University of Technology and Education

Faculty of Electrical & Electronic Engineering

Lecture:

IMAGE PROCESSING

Chapter 2:

Fundamentals

Nguyen Thanh Hai, PhD

1

Fundamentals

2.1 Image Representation

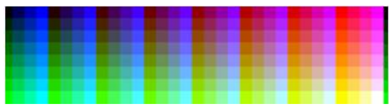
- Gray-level images or color images.
- The light intensity or brightness of an object shown at coordinates (x,y) of the image
-The maximum value on the range of gray level corresponding to a completely bright point and a point with a gray level of zero is a completely dark point
-The most popular ranges of gray level often used in typical images: 0 to 255, 0 to 511, 0 to 1023, etc.
-The gray levels are almost always set to be nonnegative integer numbers (real numbers)
-This saves a lot of digital storage spaces and significantly in processing digital images

University of Technology and Education
Faculty of Electrical & Electronic Engineering

Fundamentals

For each pixel, the gray is value from 0 to 255. Black level ladled value is 0 and 255 is white and 254 gray levels are in between. For example, in one 8-bit image, one gray level value is in the range (0 to 255).
For color images, in the system with 8-bit, there are 256 gray levels on one color

Each pixel can be represented on a byte (8-bit) and the maximum number of colors displayed at any point to 256.
For 8-bit format, based on three standard colors and it is divided as follows: 3-bit red, green and 2 bit 3 bit is blue



8-bit color, with 3 bits-red, 3 bits-green, and 2 bits-blue.

Bit	7	6	5	4	3	2	1	0
Data	R	R	R	G	G	G	B	B

Nguyen Thanh Hai, PhD.

University of Technology and Education
Faculty of Electrical & Electronic Engineering

Fundamentals

A color image has three values per pixel and they measure the intensity and chrominance of light. The actual information stored in the digital image data is the brightness information in each spectral band.

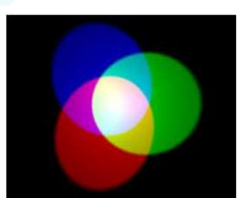
A typical grayscale image.

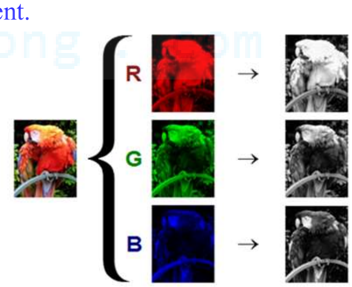
Eight bits per sample (24 bits per pixel) seem adequate for most uses

Particularly demanding applications may use 10 bits per sample or more

RGB image consists of Red, Green and Blue component.

On the other hand, some widely used image file formats and graphics cards may use only 8 bits *per pixel*, i.e., only 256 different colors, or 2–3 bits per channe





Nguyen Thanh Hai, PhD.

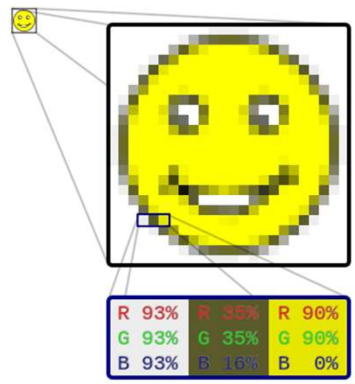
University of Technology and Education
Faculty of Electrical & Electronic Engineering

Fundamentals

YC_BC_R or $Y'C_BC_R$, is a family of color spaces used as a part of the color image pipeline in video and digital photography systems

Y′ is the luma component and C_B and C_R are the blue-difference and red-difference chroma components. Y′ (with prime) is distinguished from Y, which is luminance, meaning that light intensity is nonlinearly encoded based on gamma corrected RGB primaries

Y′CbCr is not an absolute color space; rather, it is a *way of encoding* RGB information. The actual color displayed depends on the actual RGB primaries used to display the signal



The smiley face in the top left corner is a raster image. When enlarged, individual pixels appear as squares. Zooming in further, they can be analyzed, with their colors constructed by adding the values for red, green and blue

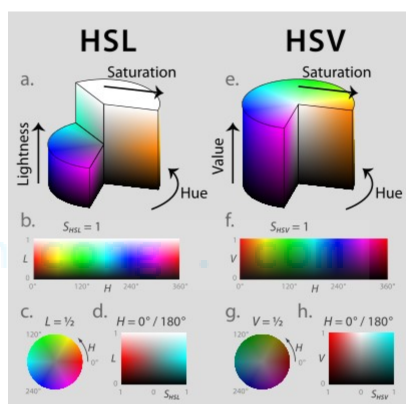
 http://en.wikipedia.org/wiki/YCbCr

University of Technology and Education
Faculty of Electrical & Electronic Engineering

Fundamentals

HSL (hue-saturation-lightness) and **HSV** (hue-saturation-value) are the two most common cylindrical (hình trụ)-coordinate representations of points in an RGB color model.

HSL (a–d) and HSV (e–h). Above (a, e): cut-away 3D models of each. Below: two-dimensional plots showing two of a model's three parameters at once, holding the other constant: cylindrical shells (b, f) of constant saturation, in this case the outside surface of each cylinder; horizontal cross-sections (c, g) of constant HSL lightness or HSV value, in this case the slices halfway down each cylinder; and rectangular vertical cross-sections (d, h) of constant hue, in this case of hues 0° red and its complement 180° cyan



University of Technology and Education
Faculty of Electrical & Electronic Engineering

Fundamentals

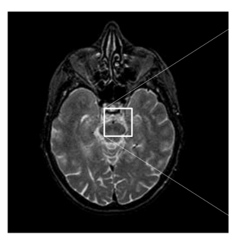
$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & \ddots & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix} \quad (2.1)$$

$$0 \le f(x,y) \le G-1$$

where *NxM (pixel)* and *G (number of gray levels)* is the integer power of 2 ($G=2^m$)

University of Technology and Education
Faculty of Electrical & Electronic Engineering

Fundamentals



173	145	141	170	159	146	165	211
124	99	121	173	174	158	182	223
78	94	120	144	140	140	145	163
116	139	127	115	149	160	129	112
130	129	104	97	131	146	146	142
88	81	73	76	86	86	104	147
72	70	74	80	61	61	72	115
68	71	73	72	75	75	72	75

Storage

- Memory size: **b = MxNxm (bit)**

where

- MxN: size of image
- m: number of bits of gray level

University of Technology and Education
Faculty of Electrical & Electronic Engineering

Fundamentals

Example 2.1:

Một ảnh có 512x512 có mức grey là 256 (2^m với m=8). Sử dụng công thức **b = NxNxm = 262.144 byte = 2.097.152 bit**

Cần phải giảm cỡ ảnh nhưng cũng đảm bảo chất lượng ảnh

Example 2.2:

Một ảnh có 512x256 có mức grey là (2^m với m=2). Sử dụng công thức **b = NxMxm = ??? byte = ??? bit**

Xác định số byte và bit của ảnh?

Example 2.3:

Một ảnh có NxM, với N=1024 có m=10.

Xác định số M nếu ảnh có số b=50x2^{15} byte.

9

Nguyen Thanh Hai, PhD.

University of Technology and Education
Faculty of Electrical & Electronic Engineering

Fundamentals

Format

- BMP — Windows Bitmap
- CUR — Cursor File
- GIF — Graphics Interchange Format
- HDF4 — Hierarchical Data Format
- ICO — Icon File
- PNG — Portable Network Graphics
- PPM — Portable Pixmap
- RAS — Sun Raster
- TIFF — Tagged Image File Format
- XWD — X Window Dump

- JPEG — Joint Photographic Experts Group
- JPEG 2000 — Joint Photographic Experts Group 2000
- PBM — Portable Bitmap
- PCX — Windows Paintbrush
- PGM — Portable Graymap

10

Nguyen Thanh Hai, PhD.

University of Technology and Education
Faculty of Electrical & Electronic Engineering

Fundamentals

2.2 Image resolution

The density of pixels in an image is viewed as its own resolution

Depends on size of an image

Original image

Nguyen Thanh Hai, PhD.

University of Technology and Education
Faculty of Electrical & Electronic Engineering

Fundamentals

2.2 Image resolution (cont)

Depends on the value of m

m=8 m=7 m=6

Nguyen Thanh Hai, PhD.

University of Technology and Education
Faculty of Electrical & Electronic Engineering

Fundamentals

m=2 m=1

Nguyen Thanh Hai, PhD.

University of Technology and Education
Faculty of Electrical & Electronic Engineering

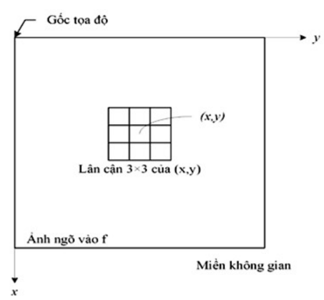
Fundamentals

2.2. Image Operators

$$g(x,y) = T[f(x,y)]$$

where $f(x, y)$ is the input image, $g(x, y)$ is the output image and the operator $T\ f$ for through the neighboring pixels of (x, y)

Lưu ý với các điểm ảnh ở biên thì các điểm ảnh lân cận có giá trị là 0 đối với các lân cận không tồn tại như hình 2.8.



The neighboring points in figure are shifted from pixel to pixel another row to create new images follow the operator T

- Negative transform
- Log transform
- Gamma correction

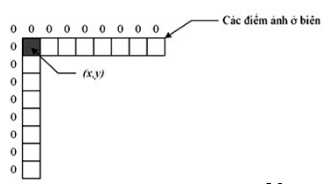


Figure 2.8. For the pixels around the image boundary, as the operator is applied for grayscale conversion, the pixel does not exist and considered as zero

Nguyen Thanh Hai, PhD.

University of Technology and Education
Faculty of Electrical & Electronic Engineering

Fundamentals

2.2.1. Negative transform

$$s = L - 1 - r$$

Negative image to grayscale intensity range (*L-1*)

(a) (b)

Nguyen Thanh Hai, PhD.

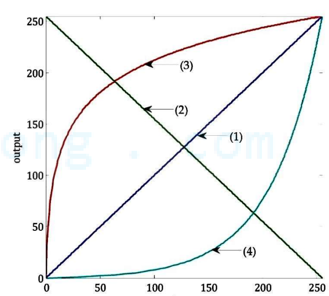
University of Technology and Education
Faculty of Electrical & Electronic Engineering

Fundamentals

2.2.2. Log transform

$$s = c\log(1 + r)$$

where c is constant and assume that one uses the shape curve (3), (4) as shown in Figure 2.10. The transform function will map a narrow range of low gray level intensity of the input image to produce the inverse large image with high gray level intensity



(a) (c)

Figure 2.10. transform functions with intensity change: (1) invariable; (2) negative transforms; (3) log function; (4) inverse log function

Nguyen Thanh Hai, PhD.

University of Technology and Education
Faculty of Electrical & Electronic Engineering

Fundamentals

2.2.3. Exponential transform

$$s = cr^{\gamma}$$

where c and γ are positive constants

Ex 2.4: The original form 2.13(a) be represented on the screen CRT as shown in Figure 2.13(b). Image being shown on the screen will be affected due to use of the exponential function $\gamma = 2.5$. To improve this phenomenon, before displaying on a CRT, the image needs to be preprocessed by an exponential function, $\gamma = 0.4$ as in Figure 2.13(c). Results as in Figure 2.13(d) to produce an image close to original image as in Figure 2.13(b).

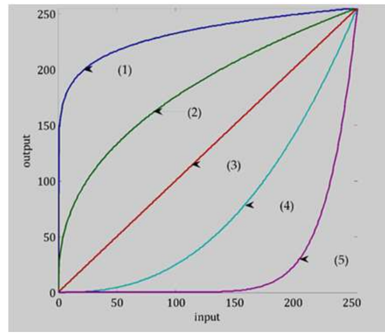


Figure 2.12. *exponential function: (1)* $\gamma = 0.10$; *(2)* $\gamma = 0.40$; *(3)* $\gamma = 1.00$; *(4)* $\gamma = 2.50$; *(5)* $\gamma = 10.0$

Nguyen Thanh Hai, PhD.

University of Technology and Education
Faculty of Electrical & Electronic Engineering

Fundamentals

```
clear all;
f1=imread('cameraman.tif');% doc anh
f=double(f1)/255;
gamma1=2.5;
g1=255*f.^gamma1;
g1=uint8(g1);
gamma2=0.4;
g2=255*f.^gamma2;
g2=uint8(g2);
g3=(double(g2)/255).^gamma1;
g3=255*g3;
g3=uint8(g3);
subplot(2,2,1)
imshow(f1)
xlabel('(a)')
subplot(2,2,2)
imshow(g1)
xlabel('(b)')
subplot(2,2,3)
imshow(g2)
xlabel('(c)')
subplot(2,2,4)
imshow(g3)
xlabel('(d)')
```

Figure 2.13. *gamma correction for CRT display: (a) original image; (b)* Show the original image on a CRT; *(c) Adjust by gamma with* $\gamma = 0.4$; *(d) image after adjusting on the CRT*

Nguyen Thanh Hai, PhD.

University of Technology and Education
Faculty of Electrical & Electronic Engineering

Fundamentals

Practice to MATLAB

Ex 1.1: *Read and display image.*

```
f=imread('peppers.png');
Info_f=imfinfo('peppers.png');
```

Ngõ ra của hàm imfinfo cho phép truy vấn thông tin của ảnh. Với ảnh peppers.png, kết quả hiển thị tại cửa sổ Command Window:

```
Info_I =
          Filename: [1x65 char]
       FileModDate: [1x20 char]
          FileSize: 287677
            Format: 'png'
     FormatVersion: []
             Width: 512
            Height: 384
          BitDepth: 24
         ColorType: [1x9 char]
FormatSignature: [1x8 double]
          Colormap: []
         Histogram: []
InterlaceType: 'none'
      Transparency: 'none'
SimpleTransparencyData: []
   BackgroundColor: []
   RenderingIntent: []
    Chromaticities: []
             Gamma: []
       XResolution: []
       YResolution: []
    ResolutionUnit: []
           XOffset: []
           YOffset: []
        OffsetUnit: []
   SignificantBits: []
      ImageModTime: [1x26 char]
             Title: []
            Author: []
       Description: [1x13 char]
         Copyright: [1x29 char]
      CreationTime: []
          Software: []
        Disclaimer: []
           Warning: []
            Source: []
           Comment: []
         OtherText: []
```


Nguyen Thanh Hai, PhD

University of Technology and Education
Faculty of Electrical & Electronic Engineering

Fundamentals

Ex 1.1: *display image*

```
A=imread('rice.png');
B=imread('cameraman.tif');
C=imread('trees.tif');
figure;
subplot(1,3,1)
imshow(A)
subplot(1,3,2)
imshow(B)
subplot(1,3,3)
imshow(C)
```

Rice

Cameraman

Trees

Nguyen Thanh Hai, PhD

University of Technology and Education
Faculty of Electrical & Electronic Engineering

Fundamentals

EX 1.3: *Save an image matrix into a graphic file in Window*

```
clear all;
f=imread('peppers.png');
f_gs=rgb2gray(f);
imwrite(f_gs,'pepper_gray.png','png');
```

EX 1.4: Imwrite function to save images into different formats such as tif, gif, jpg, bmp…

```
imwrite(f_gs,'pepper_gray.tif','tif');
imwrite(f_gs,'pepper_gray.png','png');
imwrite(f_gs,'pepper_gray.jpg','jpg','Quality',50);

Infor =
        Filename: [1x51 char]
     FileModDate: [1x20 char]
        FileSize: 7237
          Format: 'jpg'
   FormatVersion: ''
           Width: 512
          Height: 384
        BitDepth: 8
       ColorType: 'grayscale'
 FormatSignature: ''
 NumberOfSamples: 1
    CodingMethod: 'Huffman'
   CodingProcess: 'Sequential'
         Comment: {}
```

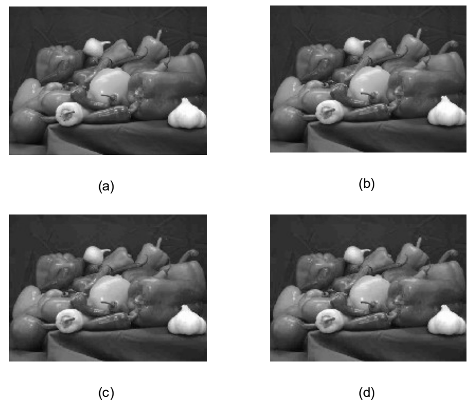


Figure 1.7. Results save images in JPEG format according to different compression ratios: (a) 80%; (b) 60%; (c) 40% and (d) 20%

Check compression ratios for the above images;

Nguyen Thanh Hai, PhD

University of Technology and Education
Faculty of Electrical & Electronic Engineering

Fundamentals

The size of the uncompressed original image is calculated using the formula:

$$I_{bytes} = Width \times Height \times BitDepth / 8 \quad ; m = BitDepth$$

The compression ratio is calculated as follows:

$$Compressio\,nRatio = I_{bytes} \,/\, FileSize$$

FileSize is the size of the file in byte

Example of the image in Figure 1.7 (d) with NxM=512x512 and 256 gray levels, so compression ratio is calculated as follows:

$$CompressionRatio = (512 \times 512 \times 8/8)/11831 \approx 22.15$$

More examples

Nguyen Thanh Hai, PhD

Fundamentals

Cho 1 ảnh f có kích cỡ ảnh (chiều rộng M= 1 mega pixel; chiều dài N= 2 mega pixel), với 16 bit mức xám và tỷ lệ nén ảnh là 32. Hãy xác định:

a. Số byte để lưu ảnh này

b. số byte của file (Filesize) của ảnh

Fundamentals

EX 1.4: *Change size of image*

```
clear all;
f=imread('peppers.png');
f_gs=rgb2gray(f);
f_256=imresize(f,0.5);
f_128=imresize(f_gs,[128 128]);
```

Imresize function allows to resize image by specifying the size of the output image [*width height*] (the example above is [*128,128*]) or coefficient ratio (*0.5*). *Imresize* function can be used with the input image is black-white and color images.

black-white images:	color images:
size(f_128)	size(f_256)
ans =	ans =
128 128	256 256 3

University of Technology and Education
Faculty of Electrical & Electronic Engineering

Fundamentals

EX 2.1: Convert the RGB color image into the grayscale type, binary and index images

```
f=imread('peppers.png');
fg=rgb2gray(f);
fi=rgb2ind(f,256);
fb=im2bw(f);
subplot(2,2,1)
imshow(f)
xlabel('(a)')
subplot(2,2,2)
imshow(fg)
xlabel('(b)')
subplot(2,2,3)
imshow(fi)
xlabel('(c)')
subplot(2,2,4)
imshow(fb)
xlabel('(d)')
```

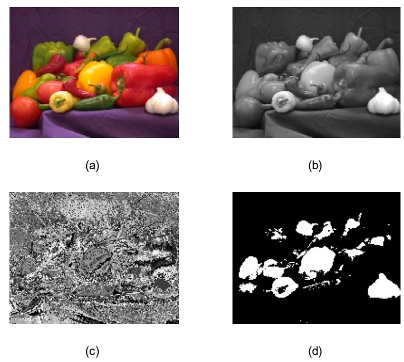


Figure 2.3. The different types of images: (a) color image; (b) gray image; (c) index gray; (d) Binary Image

Nguyen Thanh Hai, PhD

University of Technology and Education
Faculty of Electrical & Electronic Engineering

Fundamentals

EX 2.2: Changing image resolution

```
f=imread('cameraman.tif');
f128=imresize(f, [128 128]);
f64=imresize(f, [64 64]);
f32=imresize(f, [32 32]);
subplot(2,2,1)
imshow(f)
xlabel('(a)')
subplot(2,2,2)
imshow(f128,'InitialMagnification','fit')
xlabel('(b)')
subplot(2,2,3)
imshow(f64,'InitialMagnification','fit')
xlabel('(c)')
subplot(2,2,4)
imshow(f32,'InitialMagnification','fit')
xlabel('(d)')
```

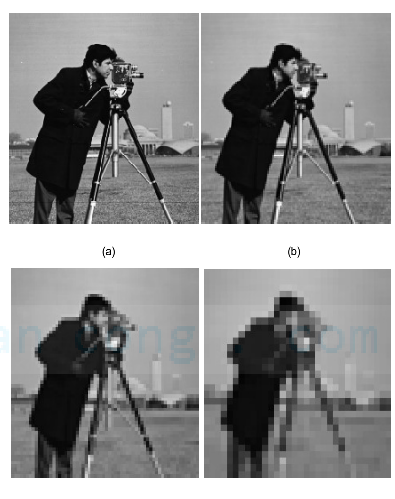


Figure 2.4. *Images with different resolutions: (a) 256x256; (b) 128x128; (c) 64x64; (d) 32x32*

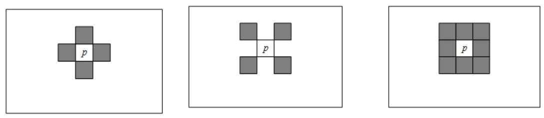
Nguyen Thanh Hai, PhD

Fundamentals

2.1.4. Điểm ảnh lân cận

Với một điểm ảnh *p* tại tọa độ (*x*, *y*), luôn có 4 điểm ảnh xung quanh theo phương ngang và dọc với tọa độ được xác định cụ thể sau:

Tập các điểm ảnh này được gọi là lân cận 4 của *p*, gọi tắt là $N_4(p)$. Ngoài ra, một dạng lân cận nữa là theo đường chéo được gọi tắt là $N_D(p)$ với các tọa độ được cho sau:



(a) (b) (c)

Figure 2.5. The adjacent basic forms: (a) 4 horizontal and vertical neighborhood points; (b) 4 diagonal points; (c) 8 neighborhood points around

Fundamentals

2.1.6. Distance measurement

Assuming the three pixels are given as *p, q* and *z*, the coordinates are determined (*x, y*), (*s, t*), (*v, w*), respectively. D is the distance to the conditions:

$$\textbf{(a)}\ D(p,q) \geq 0 \quad \big(D(p,q) = 0 \ \text{ iff } \ p = q\big),$$
$$\textbf{(b)}\ D(p,q) = D(q,p), \text{ and}$$
$$\textbf{(c)}\ D(p,z) \leq D(p,q) + D(q,z).$$

Khoảng cách Euclid giữa hai điểm *p* và *q* được tính như sau:

$$D_e(p,q) = \left[(x-s)^2 + (y-t)^2\right]^{\frac{1}{2}}.$$

Để đo được khoảng cách này, các điểm ảnh có khoảng cách nhỏ hơn hay bằng với giá trị r từ tọa độ (x, y) thì đều là những điểm nằm trong vòng tròn có bán kính r với góc tọa độ là (x, y).

Khoảng cách D_4 giữa điềm p và q được định nghĩa như sau:

$$D_4(p,q) = |x-s| + |y-t|.$$

Trong trường hợp này, các điểm ảnh có khoảng cách D_4 từ tọa độ (x, y) nhỏ hơn hay bằng với giá trị r từ tâm của hình thoi có tọa độ (x, y). Lấy ví dụ, các pixel với khoảng cách $D_4 \leq 2$ từ tọa độ trung tâm là (x, y) tạo thành đường bao với khoảng cách không đổi:

Fundamentals

Các điểm ảnh có khoảng cách $D_4 = 1$ là lân cận với điểm có tọa độ (x, y).

		2		
	2	1	2	
2	1	0	1	2
	2	1	2	
		2		

Khoảng cách D_8 giữa hai điểm p và q được xác định như sau:

$$D_8(p, q) = \max(|x - s|, |y - t|)$$

Trong trường hợp này, các điểm ảnh có kích thước D_8 từ điểm gốc có tọa độ (x, y) nhỏ hơn hay bằng với giá trị r tạo thành hình vuông có tâm đặt tại gốc tọa độ (x, y). Ví dụ, những điểm ảnh với kích thước $D_8 \leq 2$ từ tọa độ (x, y) tạo thành đường bao với khoảng cách không đổi:

2	2	2	2	2
2	1	1	1	2
2	1	0	1	2
2	1	1	1	2
2	2	2	2	2

Các điểm ảnh có khoảng cách $D_8 = 1$ là lân cận với điểm có tọa độ (x, y).

Fundamentals

Ex 2.3*:* changes of the gray level using transform functions, including negative and logarithm

```
clear all;
f1=imread('cameraman.tif');
f=double(f1);
g_neg=255-f;
g_neg=uint8(g_neg);
g_log=255/log2(256)*log2(f+1);
g_log=uint8(g_log);
subplot(1,3,1)
imshow(f1)
xlabel('(a)')
subplot(1,3,2)
imshow(g_neg)
xlabel('(b)')
subplot(1,3,3)
imshow(g_log)
xlabel('(c)')
```

(a) (b) (c)

Figure 2:11. Perform image transformations: (a) original image. (b) negative image. (c) Image with the log function

University of Technology and Education
Faculty of Electrical & Electronic Engineering

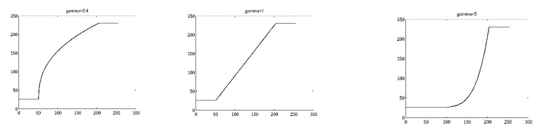
Fundamentals

Ex 2.5: Adjust gamma using *imadjust* in Toolbox

```
clear all
f1=imread('cameraman.tif');
f=double(f1);
f=f/255;

g1=imadjust(f,[0.2 0.8],[0.1 0.9],0.4);
g1=uint8(g1*255);

g2=imadjust(f,[0.2 0.8],[0.1 0.9,1]);
g2=uint8(g2*255);

g3=imadjust(f,[0.2 0.8],[0.1 0.9],5);
g3=uint8(g3*255);

subplot(1,3,1)
imshow(f1)
xlabel('(a)')
subplot(1,3,2)
imshow(g_neg)
xlabel('(b)')
subplot(1,3,3)
imshow(g_log)
xlabel('(c)')
```

Note: syntax of *imadjust* :
J = imadjust(I,[low_in; high_in],[low_out; high_out],gamma)

Where parameters [low_in; high_in], [low_out; high_out] must be in the range of [0 1]. So it needs to be standardized gray level intensity of input image



(a) (b) (c)

Figure 2.14. *mapping functions in Ex 2.5: (a) gamma=0.4;(b) gamma=1; (c) gamma=5*

University of Technology and Education
Faculty of Electrical & Electronic Engineering

Fundamentals

(a) (b) (c)

Figure 2.14. *mapping functions in Ex 2.5: (a) gamma=0.4;(b) gamma=1; (c) gamma=5*

University of Technology and Education
Faculty of Electrical & Electronic Engineering

Fundamentals

Problem: Give 1 input matrix image f with intensity r below, determine the intensity values of the output image using the following transformations:

1. Log function, in which c is 2
2. Negative function, in which the gray bit number is 3
3. Exponent function, in which c is 2 and gamma is 3,5

2	10	5
7	0	1
3	24	25

University of Technical Education
Faculty of Electrical & Electronic Engineering

Fundamentals

The End