Image Processing

Pengwei Hao
p.hao@qmul.ac.uk

Topic 2: Sampling and Quantisation
ECS605U / ECS776P
School of EECS
Queen Mary University of London

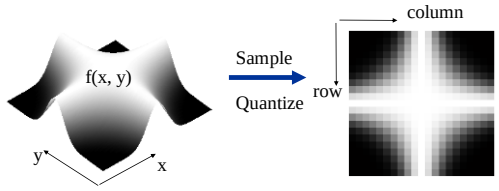
Questions

- What colour is the wall?
- What is on my desk?
- How many people are in this room?

Results: easy for humans,
difficult for computers

How do computers do?

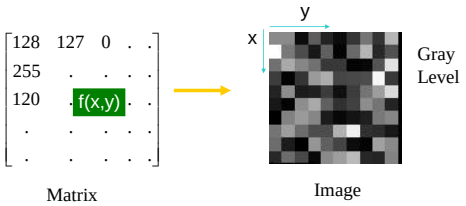
Image



2D Signal

Image

Digital Image



Matrix

Image

What are Signals?

- Signal: a function carrying information
- Examples:
  - Audio
  - Radio/Television
  - Images

Types of Signals: Dimensions

- Temporal signal: function of time
  - $f(t)$: voice, music, nerve impulses, radar
- Spatial signal: function of two (or three) spatial dimensions
  - $f(x, y)$: images (grayscale, color, multi-spectral)
  - $f(x, y, z)$: medical scans (CT, MRI, PET)
- Spatial-temporal signal: 2/3-D space, 1-D time
  - $f(x, y, t)$: video/movies

Why Signals?

- Communications:
  - Modems/Networks/Wireless
  - Audio/Video
- Images:
  - Enhancement
  - Restoration
  - Storage/Retrieval/Searching
  - Manipulation/Visualization
  - Analysis

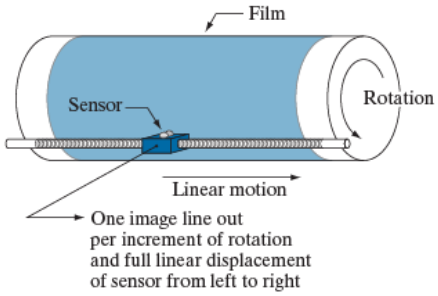
Why Digital?

- Perfect for storage, transmission, and reproduction
- Easier to manipulate (than analog):
  - Analog signals manipulated by circuits
  - Digital signals manipulated by computer
- Possible today :
  - Memory is cheap
  - Disk storage is plentiful and costs low
  - Bandwidth is increasing

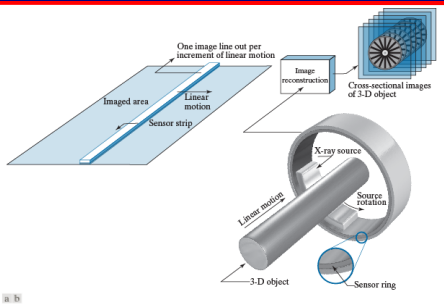
Sensors

- What is measured for images?
- Visual – intensity: Luminance of object in the scene
- Thermal – temperature: infra red
- X-rays – absorption characteristics
- Ultrasonic scanning
- Laser scanners: 3-D images
- Radars
- ……

Using a Single Sensor



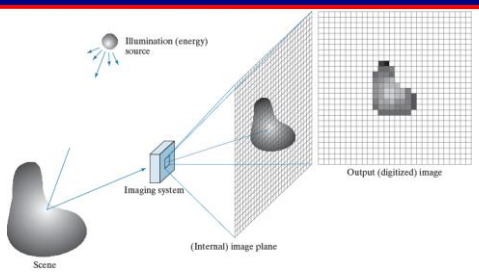
Using Sensor Strips



a b

FIGURE 2.14 (a) Image acquisition using a linear sensor strip. (b) Image acquisition using a circular sensor strip.

Using Sensor Arrays



a b c d e

FIGURE 2.15 An example of digital image acquisition. (a) Illumination (energy) source. (b) A scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.

Assessing Sensor Quality

- Resolution
- Uniformity of grid
- 2-D or 3-D images
- Indirect measurement
- Noise effects
- Active vs Passive

Applications in Business/Industry

- Scanning
- Re-use
- Multiple locations
- Security
- Fax
- Robots
  - Industry, defense, consumer, environment
- Medical
  - Patient screening and monitoring, treatment planning

Applications in Remote Sensing

- Remote sensing via satellite
  - Agriculture monitoring
  - Land use
  - Weather
  - Flood and fire control
  - Defense intelligence
  - Environment monitoring

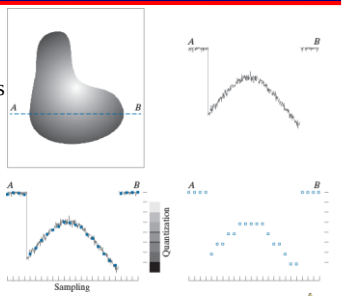
Monochrome Image

- 2D light intensity function : $f(x, y)$
  - (x, y) : spatial coordinates
  - $f(x, y)$: gray level l at (x, y)
  - $l = f(x, y) = i(x, y)\, r(x, y)$, $0<i(x,y)<\infty$, $0<r(x,y)<1$

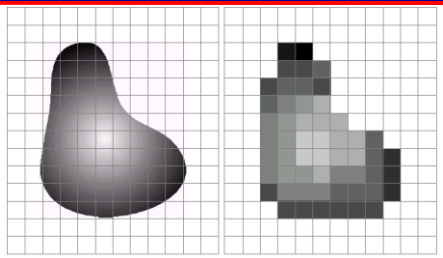
    illumination, reflectance
  - $L_{\min} \le l \le L_{\max} \Rightarrow 0 \le l \le L-1$

Basic Concepts

- Sampling
  - Digitizing the coordinate values
- Quantization
  - Digitizing the amplitude values



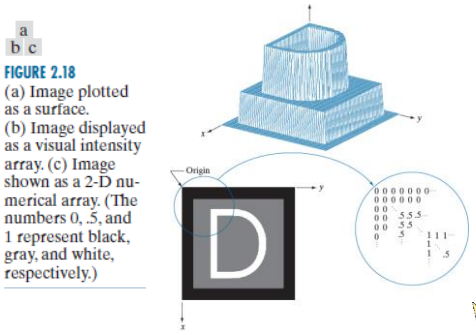
Sampling and Quantization



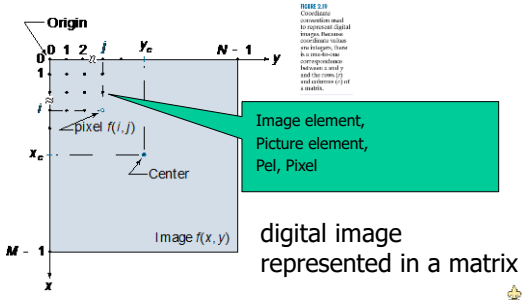
a b

FIGURE 2.17 (a) Continuos image projected onto a sensor array. (b) Result of image sampling and quantization.

Digital Image Representation



Representing Digital Images



digital image represented in a matrix

In Matrix Form

- A $M \times N$ digital image is often represented as a matrix as

$$\mathbf{A} = f(x,y) = \begin{bmatrix} f(0,0) = a_{0,0} & f(0,1) = a_{0,1} & \cdots & f(0,N-1) = a_{0,N-1} \\ f(1,0) = a_{1,0} & f(1,1) = a_{1,1} & \cdots & f(1,N-1) = a_{1,N-1} \\ \cdots & \cdots & \cdots & \cdots \\ f(M-1,0) = a_{M-1,0} & f(M-1,1) = a_{M-1,1} & \cdots & f(M-1,N-1) = a_{M-1,N-1} \end{bmatrix}$$

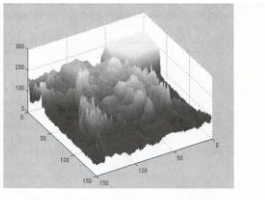
1st row, 2nd row, Mth row

1st column, 2nd column, Nth column

- Each entry of a 2D image is called a picture element, "pixel"
- In 3D case, each entry is called a volume element, "voxel"

Example of Digital Image

Usual representation



Unusual image representation

Spatial Resolution

- Smallest **discernible** detail in an image
- Determined by the sampling rate (number of samples taken per unit distance, e.g. dpi=dots per inch)
- Equivalently determined by the physical size of a pixel

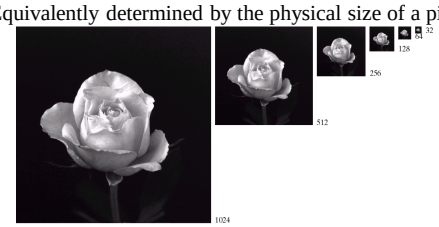
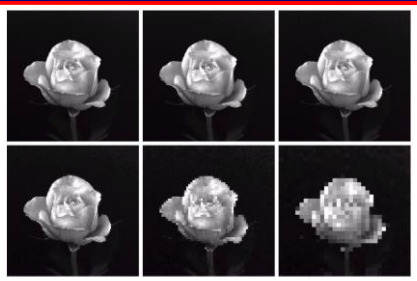


FIGURE 2.19 A 1024 × 1024, 8-bit image subsampled down to size 32 × 32 pixels. The number of allowable gray levels was kept at 256.

Spatial Resolution

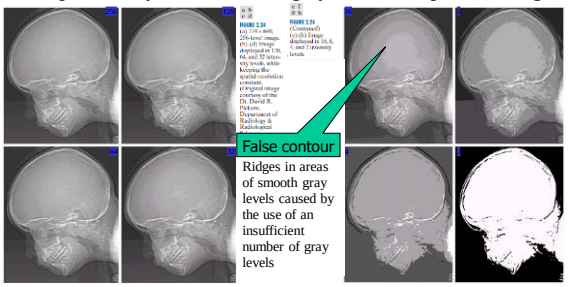


a b c
d e f

FIGURE 2.20 (a) 1024 × 1024, 8-bit image. (b) 512 × 512 image resampled into 1024 × 1024 pixels by row and column duplication. (c) through (f) 256 × 256, 128 × 128, 64 × 64, and 32 × 32 images resampled into 1024 × 1024 pixels.

Gray-Level Resolution

- Refers to the smallest discernible change in gray level
- Specified by the number of gray levels to represent image



Storage Size of Images

- Both spatial and gray level resolutions determine the storage size of an image (bytes)

  e.g. scene area: 4 km^2

  spatial resolution: 20 lines/km (400 pixels/km^2)

  gray level resolution: 64 ($\log_2 64 = 6$ bits/pixel)

  – The number of pixels:

  4 x 20 x 20 = 1600 pixels

  – The storage size (no compression, no overhead):

  1600 x 6 = 9600 bits = 1200 bytes ≈ 1.17 KB

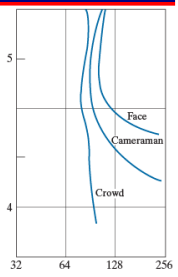
Level of Image Detail

Is a beauty still a beauty under microscope?

At what level of detail is enough?

Isopreference Curves

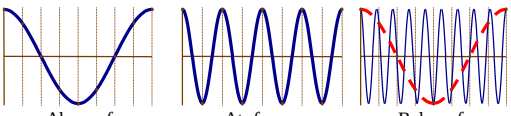
- Points lying on an isopreference curve correspond to images of equal subjective quality.
- Isoprefernce curves tend to become vertical as the detail in the image increases.



k bits, size NxN

Sampling Theorem

- If the function is sampled at a rate equal to or greater than twice its highest frequency f_{max} (Nyquist frequency, $f_N = 2\,f_{max}$), it is possible to recover completely the original function from its samples.

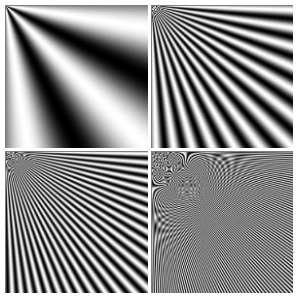


Above f_N At f_N Below f_N

Aliasing

- A phenomenon of corrupting the sampled function, if the function is *undersampled*.
- The corruption is in the form of additional frequency components being introduced into the sampled function.
- The additional frequencies are called *aliased frequencies*.
- The effect of aliased frequencies can be seen in the form of *Moire pattern*.

Moire Pattern



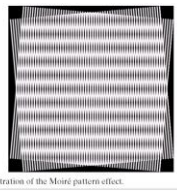


Illustration of the Moiré pattern effect.

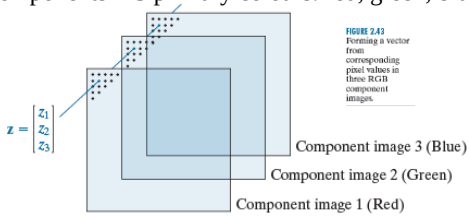
Preventing Aliasing

You have two choices:

1. Increase the sampling rate
2. Decrease the highest frequency in the signal *before sampling*

Colour Images

- 3 channels for 3 components
- 3 components = 3 primary colours: red, green, blue



red = green = blue : gray-scale/monochromatic image

Colour Images

- 3 primary colours: red, green, blue

Lab Exercises

- http://www.eecs.qmul.ac.uk/~phao/IP/Labs/
- From Week 2 to Week 9, do Lab 1 to Lab 8
- Week 10 is for coursework completion
- Deadline is at the end of Week 10, midnight of the Friday
- Assessment is done during the last 2 lab sessions, Week 11&12
- Lab 1: Download the template code: (Demo.java) and complete the GUI (menu, etc.)

image.getRGB

```
private static int[][][] convertToArray(BufferedImage image){
 int width = image.getWidth();
 int height = image.getHeight();
 int[][][] result = new int[width][height][4];
 for (int y = 0; y < height; y++) {
    for (int x = 0; x < width; x++) {
       int p = image.getRGB(x,y);
       int a = (p>>24)&0xff;  // Alpha
       int r = (p>>16)&0xff;  // Red
       int g = (p>>8)&0xff;   // Green
       int b = p&0xff;        // Blue
       result[x][y][0]=a;
       result[x][y][1]=r;
       result[x][y][2]=g;
       result[x][y][3]=b;
    }
 }
 return result;
}
```

Alpha = 255 : default value. the colour is completely opaque, visible.
Alpha = 0 : the colour is completely transparent, invisible

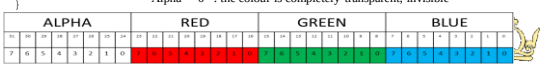


image.setRGB

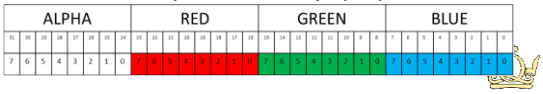
```
public BufferedImage convertToBimage(int[][][] TmpArray){
 int width = TmpArray.length;
 int height = TmpArray[0].length;
 BufferedImage
  tmpimg=new BufferedImage(width,height,BufferedImage.TYPE_INT_RGB);
 for(int y=0; y<height; y++){
   for(int x =0; x<width; x++){
      int a = TmpArray[x][y][0];  // Alpha
      int r = TmpArray[x][y][1];  // Red
      int g = TmpArray[x][y][2];  // Green
      int b = TmpArray[x][y][3];  // Blue
      int p = (a<<24) | (r<<16) | (g<<8) | b;
      tmpimg.setRGB(x, y, p);
   }
 }
 return tmpimg;
}
```

Alpha = 255 : default value. the colour is completely opaque, visible.
Alpha = 0 : the colour is completely transparent, invisible

Example: image negative

```
public BufferedImage ImageNegative(BufferedImage timg){
 int width = timg.getWidth();
 int height = timg.getHeight();
 int[][][] ImageArray = convertToArray(timg); //Convert image to array
  // Image Negative Operation:
 for(int y=0; y<height; y++){
    for(int x =0; x<width; x++){
       ImageArray[x][y][1] = 255-ImageArray[x][y][1];  //r
       ImageArray[x][y][2] = 255-ImageArray[x][y][2];  //g
       ImageArray[x][y][3] = 255-ImageArray[x][y][3];  //b
    }
 }
 return convertToBimage(ImageArray);  // Convert array to BufferedImage
}
```

Alpha = 255 : default value. the colour is completely opaque, visible.
Alpha = 0 : the colour is completely transparent, invisible



Example: Display images

```
// Display Black/White (gray-level) images as colour images:
for(int y=0; y<height; y++){
    for(int x =0; x<width; x++){
       ImageArray[x][y][1] = ImageArrayBW[x][y];  //r
       ImageArray[x][y][2] = ImageArrayBW[x][y];  //g
       ImageArray[x][y][3] = ImageArrayBW[x][y];  //b
    }
 }


// Display 2 images as one if not in 2 windows with GUI:
for(int y=0; y<height; y++){
    for(int x =0; x<width; x++){
       ImageArray[x][y][1] = ImageArray1[x][y][1];  //r
       ImageArray[x][y][2] = ImageArray1[x][y][2];  //g
       ImageArray[x][y][3] = ImageArray1[x][y][3];  //b
       ImageArray[width+x][y][1] = ImageArray2[x][y][1];  //r
       ImageArray[width+x][y][2] = ImageArray2[x][y][2];  //g
       ImageArray[width+x][y][3] = ImageArray2[x][y][3];  //b
    }
 }
```