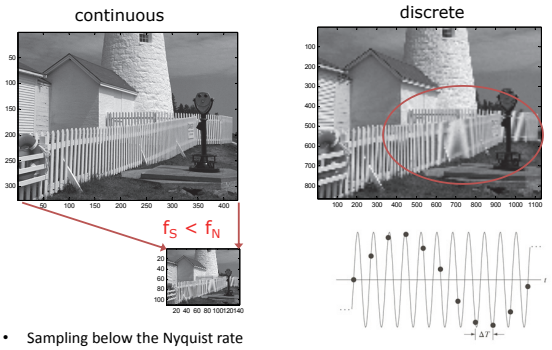
Nyquist frequency

- Examples
  - Find the Nyquist rate of the signal ?
    8 x 2 = 16
    means what? We need at least 16 Hz of sampling frequency
  - Human ears can hear the signal only within the range of 20~20KHz -> sampling rate of compact disk : 44.1KHz > 40KHz
  - Bandwidth of conventional telephone line: 0.3~3.4KHz -> sampling rate : 8KHz > 6.8 KHzo

그림 2.1-4 주파수 영역의 신호

Aliasing Example

continuous discrete



- Sampling below the Nyquist rate
  => causes distortion in restored signal.
  (Especially high frequency component)

CENGAGE Learning

Aliasing Example

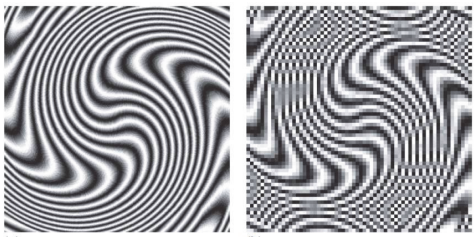
a b c

FIGURE 4.17 Illustration of aliasing on resampled images. (a) A digital image with negligible visual aliasing. (b) Result of resizing the image to 50% of its original size by pixel deletion. Aliasing is clearly visible. (c) Result of blurring the image in (a) with a 3 × 3 averaging filter prior to resizing. The image is slightly more blurred than (b), but aliasing is not longer objectionable. (Original image courtesy of the Signal Compression Laboratory, University of California, Santa Barbara.)

CENGAGE Learning

1.3 Image Sampling and Acquisition

- **Shannon's Sampling Theorem** which says, in effect, that a continuous function can be reconstructed from its samples provided that the sampling frequency is at least twice the maximum frequency(Nyquist frequency) in the function.



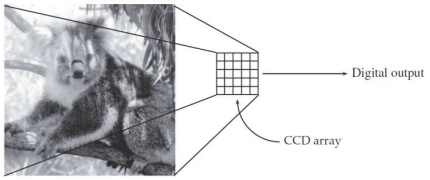
(a) (b)

FIGURE 1.8 *Effects of sampling. (a) Correct sampling; no aliasing. (b) An undersampled version with aliasing.*

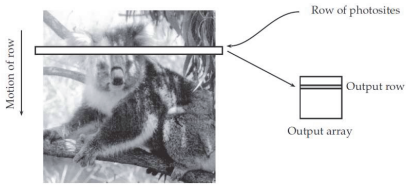
CENGAGE Learning

Image Acquisition

CCD Camera



FLAT-BED SCANNER



Original scene

FIGURE 1.10 *Capturing an image with a CCD scanner.*

CENGAGE Learning

ENERGY SOURCES FOR MEDICAL IMAGING

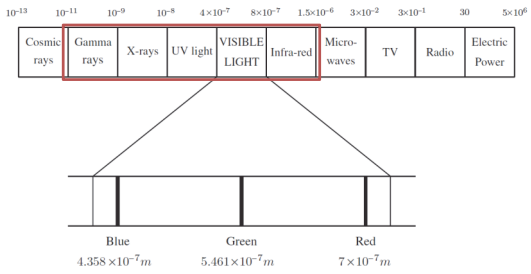


FIGURE 1.11 *The electromagnetic spectrum.*

CENGAGE Learning

Image Acquisition by X-ray Computed Tomography

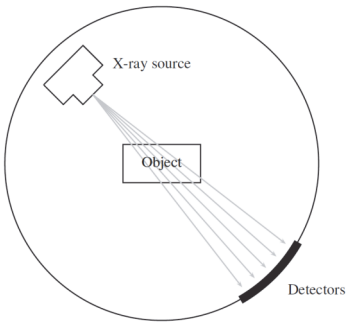


FIGURE 1.12 *X-ray tomography.*

CENGAGE Learning

1.4 Images and Digital Images

- We may consider this image as being a two-dimensional function $f(x, y)$
- We may assume that in such an image, brightness values can be any real numbers in the range 0.0 (black) to 1.0 (white)
- The $f(x, y)$ values in a digital image take only integer values ranging from 0 (black) to 255 (white)

CENGAGE Learning

Pixel Values in an Image

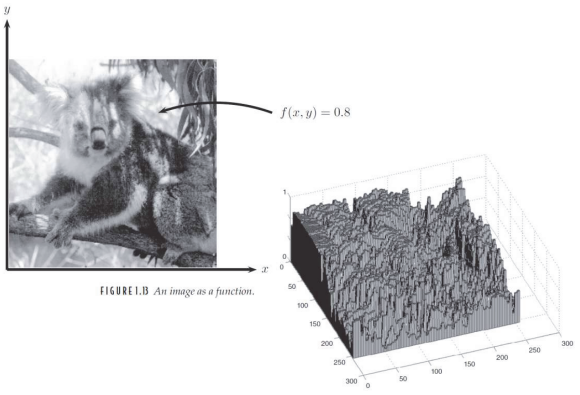


FIGURE 1.13 *An image as a function.*

FIGURE 1.14 *The image of Figure 1.13 plotted as a function of two variables.*

1.4 Images and Digital Images

- A **digital image** can be considered as a large array of sampled points from the continuous image.
- These points are the **pixels**, which constitute the digital image.
- The pixels surrounding a given pixel constitute its **neighborhood**.

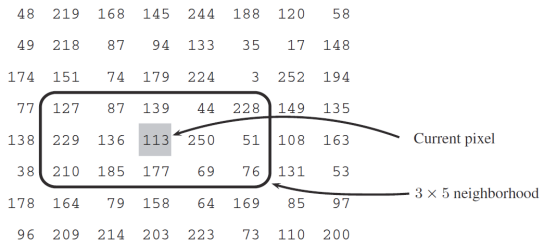


FIGURE 1.15 *Pixels with a neighborhood.*

Applications of Image Processing

- **IMAGE ENHANCEMENT: Category 1**
  - ✓ sharpening or deblurring an out-of-focus image
  - ✓ highlighting edges,
  - ✓ improving image contrast or brightening an image, and
  - ✓ removing noise
- **IMAGE RESTORATION: Category 1**
  - ✓ removing of blur caused by linear motion,
  - ✓ removal of optical distortions, and
  - ✓ removing periodic interference
- **IMAGE SEGMENTATION: Category 2**
  - ✓ finding lines, circles, or particular shapes in an image, and
  - ✓ identifying cars, trees, buildings, or roads in an aerial photograph
- A given algorithm may be used for both image enhancement or for image restoration

Type of Digital Image

- **Binary**

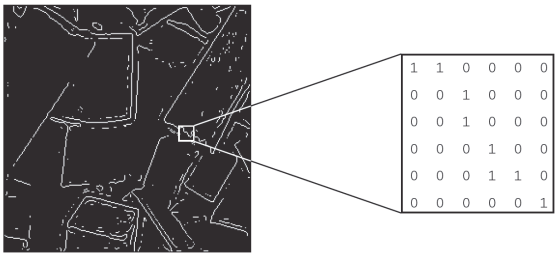


FIGURE 1.16 *A binary image.*

Type of Digital Image

- **Grayscale**

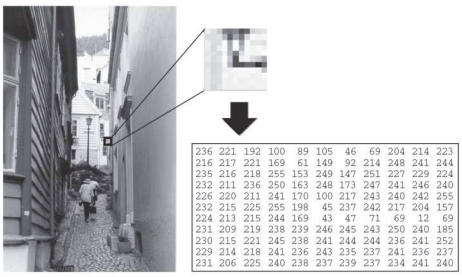


FIGURE 1.17 *A grayscale image.*

Type of Digital Image

- **True color or red-green-blue(RGB)**

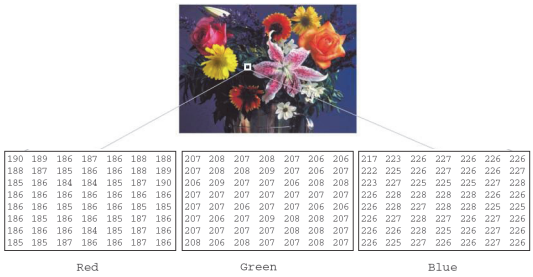


FIGURE 1.18 *A true color image.*

Type of Digital Image

- **Indexed Color**
- Most color images contain only a small subset of 16 million possible colors in RGB.
- For convenience of storage and file handling, we might need a list of all colors used in the image. => colormap

FIGURE 1.19 *An indexed color image.*

1.9 Image File Sizes

- A 512 × 512 binary image

$$512 \times 512 \times 1 = 262{,}144 \text{ bits} = 32768 \text{ bytes} = 32.768 \text{ Kb} \approx 0.033 \text{ Mb}$$

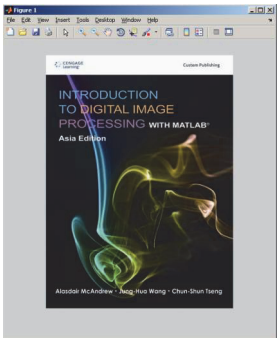
- A grayscale image

$$512 \times 512 \times 1 = 262{,}144 \text{ bytes} = 262.14 \text{ Kb} \approx 0.262 \text{ Mb}.$$

- A RGB color image

$$512 \times 512 \times 3 = 786{,}432 \text{ bytes} = 786.43 \text{ Kb} \approx 0.786 \text{ Mb}.$$

CENGAGE Learning



Chapter 2:

Images and MATLAB

CENGAGE Learning

Matlab Image Processing Toolbox

- MATLAB is a data analysis software package with powerful support for matrices and matrix operations

✓ **Command window**

```
>>
```

✓ **Reads pixel values from an image**

```
>> w=imread('wombats.tif');
```

CENGAGE Learning

How to Display Grayscale Images

```
>> figure, imshow(w)
```

`figure`

✓ Creates a figure on the screen

✓ A figure is a window in which a graphics object can be placed

`imshow(w)`

✓ displays the matrix `w` as an image

CENGAGE Learning

Pixval Command

`pixval on`

✓ turns on the pixel values in our figure.

✓ They appear at the bottom of the figure in the form.

$c, r = p$

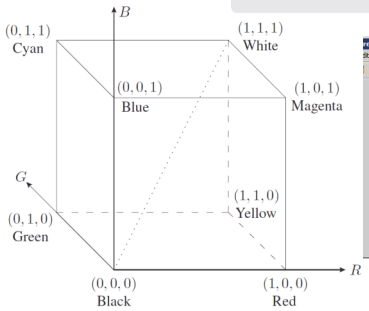
FIGURE 2.1 *The wombats image with* `pixval on`.

Note that the command `pixval on` may be removed in a later MATLAB version

CENGAGE Learning

2.2 RGB Images

```
>> a=imread('autumn.tif');
>> figure,imshow(a),pixval on
```



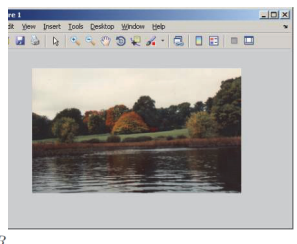


FIGURE 2.2 *The color cube for the RGB color model.*

CENGAGE Learning

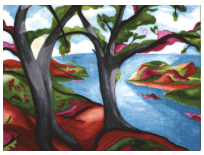
2.2 RGB Images

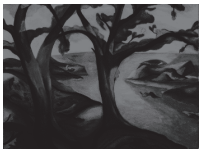
- Multidimensional array

```
>> size(a)  % 206(rows) 345(columns) 3(pages)
>> a(100,200,2) % 25
>> a(100,200,1:3) % 75  25  30
>> a(100,200,:)  % 75  25  30
>> impixel(a,200,100) % 75  25  30
```

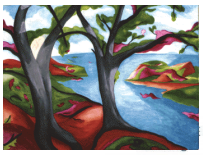

2.3 Indexed Color Images

```
>> figure, imshow('trees.tif')
>> em = imread('trees.tif');
>> figure, imshow(em)
```





```
>> [em, emap] = imread('trees.tif');
>> figure, imshow(em, emap)
```



How to Know Information on Your Image

- A great deal of information can be obtained with the **`imfinfo`** function (pp. 26)
- Useful header information
  - Filename, FileSize, Format: 'tif'
  - Width, Height, BitDepth
  - ColorType: 'indexed', 'truecolor', 'grayscale'
  - ByteOrder: 'little-endian', 'big-endian'
  - Compression: compression algorithm used for the image
  - XResolution, YResolution, ResolutionUnit: 'Inch'
  - Colormap: [256x3 double] ← 'indexed'
- imread automatically read the header of the specified image file.

2.4 Data Types and Conversions

```
>>  a=23;
>> b=uint8(a);
>> b

b =

    23

>> whos a b
  Name      Size            Bytes  Class

  a         1x1                 8  double array
  b         1x1                 1  uint8 array
```

- You can use MATLAB for image processing without ever really knowing the difference among GIF, TIFF, PNG, etc.
- However, some knowledge of the different graphics formats (data type) can be useful in order to make a reasoned decision.

2.4 Data Types and Conversions

TABLE 2.1 *Data types in MATLAB*

Data Type	Description	Range
int8	8-bit integer	−128–127
uint8	8-bit unsigned integer	0–255
int16	16-bit integer	−32768–32767
uint16	16-bit unsigned integer	0–65535
double	Double precision real number	Machine specific

TABLE 2.2 *Converting images in MATLAB*

Function	Use	Format
ind2gray	Indexed to grayscale	y=ind2gray(x,map);
gray2ind	Grayscale to indexed	[y,map]=gray2ind(x);
rgb2gray	RGB to grayscale	y=rgb2gray(x);
gray2rgb	Grayscale to RGB	y=gray2rgb(x);
rgb2ind	RGB to indexed	[y,map]=rgb2ind(x,n);
ind2rgb	Indexed to RGB	y=ind2rgb(x,map);

2.5 Images Files and Formats

- The **`imread`** and **`imwrite`** functions of MATLAB currently support the following formats.

✓ JPEG: These images are created using the Joint Photographics Experts Group compression method (Ch14).

✓ TIFF: A very general format that supports different compression methods, multiple images per file, and binary, grayscale, truecolor, and indexed images. Tagged Image File Format.

✓ GIF: A format designed for data transfer. It is still popular and well supported, but is somewhat restricted in the image types it can handle. Graphics Interchange Format.

✓ BMP: Microsoft Bitmap format has become very popular and is used by Microsoft operating systems.

✓ PNG, HDF, PCX, XWD, ICO, CUR

HEXADECIMAL DUMP

- We can see inside of image file as it is in binary.

```
function dumphex(filename, n)
%
% DUMPHEX(FILENAME,N) prints the first 16*n bytes of the file FILENAME
% in hex and ASCII.  For example:
%
%   dumphex('picture.bmp',4)
%
fid = fopen(filename, 'r');
if fid==-1
  error('File does not exist or is not in your Matlab path');
end;
a=fread(fid,16*n,'uchar');
idx=find(a>=32 & a<=126); % check if it is an ascii code
ah=dec2hex(a);
b=repmat([' '],16*n,3);
b2=repmat('.',16,n);
b2(idx)=char(a(idx));
b(:,1:2)=ah;
[reshape(b',48,n)' repmat(' ',n,2) reshape(b2,16,n)']
```

FIGURE 2.3 *A function for producing hexadecimal dumps.*

Vector versus Raster Images

- **Vector images**
  - A collection of lines or vectors
  - Can be magnified to any desired size without losing sharpness
  - Good for images consisting mostly of lines and mathematically described curves (characters, graphs)
  - PostScript by Adobe Acrobat
- **Raster images**
  - A collection of dots
  - A list of gray or color intensities of each pixel
  - Images captured by digital camera, scanner, etc.
  - The great bulk of image file formats store images as raster information.

2.5.2 A Simple Raster Format

- As well as containing all pixel information, an image file must contain some **header information**
  - ✓ This must include the size of the image, but may also include some documentation, a color map, and the compression method.
- Example: PGM format
  - designed to be a generic format used for conversion between other formats

```
P2    means ASCII PGM file
# CREATOR: The GIMP's PNM Filter Version 1.0   comments
256 256    # of cols and rows
255  # of grayscales
41 53 53 53 53 49 49 53 53 56 56 49 41 46 53 53 53
53 41 46 56 56 56 53 53 46 53 41 41 53 56 49 39 46
pixel values delimited by  space and CR
```


Header Format of Microsoft BMP

Bytes	Information	Description
0–1	Signature	BM in ASCII = 42 4D in hexadecimal.
2–5	FileSize	The size of the file in bytes.
6–9	Reserved	All zeros.
10–13	DataOffset	File offset to the raster data.
14–17	Size	Size of the information header = 40 bytes.
18–21	Width	Width of the image in pixels.
22–25	Height	Height of the image in pixels.
26–27	Planes	Number of image planes (= 1).
28–29	BitCount	Number of bits per pixel: 1: Binary images; two colors, 4: 2^4 = 16 colors (indexed), 8: 2^8 = 256 colors (indexed), 16: 16-bit RGB; 2^{16} = 65,536 colors, 24: 24-bit RGB; 2^{24} = 17,222,216 colors.

Header Format of Microsoft BMP

Bytes	Information	Description
30–33	Compression	Type of compression used: 0: No compression (most common), 1: 8-bit RLE encoding (rarely used), 2: 4-bit RLE encoding (rarely used).
34–37	ImageSize	Size of the image. If compression is 0, then this value may be 0.
38–41	HorizontalRes	The horizontal resolution in pixels per meter.
42–45	VerticalRes	The vertical resolution in pixels per meter.
46–49	ColorsUsed	The number of colors used in the image. If this value is zero, then the number of colors is the maximum obtainable with the bits per pixel, that is, 2^{BitCount}.
50–53	ImportantColors	The number of important colors in the image. If all the colors are important, then this value is set to zero.

Reading Header of Microsoft BMP

```
>> dumphex('blocksets.bmp',4)

ans =

42 4D 6E 18 00 00 00 00 00 00 36 00 00 00 28 00  BMn.......6...(.
00 00 42 00 00 00 1F 00 00 00 01 00 18 00 00 00  ..B.............
00 00 38 18 00 00 C4 0E 00 00 C4 0E 00 00 00 00  ..8.............
00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
```

1. The image width is given by bytes 18–21; they are in the second row

   42 00 00 00
2. Since BMP uses 'little-endian' for byte ordering, to find the actual width, we reorder these bytes back-to-front:

   00 00 00 42
3. Now we can convert to decimal

   $(4\times16^1)+(2\times16^0) = 66$

   which is the image width in pixels
4. The image height

   1F 00 00 00

   $(1\times16^1)+(F\times16^0) = 31$

GIF [jif]

- Proposed by Compuserve in the late 1980's as a mean for distributing images over networks.
- Colors are stored using a color map. The GIF specification allows a maximum of 256 colors per image
- GIF doesn't allow binary or grayscale images, except as can be produced with RGB values
- The pixel data is compressed using LZW (Lempel-Ziv-Welch) compression
- The GIF format allows multiple images per file. This aspect can be used to create animated GIFs

Portable Network Graphics(PNG[ping])

- The PNG format has been more recently designed to replace GIF and to overcome some of GIF's disadvantages.
- Does not rely on any patented algorithms, and it supports more image types than GIF.
- Supports grayscale, true color, and indexed images.
- Moreover, its compression utility, **zlib**, always results in genuine compression

JPEG

- The JPEG algorithm uses lossy compression, in which not all the original data can be recovered

```
>> dumphex('ngc6543a.jpg',4)

ans =

FF D8 FF E0 00 10 4A 46 49 46 00 01 01 00 00 01   ......JFIF......
00 01 00 00 FF FE 00 47 43 52 45 41 54 4F 52 3A   .......GCREATOR:
20 58 56 20 56 65 72 73 69 6F 6E 20 33 2E 30 30    XV Version 3.00
62 20 20 52 65 76 3A 20 36 2F 31 35 2F 39 34 20   b  Rev: 6/15/94
```


JPEG Header

Bytes	Information	Description
0–1	Start of image marker	Always FF D8.
2–3	Application marker	Always FF E0.
4–5	Length of segment	
6–10	JFIF\ 0	ASCII JFIF.
11–12	JFIF version	In our example above 01 01 or version 1.1.
13	Units	Values are: 0 arbitrary units; 1 pixel/inch; 2 pixels/centimeter.
14–15	Horizontal pixel density	
16–17	Vertical pixel density	
18	Thumbnail width	If this is 0, there is no thumbnail.
19	Thumbnail height	If this is 0, there is no thumbnail.

TIFF

- One of the most comprehensive(complete) image formats
- Can store multiple images per file
- Allows different compression routines and different byte orderings (none, LZW, JPEG, Huffman, run-length, etc)
- Allows binary, grayscale, truecolor or indexed images, and opacity or transparency
- An excellent format for data exchange
- Almost all journal papers accept tiff files as their standard figure format.

TIFF

Bytes	Information	Description
0–1	Byte order	Either 4D 4D: ASCII MM for big endian, or 49 49: ASCII II for little endian.
2–3	TIFF version	Always 00 2A or 2A 00 (depending on the byte order) = 42.
4–8	Image offset	Pointer to the position in the file of the data for the first image.

```
>> dumphex('newborn.tif',4)

ans =

49 49 2A 00 E0 01 01 00 32 7C 5B 2D 23 19 0E 15   II*.....2|[-#...
10 0E 0D 0F 10 0F 0E 10 11 11 0E 12 13 12 10 17   ................
10 1D 70 8E 99 A0 AE B5 BB BA C2 C6 C6 CB D3 D0   ..p.............
D2 D1 CA DB DE DE E1 E5 E6 DF E4 E9 FE EB 0B ED   ................
```

TIFF

- This particular image uses the little-endian byte ordering.
- The first image in this file (which is in fact the only image), begins at byte

```
E0 01 01 00
```

- Because this is a little-endian file, we reverse the order of the bytes: **00 01 01 E0**. This works out to 66016.

big endian ordering: E0 at the lowest address(say 1000), 00 at the highest address (then 1003)

DICOM

- DICOM (Digital Imaging and Communications in Medicine)
- Like GIF, may hold multiple image files.
- May be considered as slices or frames of a three dimensional object.
- The DICOM specification is huge and complex. Drafts have been published on the World Wide Web.
- Header contains protocol parameters of image modalities(CT, MR, PET, SPECT, X-ray etc) as well as image information itself.

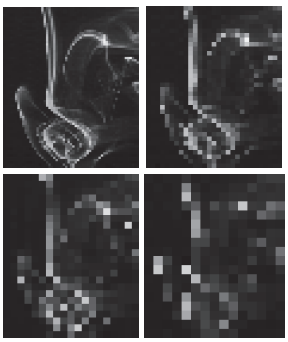
Storing an Image into a File in MATLAB

```
imwrite(X,map,'filename','fmt')
```

- Which writes the image stored in matrix X with color map **map** (if appropriate) to file **filename** with format **fmt.**
- Without the **map** argument, the image data is supposed to be grayscale or RGB.

e.g.

```
a=imread('autumn.tif');
imwrite(a,'autumn.png','png');
```

Chapter 3:

Image Display

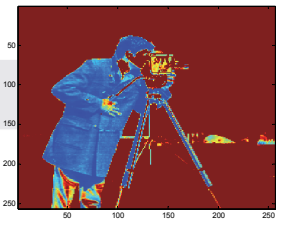
Introduction

- We take a closer look at the use of the **imshow** function.
- We look at image quality and how that may be affected by various image attributes.
- For human vision in general, the preference is for images to be sharp and detailed.
- There are many factors that will affect the display
  - ✓ ambient lighting,
  - ✓ the monitor type and settings,
  - ✓ the graphics card, and
  - ✓ monitor resolution

3.2 Basics of Image Display

- This function **image** simply displays a matrix as an indexed color image with default color map named **jet** that consists of 64 bright colors.

```
>> c=imread('cameraman.tif');
>> image(c)
```



3.2 Basics of Image Display

- To display the image properly, we need to add several extra commands to the **`image`** line

```
>> size(unique(c))

ans =

  247     1
```

```
>> image(c),truesize,axis off, colormap(gray(247))
```

By default, display 256x256 image

turns off axis labeling

adjust color map to use shades of gray only

GRAY(M) returns an M-by-3 matrix containing a gray-scale colormap. GRAY, by itself, is the same length as the current figure's colormap.

3.2 Basics of Image Display

- We may to adjust the color map to use fewer or more colors; however, this can have a dramatic effect on the result.

`colormap(gray(512))`

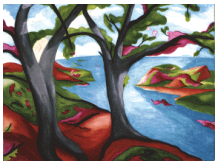
`colormap(gray(128))`

3.2 Basics of Image Display

- use **`imread`** to pick up the color map

```
>> [x,map]=imread('trees.tif');
>> image(x),truesize,axis off,colormap(map)
```

✓ **`map`** is <256x3 double> in the **workspace**



3.2 Basics of Image Display

- True color image will be read (by **`imread`**) as a three-dimensional array
  - ✓ In such a case, **`image`** will ignore the current color map and assign colors to the display based on the values in the array

```
>> t=imread('twins.tif');
>> image(t),truesize,axis off
```

3.3 The imshow Function

- We have two choices with a matrix of type **`double`**:
  - ✓ Convert to type **`uint8`** and then display
  - ✓ Display the matrix directly
- **`imshow`** will display a matrix of type **`double`** as a grayscale image (matrix elements are between 0 and 1)

```
>> c=imread('caribou.tif');
>> cd=double(c);
>> imshow(c),figure,imshow(cd)
```

imshow() assumes that min =0, max=1, then display values between 0 and 1 with the range of 0 to 255 intensity levels.

Possible Solutions ?

```
>> imshow(cd/512)
>> imshow(cd/128)
```

Capt. Budd Christman/NOAA Corps.

(a) (b)

FIGURE 3.2 *Scaling by dividing an image matrix by a scalar. (a) The matrix cd divided by 512. (b) The matrix cd divided by 128.*

3.3 The imshow Function

- We can convert the original image to **`double`** more properly using the function **`im2double`**.

```
>> cd=im2double(c);
```

im2double() adjust all the pixel values so that min pixval = 0, max pixval = 1.
double() only changes data type, not pixel values.

- If we take **`cd`** of type **`double`**, properly scaled so that all elements are between 0 and 1, we can convert it back to an image of type **`uint8`** in two ways:

```
>> c2=uint8(255*cd);
>> c3=im2uint8(cd);
```

im2uint8() adjust all the pixel values so that min pixval = 0, max pixval = 255.

3.3 The imshow Function

- **BINARY IMAGES:** MATLAB have a `logical` flag, where `uint8` values 0 and 1 can be interpreted as logical data

```
>> c1=c>120;
```

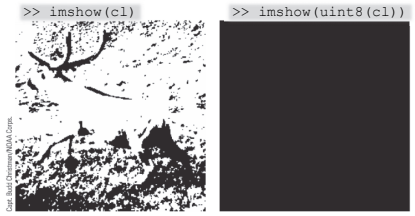
- Check `c1` with `whos`

```
Name      Size          Bytes       Class      Attributes
c1        256x256       65536       logical
```


More about imshow()

```
>> imshow(c1)
```

```
>> imshow(uint8(c1))
```



(a) (b)

FIGURE 3.3 *Making the image binary. (a) The caribou image turned binary. (b) After conversion to type* `uint8`.

uint8() only changes the type, not a pixel value.
minus value → 0, bigger than 255 → 255.

A possible panacea : **figure, imshow(c, []);**

3.4 Bit Planes

- Grayscale images can be transformed into a sequence of binary images by breaking them up into their **bitplanes**
- **The first bit plane(plane 0): LSB plane**
  - ✓ the least significant bit plane
- **The eighth bit plane(plane 7): MSB plane**
  - ✓ the most significant bit plane

3.4 Bit-Planes

- It highlights the contributions made to the total image appearance by specific bits.

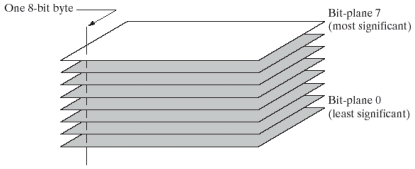


FIGURE 3.12 Bit-plane representation of an 8-bit image.

3.4 Bit Planes

- We start by making it a matrix of type `double`; this means we can perform arithmetic on the values

```
>> c=imread('cameraman.tif');
>> cd=double(c);
>> c0=mod(cd,2);
>> c1=mod(floor(cd/2),2);
>> c2=mod(floor(cd/4),2);
>> c3=mod(floor(cd/8),2);
>> c4=mod(floor(cd/16),2);
>> c5=mod(floor(cd/32),2);
>> c6=mod(floor(cd/64),2);
>> c7=mod(floor(cd/128),2);
```

FIGURE 3.4

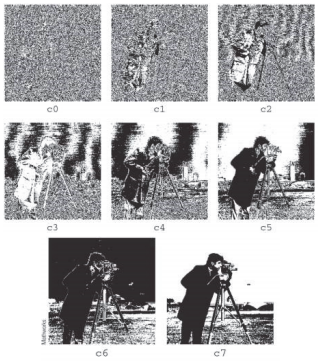


FIGURE 3.4 *The bit planes of an 8-bit grayscale image.*

CENGAGE Learning

Point Processing: Bit-Plane Slicing

thresholding
(0~127:0, 128~255:1)

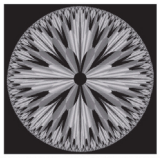


FIGURE 3.13 An 8-bit fractal image. (A fractal is an image generated from mathematical expressions). (Courtesy of Ms. Melissa D. Binde, Swarthmore College, Swarthmore, PA.)

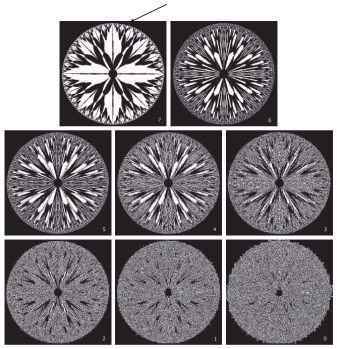


FIGURE 3.14 The eight bit planes of the image in Fig. 3.13. The number at the bottom, right of each image identifies the bit plane.

- MSB 4bits에 시각적으로 중요한 데이터 집중, LSB 4bits는 세부묘사
- 영상 압축시 유용한 정보

CENGAGE Learning

Digital Image Representation & Image Resolution

f(x,y)

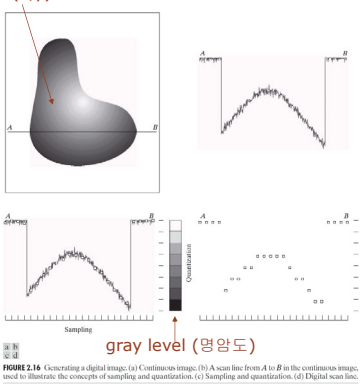


FIGURE 2.16 Generating a digital image. (a) Continuous image. (b) A scan line from *A* to *B* in the continuous image, used to illustrate the concepts of sampling and quantization. (c) Sampling and quantization. (d) Digital scan line.

f(x,y) :

f() at any point (x,y) is proportional to the brightness or grey level of the image)

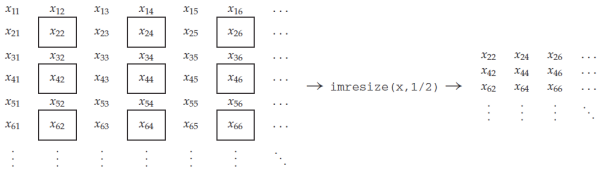
continuous -> discrete signal

- ✓ sampling (표본화)
- ✓ quantization (양자화)
- ✓ encoding (부호화)

CENGAGE Learning

3.5 Spatial Resolution

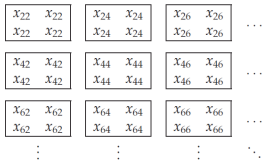
- The greater the spatial resolution, the more pixels are used to display the image.
- We can experiment with spatial resolution with MATLAB's **`imresize`** function



CENGAGE Learning

3.5 Spatial Resolution

```
x2=imresize(imresize(x,1/2),2);
```



Command	Effective resolution
`imresize(imresize(x,1/4),4);`	64 × 64
`imresize(imresize(x,1/8),8);`	32 × 32
`imresize(imresize(x,1/16),16);`	16 × 16
`imresize(imresize(x,1/32),32);`	8 × 8

```
imresize(imresize(x, factor, 'nearest'), factor, 'nearest');
```

CENGAGE Learning

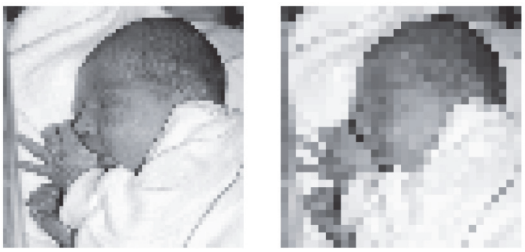
FIGURE 3.5

(a) (b)

FIGURE 3.5 *Reducing resolution of an image. (a) The original image. (b) Image at 128 × 128 resolution.*

CENGAGE Learning

FIGURE 3.6

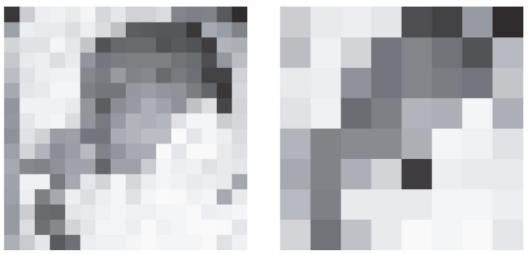


(a) (b)

FIGURE 3.6 *Further reducing the resolution of an image. (a) Image at 64 × 64 resolution. (b) Image at 32 × 32 resolution.*

CENGAGE Learning

FIGURE 3.7



(a) (b)

FIGURE 3.7 *Even more reducing the resolution of an image. (a) Image at 16 × 16 resolution. (b) Image at 8 × 8 resolution.*

CENGAGE Learning

3.6 Quantization

- **Uniform quantization**

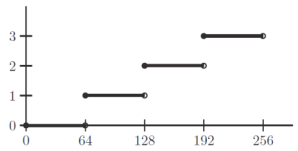


FIGURE 3.8 *A mapping for uniform quantization.*

Original values	Output value
0–63	0
64–127	1
128–191	2
192–255	3

CENGAGE Learning

3.6 Quantization

- To perform such a mapping in MATLAB, we can perform the following operations, supposing **`x`** to be a matrix of type **`uint8`**

```
f=floor(double(x)/64);
q=uint8(f*64);
```

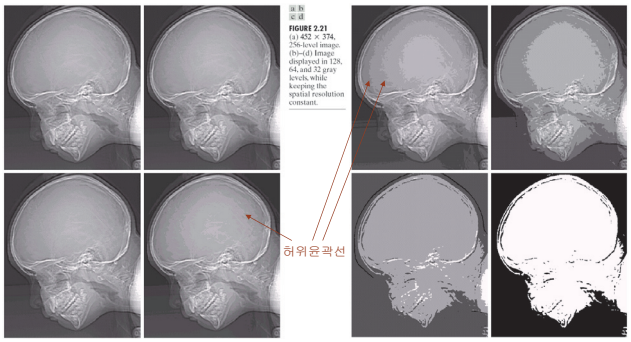
- There is, a more elegant method of reducing the grayscales in an image, and it involves using the **`grayslice`** function

Command	Number of grayscales
`imshow(grayslice(x,128),gray(128))`	128
`imshow(grayslice(x,64),gray(64))`	64
`imshow(grayslice(x,32),gray(32))`	32
`imshow(grayslice(x,16),gray(16))`	16
`imshow(grayslice(x,8),gray(8))`	8
`imshow(grayslice(x,4),gray(4))`	4
`imshow(grayslice(x,2),gray(2))`	2

GRAYSLICE creates indexed image from intensity image by thresholding.

CENGAGE Learning

Quantization & Gray-level Resolution



- reduced gray level from 256 to 128,64,32,16,8,4,2
- 452x374 fixed spatial resolution
- best gray-level resolution in this example *can be* 64 => case by case

CENGAGE Learning

3.6 Dithering

- In general terms, it refers to the process of reducing the number of colors in an image.
- Representing an image with only two tones is also known as **halftoning**.

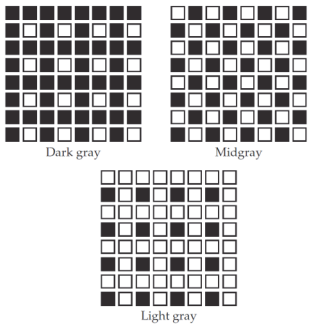


FIGURE 3.B *Patterns for dithering output.*

CENGAGE Learning

Dithering

- A Standard dithering matrix

$$D = \begin{bmatrix} 0 & 128 \\ 192 & 64 \end{bmatrix} \qquad D_2 = \begin{bmatrix} 0 & 128 & 32 & 160 \\ 192 & 64 & 224 & 96 \\ 48 & 176 & 16 & 144 \\ 240 & 112 & 208 & 80 \end{bmatrix}$$

- Suppose $d(i, j)$ is the matrix obtained by replicating the dithering matrix, then an output pixel $p(i, j)$ is defined by

$$p(i,j) = \begin{cases} 1 & \text{if } x(i,j) > d(i,j) \\ 0 & \text{if } x(i,j) \le d(i,j) \end{cases}$$

CENGAGE Learning

Dithering

- D or D2 is repeated until it is as big as the image matrix, when the two are compared.

$$D = \begin{bmatrix} 0 & 128 \\ 192 & 64 \end{bmatrix}$$

$$D_2 = \begin{bmatrix} 0 & 128 & 32 & 160 \\ 192 & 64 & 224 & 96 \\ 48 & 176 & 16 & 144 \\ 240 & 112 & 208 & 80 \end{bmatrix}$$

```
>> D=[0 128;192 64]
>> r=repmat(D,128,128);
>> x2=x>r;imshow(x2)
>> D2=[0 128 32 160;192 64 224 96;48 176 16 144;240 112 208 80];
>> r2=repmat(D2,64,64);
>> x4=x>r2;imshow(x4)
```

CENGAGE Learning

Dithering: Four Output Levels

- Dithering can be extended easily to more than two output gray values.
- For example, we wish to quantize to four output levels 0, 1, 2, and 3.

$$q(i,j) = [x(i,j)/85] \quad \text{(Since 255/3 = 85)}$$

$$p(i,j) = q(i,j) + \begin{cases} 1 & \text{if } x(i,j) - 85q(i,j) > d(i,j) \\ 0 & \text{if } x(i,j) - 85q(i,j) \le d(i,j) \end{cases}$$

CENGAGE Learning

Dithering : 4- and 8-Levels Grayscale

```
>> D = [0 56; 84 28];
>> r = repmat(D, 128, 128);
>> x = double(x);
>> q = floor(x/85);
>> x4 = q+(x>85*q>r);
>> imshow(uint8(85*x4))
```

```
>> D = [0 24; 36 12];
>> r = repmat(D, 128, 128);
>> x = double(x);
>> q = floor(x/37);
>> x8 = q+(x>37*q>r);
>> imshow(uint8(37*x8))
```

(a)

(b)

FIGURE 3.15 *Dithering to more than two grayscales. (a) Dithering to four output grayscales. (b) Dithering to eight output grayscales.*

CENGAGE Learning

Error Diffusion

- ✓ The image is quantized at two levels.
- ✓ For each pixel we take into account the error between its gray value and its quantized value.
- ✓ The idea is to spread this error over neighboring pixels.

CENGAGE Learning

Dithering: Floyd and Steinberg method

- ✓ Pixel values around 128 has big quantization error !
- ✓ For each pixel $p(i, j)$ in the image we perform the following sequence of steps:
  1. Perform the quantization
  2. Calculate the quantization error

$$E = \begin{cases} p(i,j) & \text{if } p(i,j) < 128 \\ p(i,j) - 255 & \text{if } p(i,j) >= 128 \end{cases}$$

  3. Spread this error E over pixels to the right and below according to this table.

		$+\frac{7}{16}E$
$+\frac{3}{16}E$	$+\frac{5}{16}E$	$+\frac{1}{16}E$

New p(i,j) = E·old(p(i,j)) : changes input pixels p(i,j) before they are used.

CENGAGE Learning

FIGURE 3.16

```
function y = fl_stein(x)
%
% FL_STEIN applies Floyd-Steinberg error diffusion to an image x, which is
% assumed to be of type uint8.
%
height=size(x,1);
width=size(x,2);
y=uint8(zeros(height,width));
z=zeros(height+2,width+2);
z(2:height+1,2:width+1)=x;
for i=2:height+1,
  for j=2:width+1,
    if z(i,j) < 128
      y(i-1,j-1) = 0;
      e = z(i,j);
    else
      y(i-1,j-1) = 255;
      e = z(i,j)-255;
    end
    z(i,j+1)=z(i,j+1)+7*e/16;
    z(i+1,j-1)=z(i+1,j-1)+3*e/16;
    z(i+1,j)=z(i+1,j)+5*e/16;
    z(i+1,j+1)=z(i+1,j+1)+e/16;
  end
end
```

FIGURE 3.16 *A MATLAB function for applying Floyd-Steinberg error diffusion to a grayscale image.*

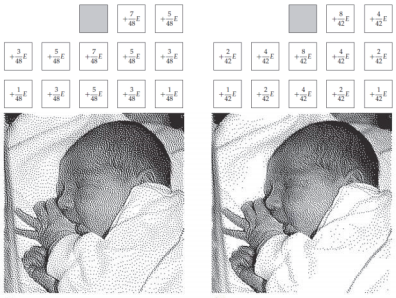
CENGAGE Learning

FIGURE 3.17

FIGURE 3.17 *The newborn baby image after Floyd-Steinberg error diffusion.*

CENGAGE Learning

FIGURE 3.18



(a) (b)

FIGURE 3.18 *Using other error-diffusion schemes. (a) Result of Jarvis-Judice-Ninke error diffusion. (b) Result of Stucki error diffusion.*

CENGAGE Learning

Summary

- **Chap 1. Introduction**
  - ✓ Sampling, image pixels, image file size
- **Chap 2. Image and Matlab**
  - ✓ Gray scale image, color images(RGB, indexed)
  - ✓ Image file formats(GIF, TIFF, BMP, JPEG)
  - ✓ Image header
- **Chap 3. Image Display**
  - ✓ Bit planes
  - ✓ Quantization
  - ✓ Dithering
  - ✓ Error diffusion

CENGAGE Learning