C2 MATLAB VÀ XỬ LÝ ẢNH CƠ BẢN

MATLAB

- Matlab là công cụ rất mạnh để giải các bài toán xử lý ảnh và thị giác máy tính, có các toolbox sau:
  - Image Acquisition Toolbox thu nhận ảnh
  - Image Processing Toolbox xử lý ảnh
  - Computer vision ToolBox thị giác máy tính
  - Parallel Computing Toolbox tính toán song song
- Matlab sử dụng bộ xử lý đồ họa GPU với kiến trúc **CUDA** (Compute Unified Device Architecture - Kiến trúc thiết bị tính toán hợp nhất) là một kiến trúc tính toán song song do NVIDIA phát triển. Kết hợp Parallel Computing ToolBox để giảm thời gian tính tóan
- Bản mới nhất là R2018a X64

Image Acquisition Toolbox

Getting Started | Examples | Troubleshooting | Release Notes

Device Connection

Establish and manage connection between image acquisition device and MATLAB®

Image Preview and Device Configuration

Preview image and adjust acquisition parameters such as brightness, resolution, and region of interest (ROI)

Image Data Acquisition

Specify acquisition parameters such as data logging, number of frames, and triggering; acquire image data

Image Acquisition in Simulink

Bring live video data into Simulink® models, generate code

Creating Custom Adaptors

Use the Image Acquisition Toolbox™ Adaptor Kit to create an adaptor

MATLAB Functions | Simulink Blocks | PDF Documentation

Image Processing Toolbox

Getting Started | Examples | Release Notes

Import, Export, and Conversion
Image data import and export, conversion of image types and classes

Display and Exploration
Interactive tools for image display and exploration

Geometric Transformation, Spatial Referencing, and Image Registration
Scale, rotate, perform other N-D transformations, provide spatial information, align images using automatic or control point registration

Image Enhancement
Contrast adjustment, morphological filtering, deblurring, and other image enhancement tools

Image Analysis
Region analysis, texture analysis, pixel and image statistics

Color
Color transforms, support for International Color Consortium (ICC) profiles

Code Generation
Generate C/C++ code and MEX functions for toolbox functions

GPU Computing
Run image processing code on a graphics processing unit (GPU)

MATLAB Functions | PDF Documentation

Computer Vision System Toolbox

Getting Started | Examples | Release Notes

Video Input, Output, and Graphics
Importing, exporting, color space formatting, conversions, display, annotation

Registration and Stereo Vision
Registration, stereo rectification, disparity map computation, feature detection, feature extraction, feature matching

Object Detection, Motion Estimation, and Tracking
Object detection, optical flow, block matching, background estimation

Geometric Transformations
Similarity, affine, and projective transformations

Filters, Transforms, and Enhancements
FIR filtering, frequency and Hough transforms, Gaussian pyramiding, deinterlacing, contrast enhancement, noise removal

Statistics and Morphological Operations
Statistical operations, morphology, connected component analysis

Code Generation and Fixed-Point Design
C Code generation, fixed-point data type support

Define New System Objects
Define new kinds of System object™

Classes | MATLAB Functions | System Objects | Simulink Blocks | PDF Documentation

ĐỌC ẢNH

- Ảnh số được lưu lại trong máy tính với nhiều định dạng

Format Name	Description	Recognized Extensions
TIFF	Tagged Image File Format	.tif, .tiff
JPEG	Joint Photographic Experts Group	.jpg, .jpeg
GIF	Graphics Interchange Format†	.gif
BMP	Windows Bitmap	.bmp
PNG	Portable Network Graphics	.png
XWD	X Window Dump	.xwd

Trong Matlab dùng lệnh imread('filename', 'format') đọc ảnh vào một biến

a=imread('c:/miss.jpg');

size(a) % *kich thước ảnh*

whos a *% thông số ảnh*

imshow(a,[ ]); *%hiển thị ảnh*

image(a) %hiển thị ảnh

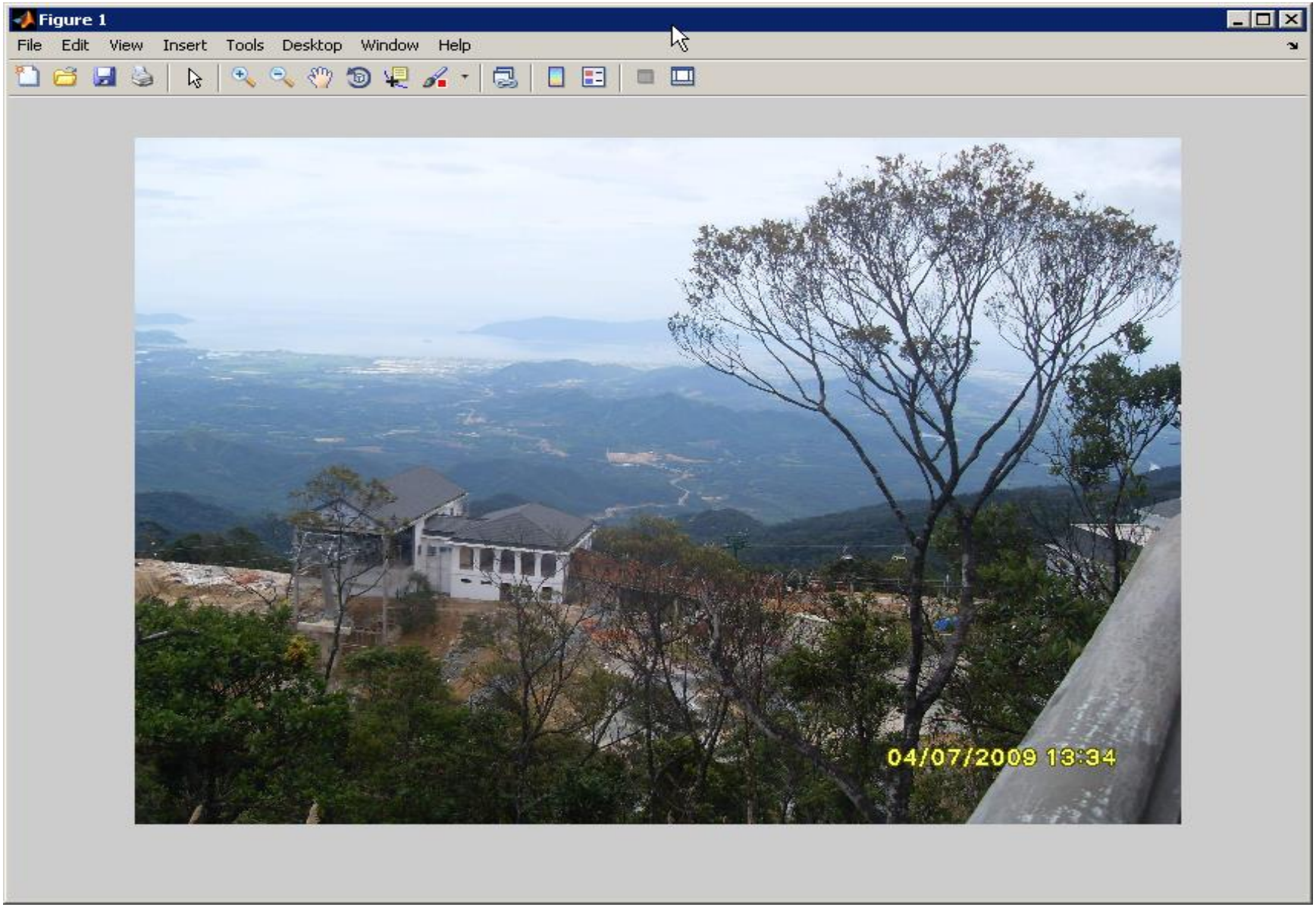
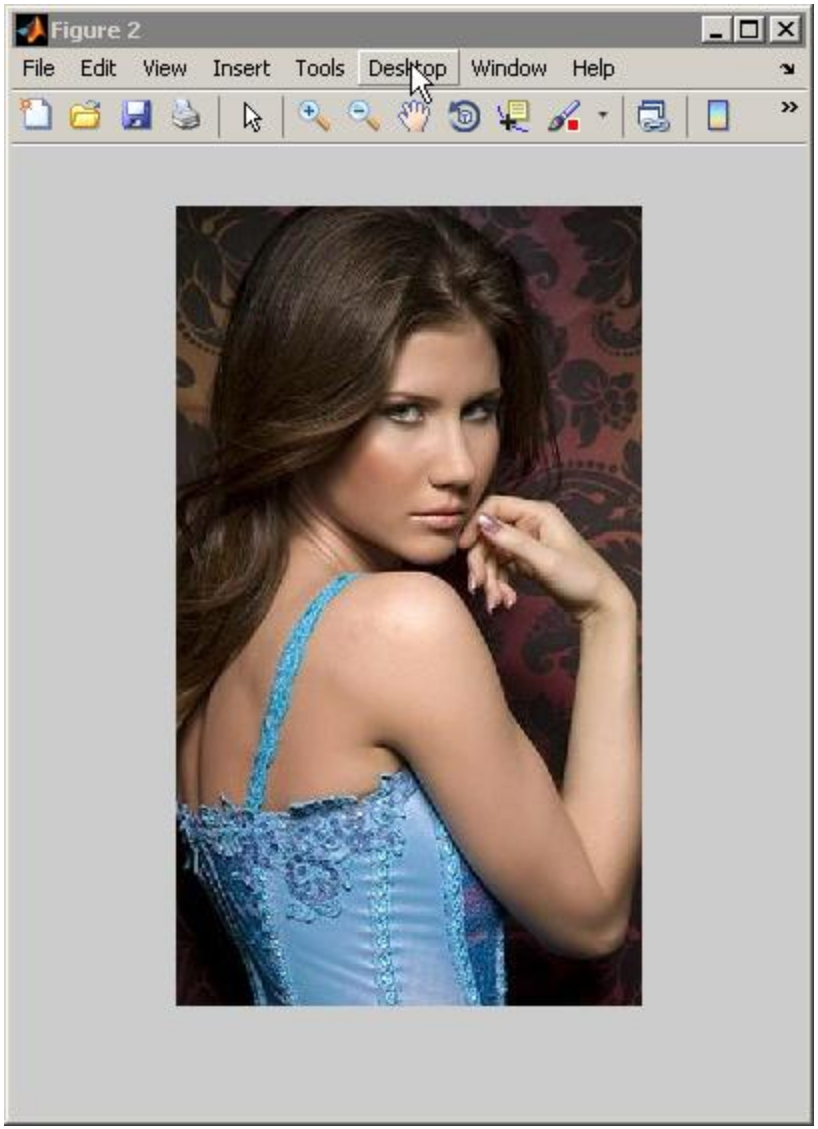
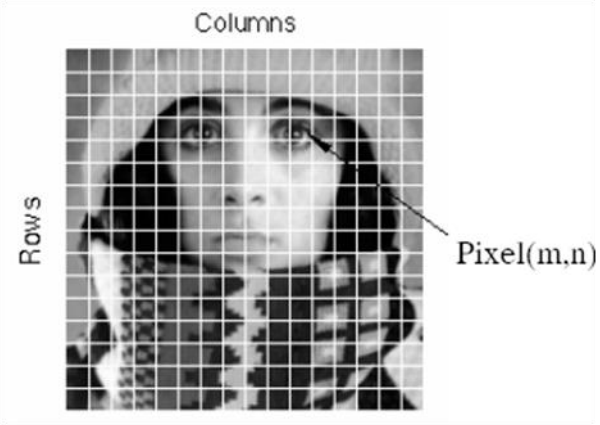


Figure 1
File Edit View Insert Tools Desktop Window Help
04/07/2009 13:34

- im=imread('c:\annachapman.jpg');
- figure, imshow(im);



BIỂU DIỄN ẢNH SỐ



Ảnh được biểu diễn bởi các pixel.

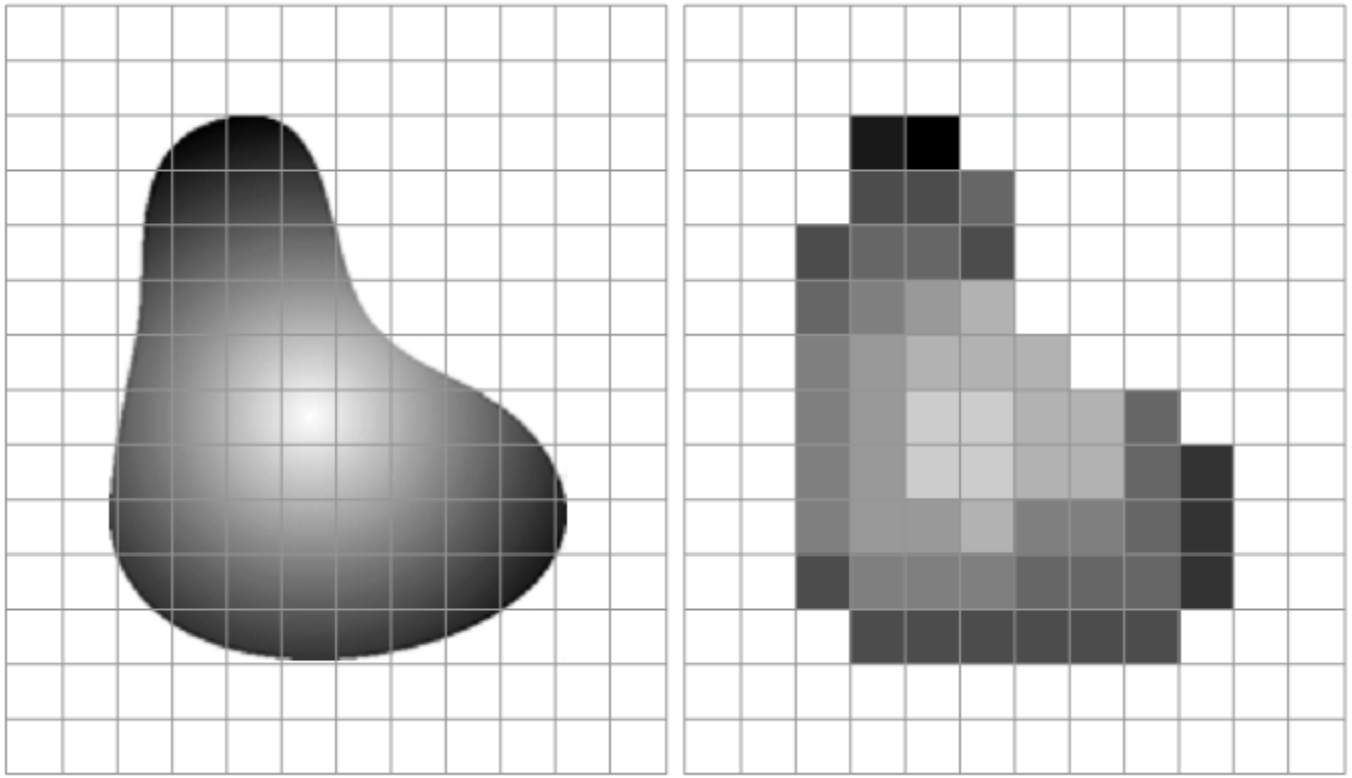
BIỂU DIỄN ẢNH SỐ

- Ảnh gồm tập hợp các điểm ảnh
- Ảnh 2D là hàm f(x,y); x, y là tọa độ điểm ảnh, f là cường độ sáng ở điểm đó; với ảnh đơn sắc, f gọi là mức xám
- Ảnh màu là tổ hợp các ảnh 2D
- Ảnh RGB là tổ hợp ba ảnh thành phần đỏ, lá cây, xanh
- Với ảnh số các đại lượng x, y, f là rời rạc
- Ảnh số biểu thị bằng ma trận M hàng N cột, mỗi phần tử trong ma trận gọi là điểm ảnh (picture element, pel, pixel), có giá trị là cường độ sáng ảnh, là số nguyên $L=2^K$, K là số bit biểu diễn cường độ

BIỂU DIỄN ẢNH SỐ

- Ảnh M, N ,K có dung lượng là M*N*K
- Độ phân giải ảnh (spatial resolution) là tích số M*N, ảnh có độ phân giải lớn thì kích thước điểm ảnh nhỏ
- Ảnh nhị phân có K = 1, cường độ sáng tại mỗi điểm ảnh có một trong hai giá trị.
- Ảnh đơn sắc có K > 1, thường là 8
- Ảnh màu gồm ba màu cơ bản phối hợp, mỗi màu cơ bản biểu diễn độ sáng bằng K >1
- Ảnh số được lưu trong máy tính dưới nhiều định dạng
- Ảnh xám biểu thị bằng ma trận M*N, ảnh màu biểu thị bằng ba hay bốn ma trận M*N
- Ảnh số được xử lý theo các phép tính ma trận

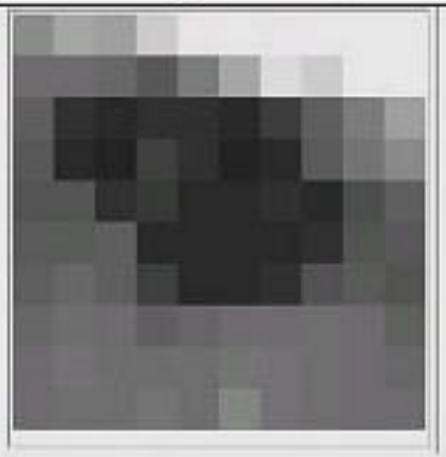
BIỂU DIỄN ẢNH SỐ



Ảnh liên tục đổi sang ảnh số 12*14 pixel

BIỂU DIỄN ẢNH SỐ

a,



b,

142	174	164	218	250	255	250	252	255	255
107	107	102	80	127	174	237	218	252	255
90	34	24	34	34	24	51	88	127	164
80	26	19	53	34	19	24	85	117	137
78	76	34	44	26	26	34	24	71	90
85	85	90	26	26	26	26	34	76	83
88	102	90	53	26	26	34	73	85	78
102	110	105	90	98	105	105	110	107	93
107	115	110	110	110	117	115	110	107	102
105	110	110	117	110	132	115	110	107	105

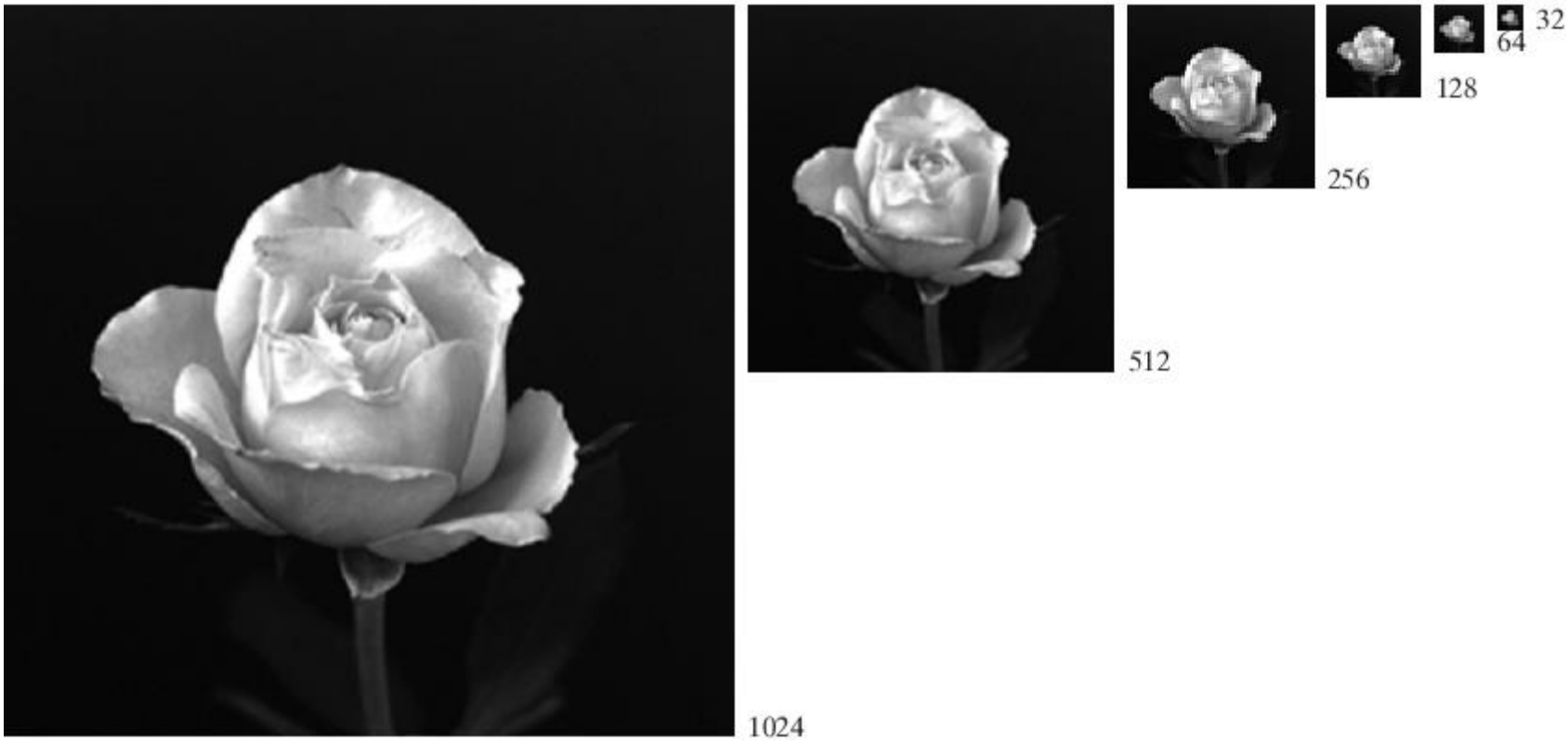
c,

a, Ảnh thật 10x10; b, Ảnh được zoom; c, Mô tả ảnh bằng ma trận điểm

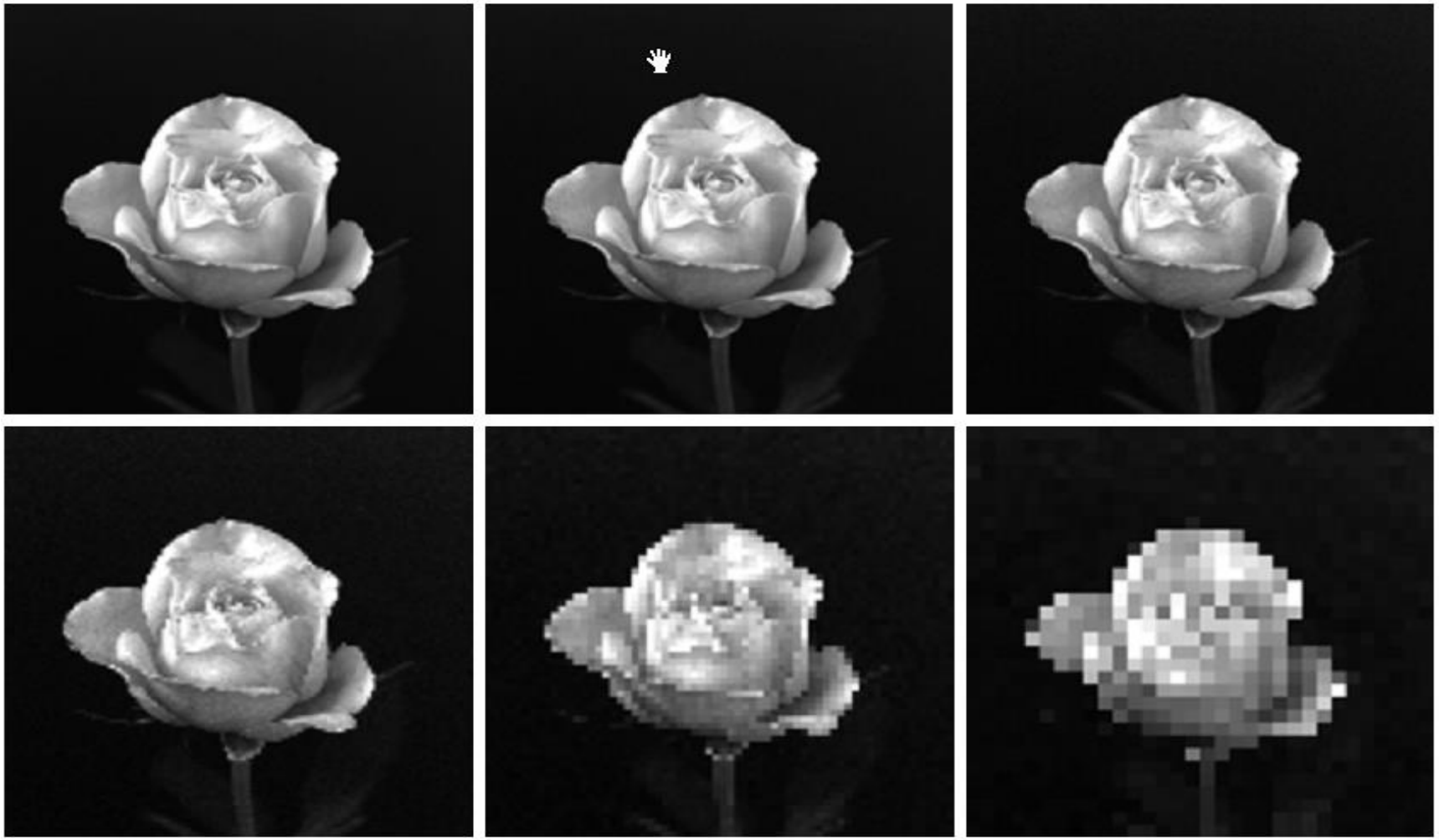
BIỂU DIỄN ẢNH SỐ

- Ảnh kích thước lớn có thể đưa về ảnh kích thước nhỏ hơn bằng cách bỏ một số hàng và cột, gọi là subsampling, sau đó đặt chúng sát nhau, giả sử mỗi pixel biểu thị trên giấy hay màn hình có kích thước cố định thì ảnh subsampling sẽ có kích thước nhỏ hơn
- Ảnh kích thước nhỏ khi zoom lên sẽ bị hiện tượng bàn cờ (checkerboard)

CO ẢNH (SHRINKING, SUBSAMPLING, UNDERSAMPLING)



ZOOM ẢNH



a b c
d e f

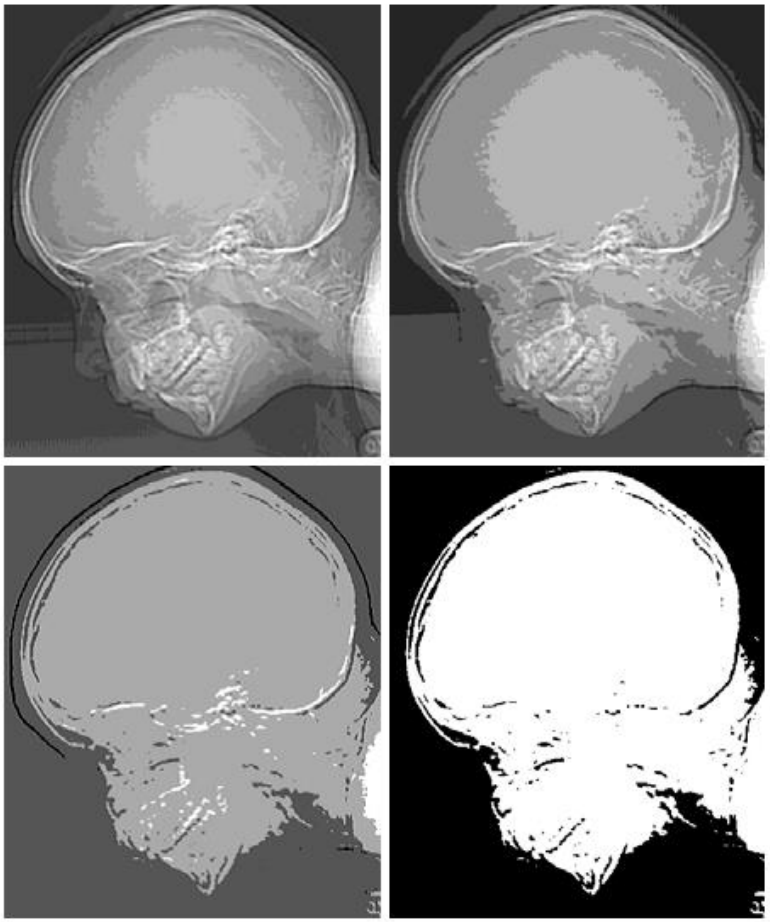
(a) 1024 × 1024,8-bit image. (b) 512 × 512 image resampled into 1024 × 1024 pixels by row and column duplication. (c) through (f) 256 × 256, 128 × 128, 64 × 64, and 32 × 32 images resampled into 1024 × 1024 pixels.

CƯỜNG ĐỘ XÁM

- Cường độ sáng tại mỗi pixel được lấy mẫu bằng k bit, gọi là độ phân giải mức xám, nếu k nhỏ sẽ bị hiện tượng mất đường biên (false contouring)
- Đối với thị giác máy tính ảnh số có kích thước 256*256*8 là phù hợp
- Ảnh nhị phân có k = 1, mức đen giá trị 0, mức trắng giá trị 1
- Thông thường k=8, có 256 mức xám, 0 là mức đen, 255 là mức trắng

CƯỜNG ĐỘ SÁNG

Mức xám thay đổi từ 16 (k=4) đền k=1



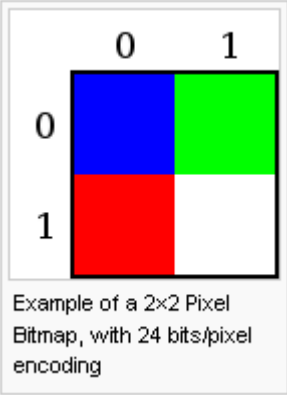
NỘI SUY ẢNH

- Để tránh hiện tượng bàn cờ khi zoom ảnh cần phải thêm các điểm ảnh mới và gán cường độ sáng cho các điểm ảnh này

ĐỊNH DẠNG ẢNH (Image Format)

- Ảnh được lưu trữ trên máy tính với nhiều định dạng
- Có nhiều định dạng ảnh
- Ảnh thường được nén để giảm kích thước file
- Ảnh định dạng theo kiểu quét (raster) hay vector
- Ảnh raster biểu diễn dưới dạng ma trận các điểm ảnh
- Ảnh vector lưu trữ sự quan hệ giữa các điểm ảnh
- BMP (Windows bit map) định dạng lưu ảnh không nén, tốn nhiều bộ nhớ, được hỗ trợ bởi nhiều phần mềm chạy trên windows, hỗ trợ ảnh màu đến 32 bit, bảng màu BGR
- Cấu trúc tập tin ảnh BMP bao gồm 4 phần
  - **Bitmap Header** (14 bytes): giúp nhận dạng tập tin bitmap.
  - **Bitmap Information Header** (số byte thay đổi): lưu một số thông tin chi tiết giúp hiển thị ảnh.
  - **Color Palette** (4*x bytes), x là số màu của ảnh: định nghĩa các màu sẽ được sử dụng trong ảnh.
  - **Bitmap Data:** lưu dữ liệu ảnh từ trái sang phải, từ dưới lên trên, mỗi hàng ảnh được bổ sung sao cho số byte mỗi hàng là bội số của 4

ẢNH BMP



Example of a 2×2 Pixel Bitmap, with 24 bits/pixel encoding

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

Offset	Size	Hex Value	Value	Description
BMP Header				
0h	2	42 4D	"BM"	Magic number (unsigned integer 66, 77)
2h	4	46 00 00 00	70 Bytes	Size of the BMP file
6h	2	00 00	Unused	Application specific
8h	2	00 00	Unused	Application specific
Ah	4	36 00 00 00	54 bytes	Offset where the Pixel Array (bitmap data) can be found
DIB Header				
Eh	4	28 00 00 00	40 bytes	Number of bytes in the DIB header (from this point)
12h	4	02 00 00 00	2 pixels	Width of the bitmap in pixels
16h	4	02 00 00 00	2 pixels	Height of the bitmap in pixels
1Ah	2	01 00	1 plane	Number of color planes being used
1Ch	2	18 00	24 bits	Number of bits per pixel
1Eh	4	00 00 00 00	0	BI_RGB, no Pixel Array compression used
22h	4	10 00 00 00	16 bytes	Size of the raw data in the Pixel Array (including padding)
26h	4	13 0B 00 00	2,835 pixels/meter	Horizontal resolution of the image
2Ah	4	13 0B 00 00	2,835 pixels/meter	Vertical resolution of the image
2Eh	4	00 00 00 00	0 colors	Number of colors in the palette
32h	4	00 00 00 00	0 important colors	0 means all colors are important
Start of Pixel Array (bitmap data)				
36h	3	00 00 FF	0 0 255	Red, Pixel (0,1)
39h	3	FF FF FF	255 255 255	White, Pixel (1,1)
3Ch	2	00 00	0 0	Padding for 4 byte alignment (could be a value other than zero)
3Eh	3	FF 00 00	255 0 0	Blue, Pixel (0,0)
41h	3	00 FF 00	0 255 0	Green, Pixel (1,0)
44h	2	00 00	0 0	Padding for 4 byte alignment (could be a value other than zero)

ĐỊNH DẠNG ẢNH (Image Format)

- JPEG (http://en.wikipedia.org/wiki/JPEG) định dạng ảnh nén có mất mát (lossy) hay không mất mát (lossless) do nhóm Joint Photographic Expert Group hoàn thiện, giúp làm giảm kích thước ảnh
- GIF (Graphic Interchange Format) dùng biểu diễn ảnh động trên trang web, ảnh nén không mất mát có kích thước nhỏ và 256 màu.
- PNG (Portable Network Graphic) tương tự GIF, có thể biểu diễn 24 bit màu
- TIFF (Tagged image File Format) định dạng ảnh mềm dẻo, bao gồm ảnh nén và không nén

ĐỔI MA TRẬN RA ẢNH XÁM, LƯU ẢNH

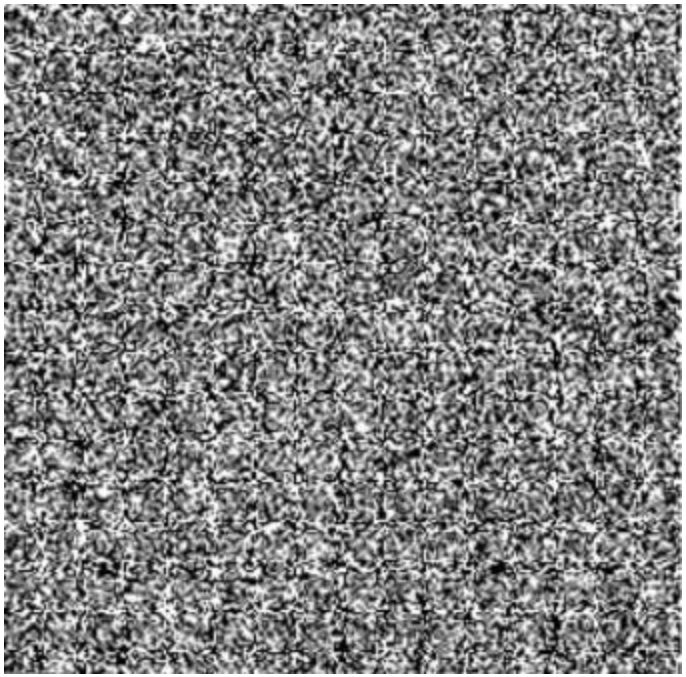
- I = mat2gray(A, [amin amax])
  I = mat2gray(A)
- A là ma trận, amin amax là giá trị của phần tử tương ứng đen (0.0) và trắng (1.0), I là ảnh xám, nếu không có thông số amin amax thì lấy giá trị trong A

```
>> s =s=[1 2 3;4 5 6;7 8 9]
   s- 1     2     3
      4     5     6
      7     8     9
>> i=mat2gray(s,[1  10])
i =
         0    0.1111    0.2222
    0.3333    0.4444    0.5556
    0.6667    0.7778    0.8889
>> imshow(i)
>> imwrite(i,'a.jpg','jpg')
>> imshow('a.jpg')
```

- >> s=[10 10 10 10 10; 10 0 0 0 10;10 0 0 0 10;10 0 0 0 10;10 0 0 0 10;10 0 0 0 10;10 0 0 0 10;10 10 10 10 10] % number zero
- >>i=mat2gray(s,[1 10]);
- >>i=mat2gray(s,[10 1]);%inverse
- >>imshow(i);

- ```
  row = 256;
  col = 256;
  img = zeros(row, col);
  img(100:105, :) = 0.5;
  img(:, 100:105) = 1;
  figure;
  imshow(img);
  ```
  ```

- row = 256;
  col = 256;
  img = rand(row, col);
  img = round(img);
  figure;
  imshow(img);



MỘT SỐ HÀM MA TRẬN

- Giảm kích thước ảnh

  a=imread('c:/SDC16361.JPG');

  b=a(1:10:end,1:10:end,:);

  imshow(b)
- Lật ảnh theo chiều dọc (flip vertical)

  b=b(end:-1:1,:,:);

  imshow(b)
- Lật ảnh theo chiều ngang (flip horizontal)

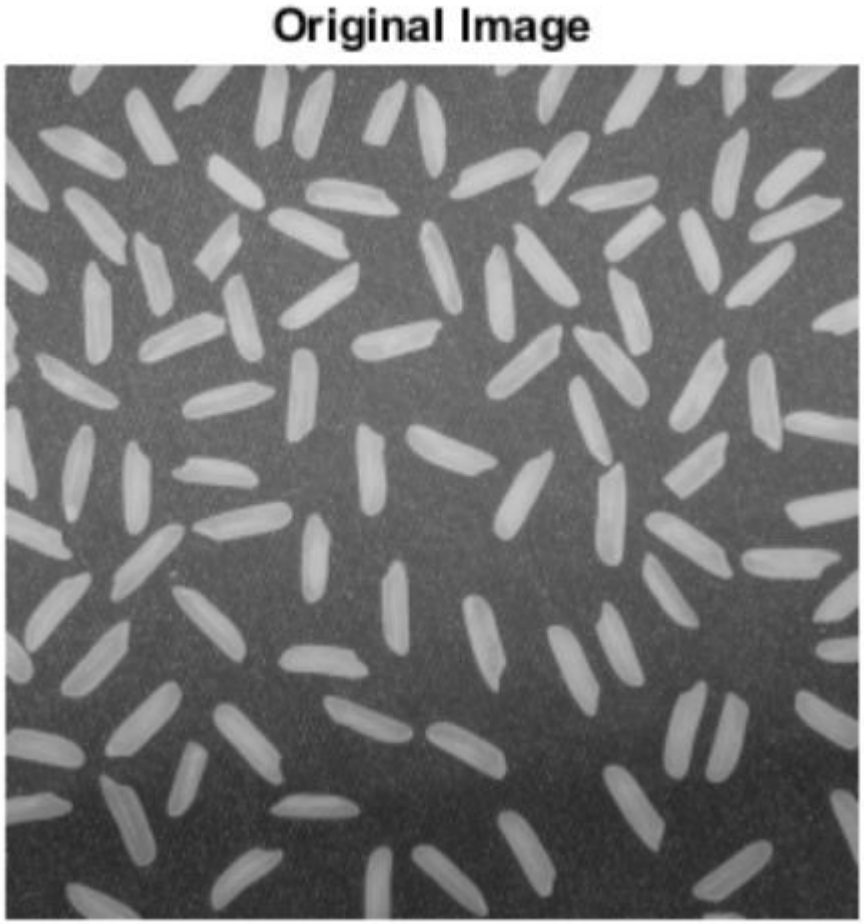
  b=b(:,end:-1:1,:);

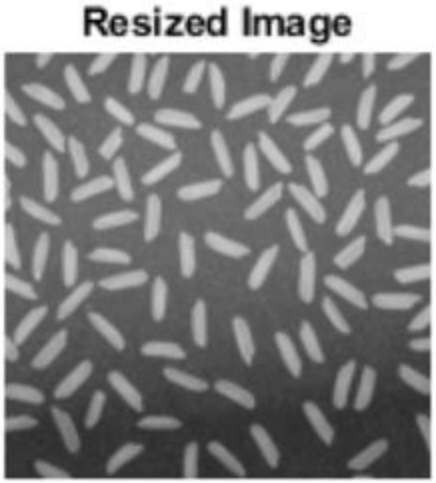
  imshow(b)

Zoom ảnh

- B= imresize(A,scale) returns image B that is scale times the size of A. The input image A can be a grayscale, RGB, or binary image. If A has more than two dimensions, imresize only resizes the first two dimensions. If scale is in the range [0, 1], B is smaller than A. If scale is greater than 1, B is larger than A. By default, imresize uses bicubic interpolation.
- B= imresize(A,[numrows numcols]) returns image B that has the number of rows and columns specified by the two-element vector [numrows numcols].

```
I = imread('rice.png');
J = imresize(I, 0.5);
figure
imshow(I)
title('Original Image')
figure
imshow(J)
title('Resized Image')
```





Method	Description
`'nearest'`	Nearest-neighbor interpolation; the output pixel is assigned the value of the pixel that the point falls within. No other pixels are considered.
`'bilinear'`	Bilinear interpolation; the output pixel value is a weighted average of pixels in the nearest 2-by-2 neighborhood
`'bicubic'`	Bicubic interpolation; the output pixel value is a weighted average of pixels in the nearest 4-by-4 neighborhood

Ví dụ zoom ảnh dùng nội suy nearest neighbor

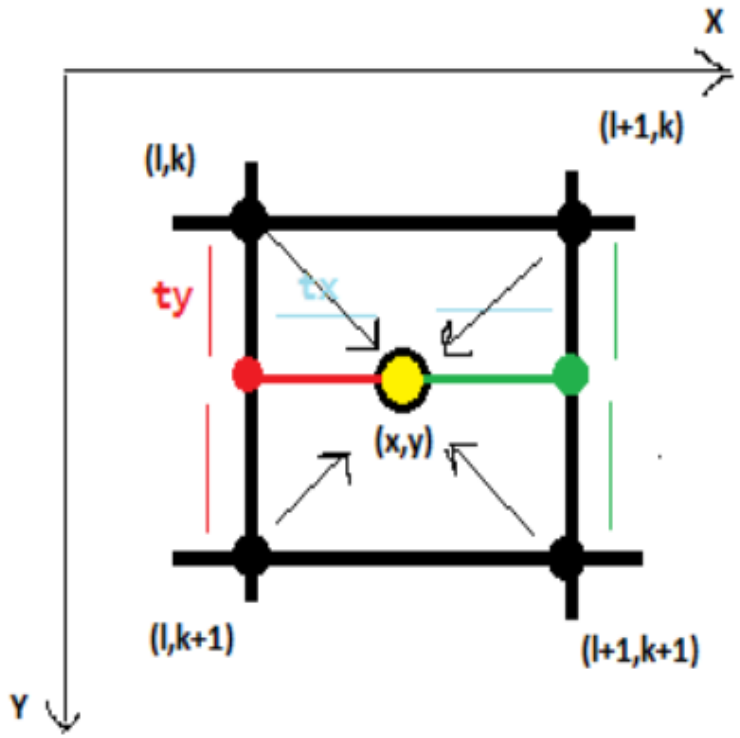
```
clear all
close all
I1 = imread('cameraman.tif');
imshow(I1)
[M1,N1,p] = size(I1)
scale=0.5;
M2 = round(M1*scale);
N2 = round(N1* scale);
```

Ví dụ zoom ảnh dùng nội suy nearest neighbor

```
I2 = zeros([M2 N2 p], class(I1)); % Allocate output image
for x=1:N2
for y=1:M2
% We'll just pick the nearest neighbor to (v,w)
I2(y,x,:) = I1(round(y/scale),round(x/scale),:);
end
end
figure
imshow(I2)
```

Thuật toán bilinear

- Cường độ ảnh là trung bình cường độ 4 điểm chung quanh

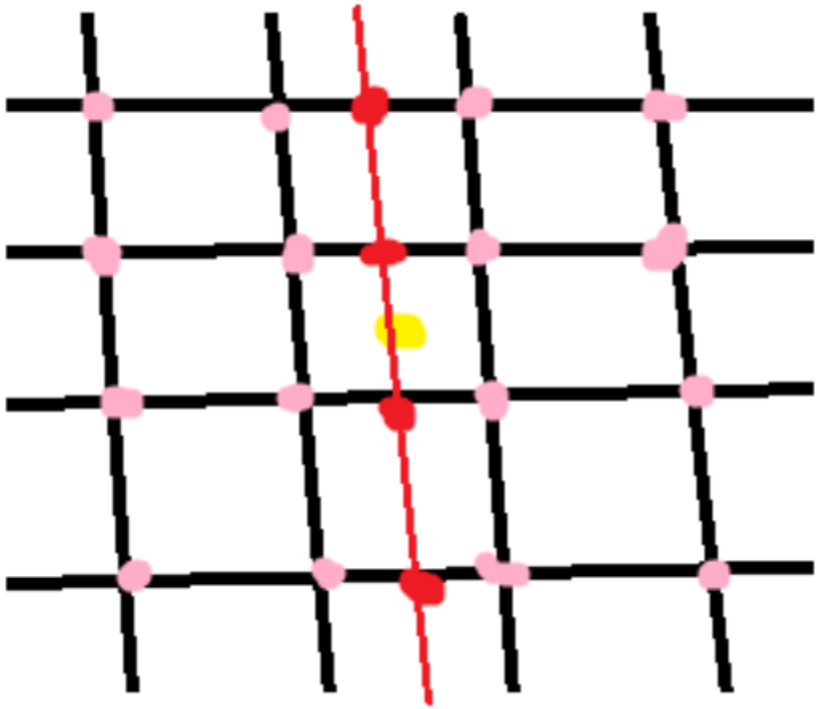



```
out(x,y) = (1-tx)*(1-ty)*img(l,k) +
           (1-tx)*ty*img(l,k+1)   +
           tx*(1-ty)*img(l+1,k)   +
           tx*ty*img(l+1,k+1)

where:
   tx = x - l
   ty = y - k
```

Thuật toán bicubic

- Lấy trung bình 16 điểm chung quanh



CROP ẢNH

```
%  This program crops a rectangular portion of a given image
%  Submitted By : Chiranjit Bordoloi & Hemashree Bordoloi
% Get Image
  clc; close all; clear all;          %clean board
  a = imread('e:\baigiang\computer vision\SDC16361.JPG');     %read image
  [m n]= size(a);                  %get no of rows and column of the image matrix
  imshow(a)                        %display original image hoặc image(a)
  % Crop Image Using Submatrix Operation
  [y,x] = ginput(2);                  %select two cursor points
  r1 = x(1,1); c1 = y(1,1);   %get first cursor point = first corner of the rectangle
  r2 = x(2,1); c2 = y(2,1);  %get second cursor point = second corner of the
    rectangle
  b = a(r1:r2,c1:c2,:); figure;;            %create the sub-matrix
imshow(b)                          %display croped image
```

CẮT ẢNH

- Dùng hàm m:

```
function s=subim1(f,m,n,rx,cy)
rowhigh=rx+m-1;
colhigh=cy+n-1;
s=f(rx:rowhigh,cy:colhigh,:);

a=imread('e:\baigiang\computer vision\SDC16361.JPG');
s= subim1(a,200,500,100,100);
figure;
Imshow(s);
```

- Dùng hàm imcrop(I,[XMIN YMIN WIDTH HEIGHT]);

- J= imcrop(I,rect) crops the image I according to rect, which specifies the size and position of the crop rectangle as [xmin ymin width height], in terms of spatial coordinates. The cropped image includes all pixels in the input image that are completely or partially enclosed by the rectangle.

```
>>I = imread('circuit.tif');
%Crop image, specifying crop rectangle.
>>I2 = imcrop(I,[75 68 130 112]);
%Display original image and cropped image.
>>subplot(1,2,1);imshow(I);
>>title('Original Image');subplot(1,2,2);
>>imshow(I2);title('Cropped Image');
```

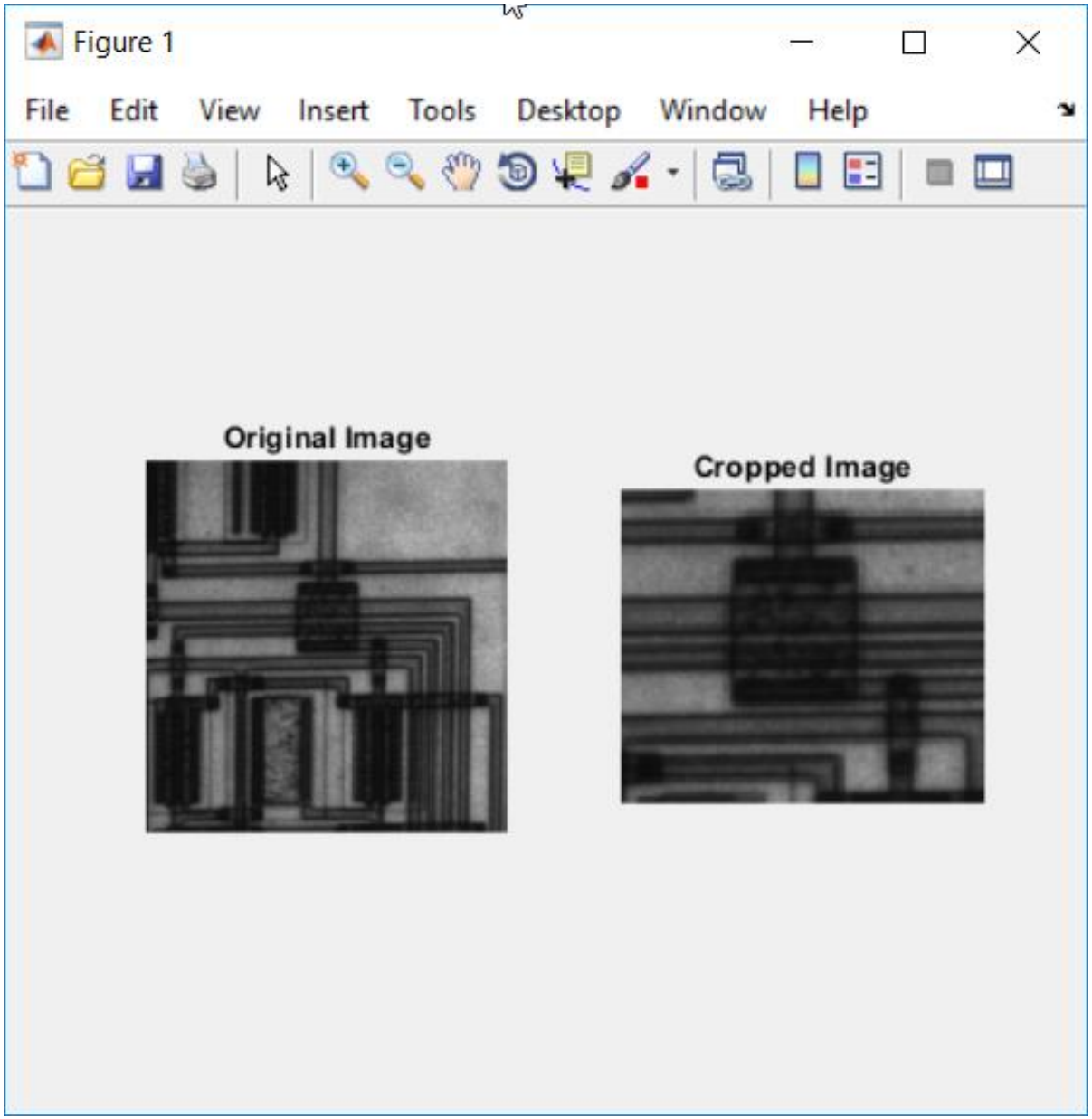


Figure 1
File Edit View Insert Tools Desktop Window Help
Original Image
Cropped Image

CHÈN CHỮ SỐ VÀO ẢNH

- Mở ảnh dùng hàm I=imread(‘tên ‘)
- Dùng hàm insertText chèn chữ số vào, khai báo vị trí, kích thước font, độ mờ…
- Lưu ảnh dùng hàm imwrite
- Ví dụ chèn chữ và số

```
I = imread('peppers.png');
%Define the ( x,y ) position for the text and the value.
position =  [1 50;200 60];
value = [399  pi];
RGB = insertText(I,position,value);
RGB = insertText(RGB,[300 50], 'INSTEXT');
figure
imshow(RGB),title(‘Insert Numeric and Text');
imwrite(RGB,‘d:/instext.jpg');
```

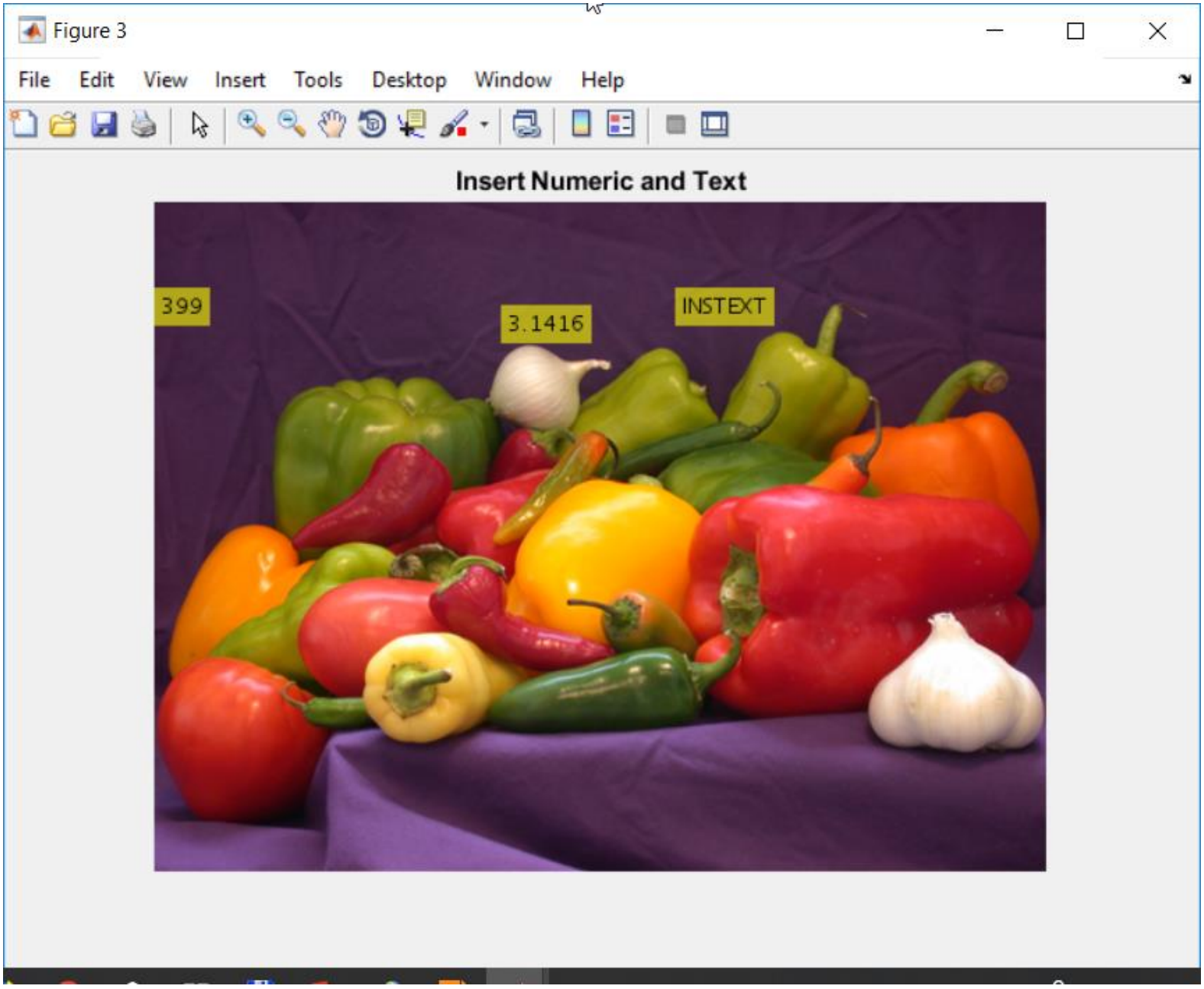


Figure 3
File Edit View Insert Tools Desktop Window Help
Insert Numeric and Text
399
3.1416
INSTEXT

Quay ảnh

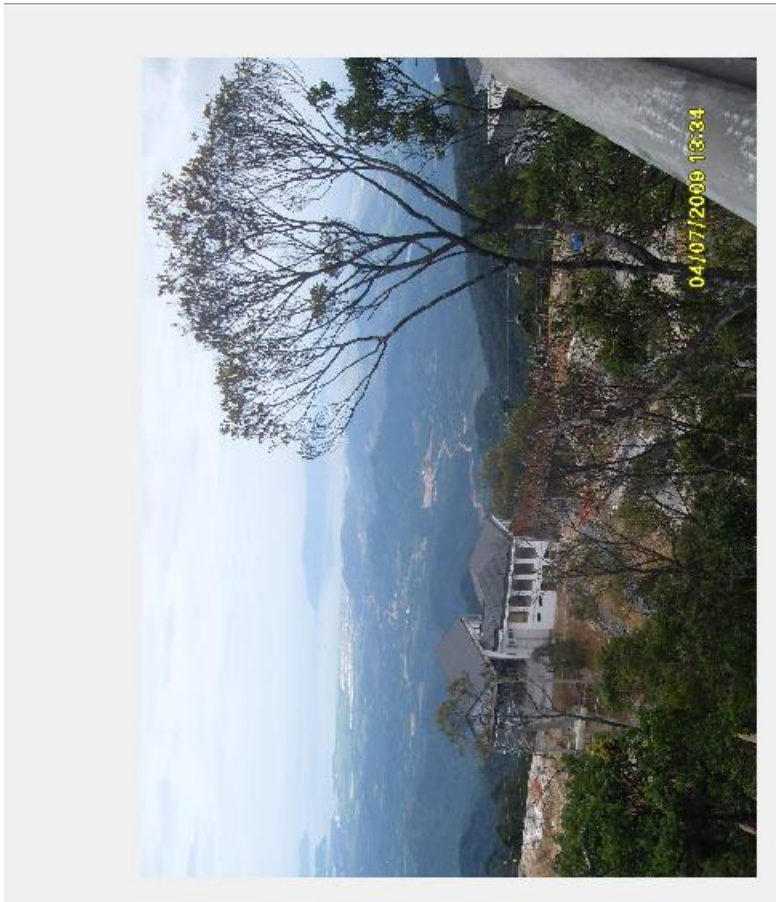
- Quay ảnh 90^o

```
I=imread('c:/SDC16361.JPG')
%Transpose
It=permute(I, [2 1 3]);
%Flip vertical
Irot=It(end:-1:1,:,: );
imshow(Irot)
```

- Gọi (x1,y1) toa độ điểm ảnh, sau khi quay góc θ, tọa độ mới là

$x2 = x1\cos d\,\theta - y1\sin d\,\theta$

$y2 = x1\sin d\,\theta + y1\cos d\,\theta$



QUAY ẢNH

- B = imrotate(A,ANGLE) rotates image A by ANGLE degrees in a counterclockwise direction around its center point. To rotate the image clockwise, specify a negative value for ANGLE. imrotate makes the output image B large enough to contain the entire rotated image. imrotate uses nearest neighbor interpolation, setting the values of pixels in B that are outside the rotated image to 0 (zero).

- B = imrotate(A,ANGLE,METHOD) rotates image A, using the interpolation method specified by METHOD. METHOD is a string that can have one of the following values. The default value is enclosed in braces ({}).

{'nearest'} Nearest neighbor interpolation

'bilinear' Bilinear interpolation

'bicubic' Bicubic interpolation. Note: This interpolation method can produce pixel values outside the original range

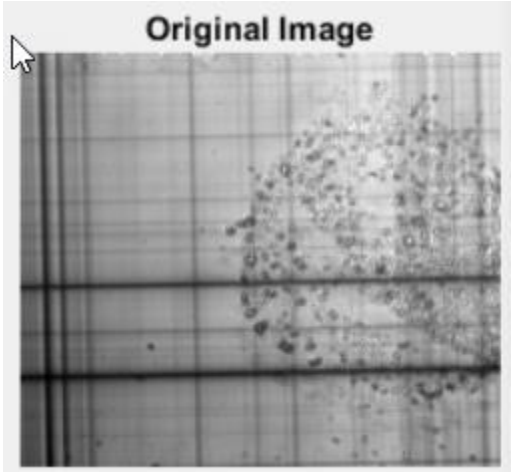
QUAY ẢNH

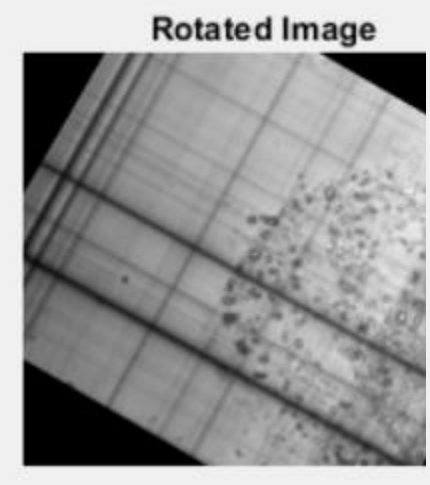
- B = imrotate(A,ANGLE,METHOD,BBOX) rotates image A, where BBOX specifies the size of the output image B. BBOX is a text string that can have either of the following values. The default value is enclosed in braces ({}).

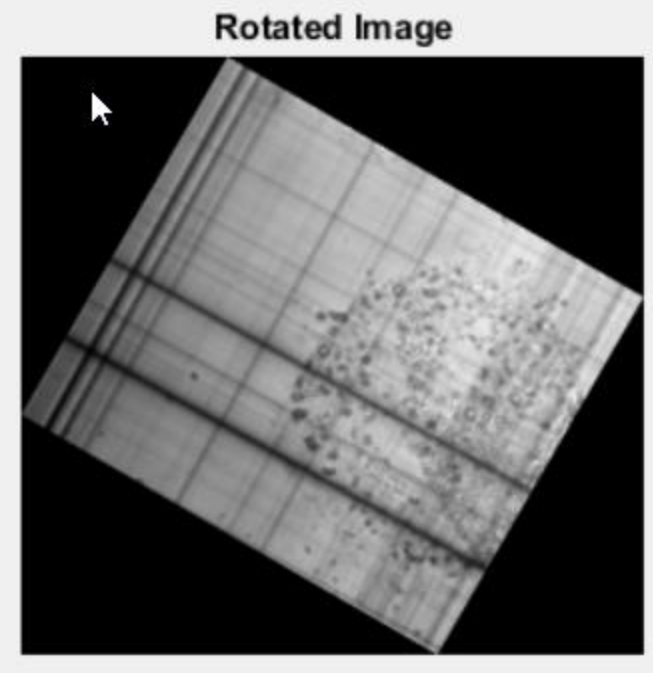
{'loose'} Make output image B large enough to contain the entire rotated image. B is generally larger than A.

'crop' Make output image B the same size as the input image A, cropping the rotated image to fit.

```
I = fitsread('solarspectra.fts');
I = rescale(I);
figure
imshow(I)
title('Original Image')
J = imrotate(I,-30,'bilinear','crop');
%J = imrotate(I,-30,'bilinear',‘loose');
figure
imshow(J)
title('Rotated Image')
```







Quay ảnh không dùng imrotate

- Nếu muốn viết một m file quay ảnh không dùng hàm imrotate, tham khảo đoạn code sau

```
image = imread('peppers.png');
figure(1), clf, hold on
subplot(1,2,1)
imshow(image);
degree = 45;
switch mod(degree, 360)
    % Special cases
    case 0
        imagerot = image;
```

Quay ảnh không dùng imrotate

```
case 90
    imagerot = rot90(image);
  case 180
    imagerot = image(end:-1:1, end:-1:1);
  case 270
    imagerot = rot90(image(end:-1:1, end:-1:1));
  % General rotations
  otherwise
    % Convert to radians and create transformation
matrix
    a = degree*pi/180;
```

Quay ảnh không dùng imrotate

```
R = [+cos(a) +sin(a); -sin(a) +cos(a)];
 % Figure out the size of the transformed image
 [m,n,p] = size(image);
 dest = round( [1 1; 1 n; m 1; m n]*R );
 dest = bsxfun(@minus, dest, min(dest)) + 1;
  % dest = dest- min(dest)+ 1; %Matlab 2016 ->
 imagerot = zeros([max(dest) p],class(image));
 % Map all pixels of the transformed image to the original image
       for ii = 1:size(imagerot,1)
          for jj = 1:size(imagerot,2)
```

Quay ảnh không dùng imrotate

```
 source = ([ii jj]-dest(1,:))*R.';
if all(source >= 1) && all(source <= [m n])
 % Get all 4 surrounding pixels
C = ceil(source);
 F = floor(source);
% Compute the relative areas
    A = [...
          ((C(2)-source(2))*(C(1)-source(1))),...
          ((source(2)-F(2))*(source(1)-F(1)));
          ((C(2)-source(2))*(source(1)-F(1))),...
           ((source(2)-F(2))*(C(1)-source(1)))];
```

Quay ảnh không dùng imrotate

```
% Extract colors and re-scale them relative to area
   cols = bsxfun(@times, A,
double(image(F(1):C(1),F(2):C(2),:)));
% Assign
   imagerot(ii,jj,:) = sum(sum(cols),2);
                    end
          end
       end
end
subplot(1,2,2)
imshow(imagerot);
```

Dời ảnh

- B = imtranslate(A,TRANSLATION) translates image A by a translation vector TRANSLATION. TRANSLATION is of the form [TX TY] for 2-D inputs, and [TX TY TZ] for 3-D inputs. If TRANSLATION is a two-element vector and A has more than two dimensions, a 2-D translation is applied to A one plane at a time. TRANSLATION can be fractional.
- [B,RB] = imtranslate(A,RA,TRANSLATION) translates the spatially referenced image defined by A and RA by a translation vector TRANSLATION. TRANSLATION is in the world coordinate system. The output is a translated spatially referenced image defined by B and RB.

Dời ảnh

```
I =
imread('cameraman
.tif');
J = imtranslate(I,[15,
25]);
%unclipped
K =
imtranslate(I,[15,
25],'OutputView','full
');
```

Dời ảnh

- B = imtranslate(A,TRANSLATION,METHOD) translates image A, using the interpolation method specified by METHOD. METHOD is a string that can have one of the following values. The default value is enclosed in braces ({}).

'nearest' Nearest neighbor interpolation

{'linear'} Linear interpolation

'cubic' Cubic interpolation. Note: This interpolation method can produce pixel values outside the original range.

BLEND IMAGE

C= imfuse(A,B) creates a composite image from two images, A and B. If A and B are different sizes, imfuse pads the smaller dimensions with zeros so that both images are the same size before creating the composite. The output, C, is a numeric matrix containing a fused version of images A and B.

[C RC] = imfuse(A,RA,B,RB) creates a composite image from two images, A and B, using the spatial referencing information provided in RA and RB. The output RC defines the spatial referencing information for the output fused image C.

C = imfuse(___,method) uses the algorithm specified by method.

C = imfuse(___,Name,Value) specifies additional options with one or more Name,Value pair arguments, using any of the previous syntaxes.

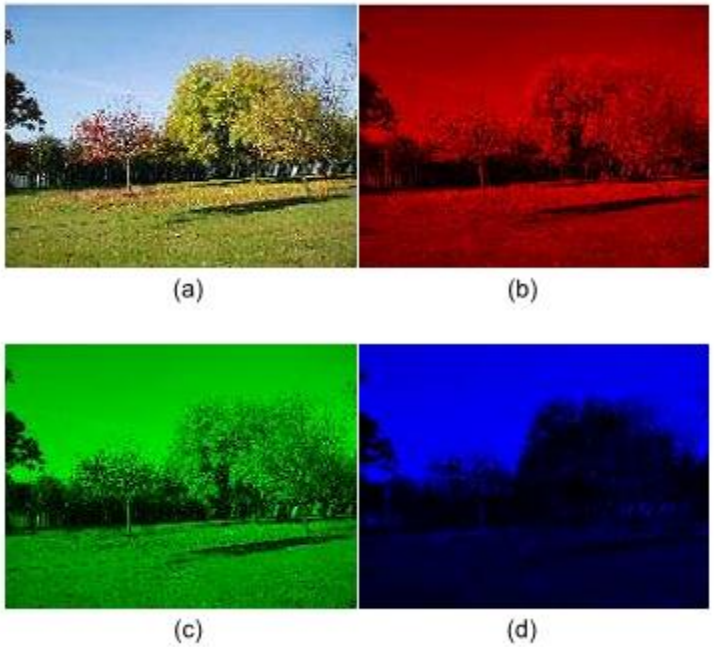
Method	Description
`'falsecolor'`	Creates a composite RGB image showing `A` and `B` overlaid in different color bands. Gray regions in the composite image show where the two images have the same intensities. Magenta and green regions show where the intensities are different. This is the default method.
`'blend'`	Overlays `A` and `B` using alpha blending.
`'checkerboard'`	Creates an image with alternating rectangular regions from `A` and `B`.
`'diff'`	Creates a difference image from `A` and `B`.
`'montage'`	Puts `A` and `B` next to each other in the same image.

Intensity scaling option, specified as one of the following values:

`'independent'`	Scales the intensity values of `A` and `B` independently when `C` is created.
`'joint'`	Scales the intensity values in the images jointly as if they were together in the same image. This option is useful when you want to visualize registrations of monomodal images, where one image contains fill values that are outside the dynamic range of the other image.
`'none'`	No additional scaling.

HỆ MÀU

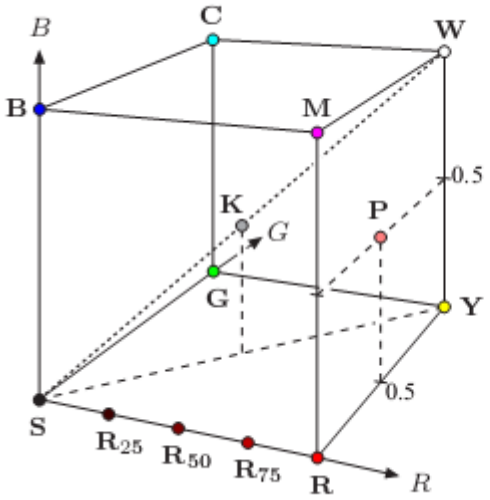
ẢNH MÀU



•Màu sắc là kết hợp các màu cơ bản phụ thuộc cơ chế nhìn màu của mắt
•Mô hình RGB dùng các màu cơ bản là Red (Đỏ), Green (Lá cây, Lục) , Blue (Xanh) thường dùng cho TV
•Mô hình CMY dùng các màu cơ bản là Cyan (lục lam, xanh cô ban, xanh ngọc) , Magenta (đỏ tươi, đỏ cánh sen) Yellow (Vàng) thường dùng cho máy in màu
•Mô hình CMYK dùng thêm màu đen K

MÔ HÌNH RGB

Phối trộn màu bổ sung: thêm đỏ vào xanh lá cây tạo ra vàng; thêm vàng vào xanh lam tạo ra trắng.



RGB Value

Point	Color	R	G	B
S	Black	0.00	0.00	0.00
R	Red	1.00	0.00	0.00
Y	Yellow	1.00	1.00	0.00
G	Green	0.00	1.00	0.00
C	Cyan	0.00	1.00	1.00
B	Blue	0.00	0.00	1.00
M	Magenta	1.00	0.00	1.00
W	White	1.00	1.00	1.00
K	50% Gray	0.50	0.50	0.50
R_{75}	75% Red	0.75	0.00	0.00
R_{50}	50% Red	0.50	0.00	0.00
R_{25}	25% Red	0.25	0.00	0.00
P	Pink	1.00	0.50	0.50

- Các màu cơ bản không định nghĩa chính xác, do đó các màn hình khác nhau cho các ảnh màu khác nhau
- Mô hình RGB biểu diễn màu như một điểm trong hình hộp đơn vị RGB, màu đen có tọa độ (0 0 0) và màu trắng là (1 1 1)
- Nếu dùng 8 bit để biểu diễn một màu cơ bàn thì màu đen là (0 0 0) và màu trắng là (255 255 255), Cyan là hỗn hợp G+B, Magenta: R+B, Yellow: R+G
- Màu RGB 24 bit (true color) dùng 8 bit cho mỗi màu cơ bản, tổng cộng khoảng 16,7 triệu màu
- RGBA dùng 32 bit, 8 bit thêm vào là kênh alpha, biểu thị độ trong suốt, 0 là hoàn toàn trong suốt, 255 là đặc, giúp tạo các ảnh chồng lên nhau

MÔ HÌNH RGB

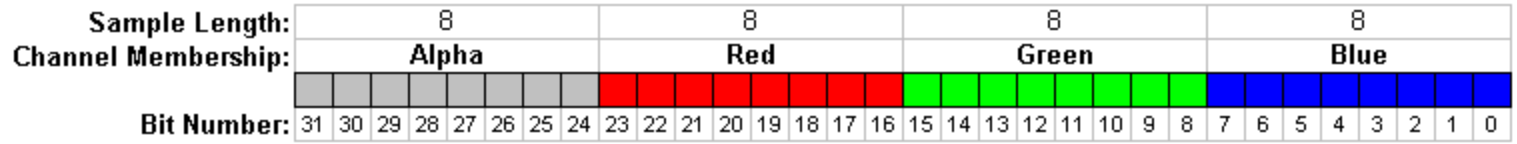
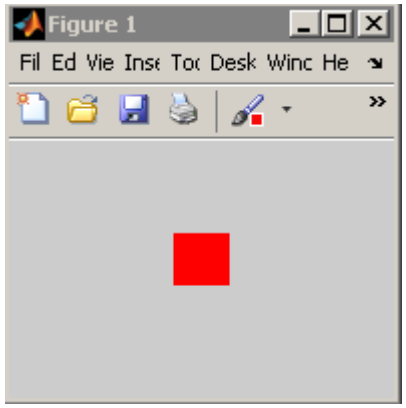
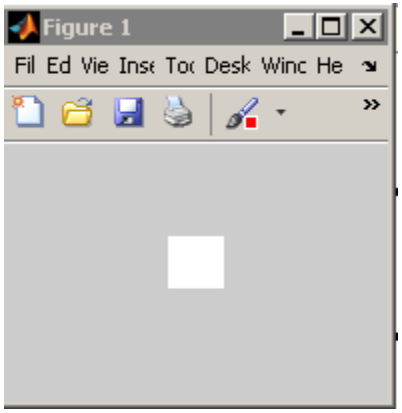


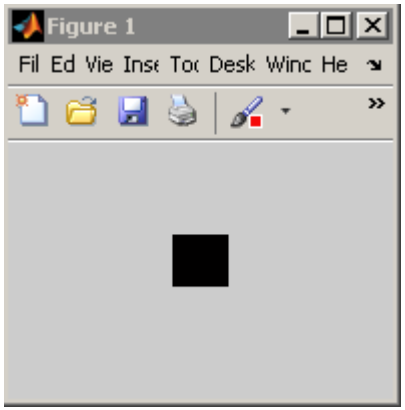
Fig.1 - An example of the most common 32bpp pixel layout

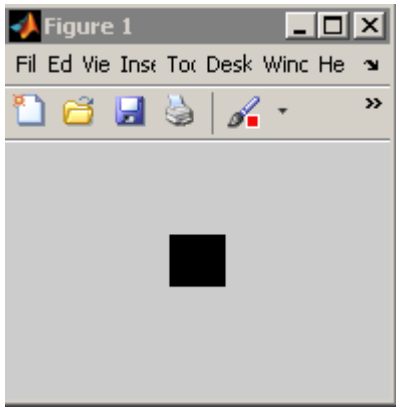
- Nếu a là ảnh thì các lệnh sau đọc các ma trận màu cơ bản RGB
  - ar = a(:,:,1);
  - ag= a(:,:,2);
  - ab= a(:,:,3);
- Các ảnh thành phần là các ảnh xám biểu thị cường độ mỗi màu cơ bản
- Ngược lại, lệnh sau tạo ảnh màu từ các màu cơ bản
  - a = cat(3,ar,ag,ab);
- Hàm rgb2gray đổi ảnh màu ra ảnh xám theo thuật toán

  0.2989 * R + 0.5870 * G + 0.1140 * B



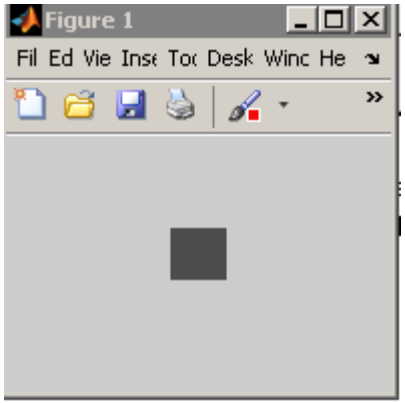






Ba thành phần màu của một hình có màu đỏ

Ảnh xám



RGB

R

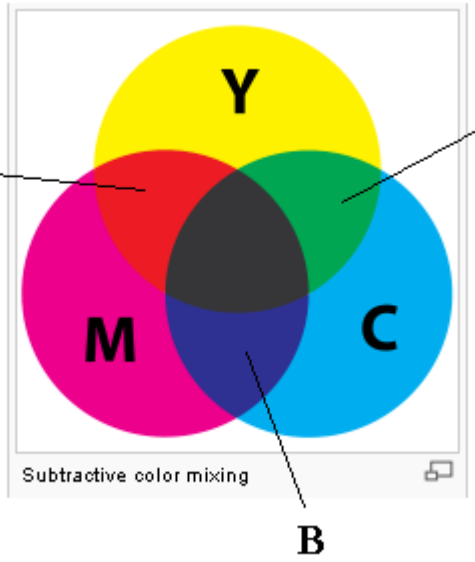
G

B

HỆ MÀU CMYK

- Hệ màu CMYK sử dụng trong in màu, sơn màu, gồm ba màu cơ bản là Cyan (xanh ngọc), Magenta (hồng cánh sen) , Yellow (vàng) và K (đen)
- Ba màu CMY kết hợp tạo ra màu đen, cần trong in ấn, còn ba màu RGB tạo ra màu trắng, cần cho màn hình, hệ m àu RGB gọi là hệ màu cộng còn hệ màu CMY gọi là h ệ màu trừ
- Hàm imcomplement trong Matlab chuyển đổi giữa hai hệ

Cyan, magenta, yellow, and key (black).



Subtractive color mixing

C=1-R
M=1-G
Y=1-B

HỆ MÀU CMYK

- Chuyển đổi giữa CMY và CMYK theo công thức

  K= min (C,M,Y)

  C=C-K

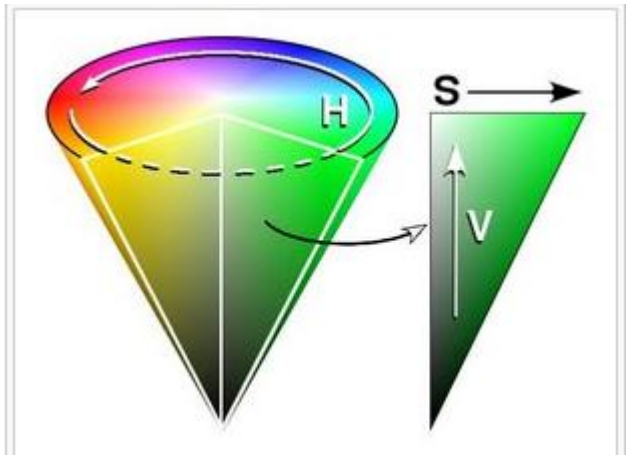
  M=M-K

  Y=Y-K

KHÔNG GIAN MÀU HSV

- Không gian màu H (Hue, màu sắc xanh đỏ) từ 0 đến 360o, S (Saturation, bão hòa) Độ bão hòa màu thể hiện độ thuần khiết của màu. Khi có độ bão hòa cao, màu sẽ sạch và rực rỡ. Khi có độ bão hòa thấp, màu sẽ đục và xỉn. Độ bão hòa thay đổi từ 0% (xám) đến 100%., V (Value) độ sáng, HSV không dùng các màu cơ bản, gần giống cảm nhận con người về màu, thường dùng trong đồ họa máy tính
- Không gian màu HSV có tọa độ trụ, phát xuất từ không gian RGB với gốc là điểm đen, trục hình trụ là đường nối điểm đen và trắng, điểm đen nối các màu cơ bản tạo hình nón, Hue tương ứng với bước sóng của ánh sáng, biểu thị theo góc từ 0^0 đến 360^0 phát xuất từ điểm R; S biểu thị độ tinh khiết của màu, ví dụ Red có S là 100% còn màu hồng có S kém hơn vì pha trộn thêm màu trắng, V biểu thị độ sáng tối, tính theo trục hình nón, đen có V=0 còn trắng có V=1. Thành phần H và S không phụ thuộc độ chiếu sáng.

KHÔNG GIAN MÀU HSV



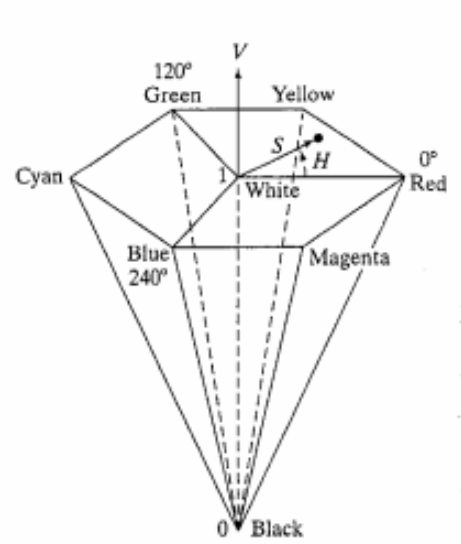
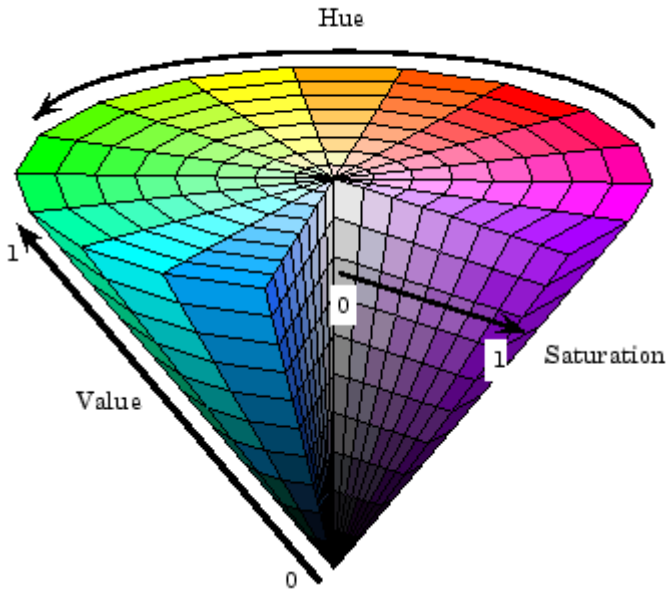


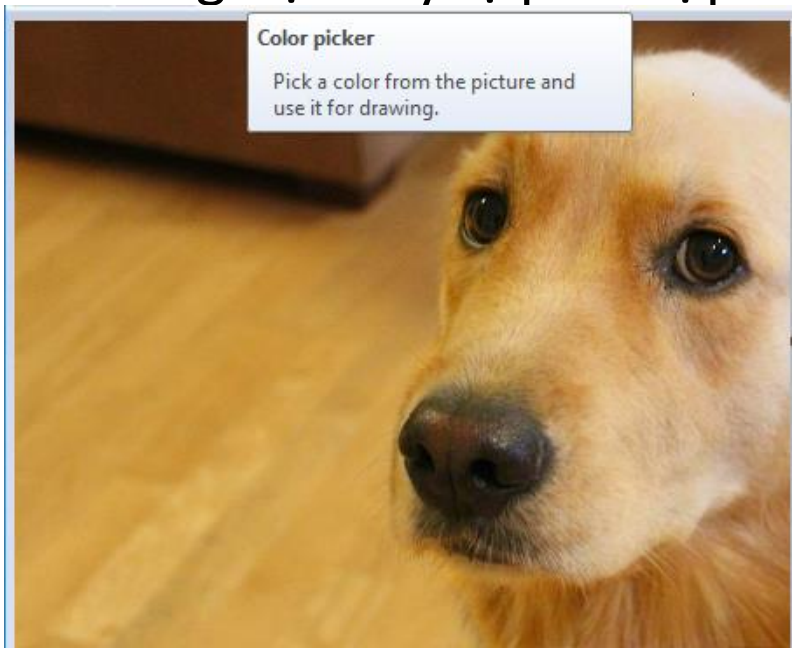
Illustration of the HSV Color Space

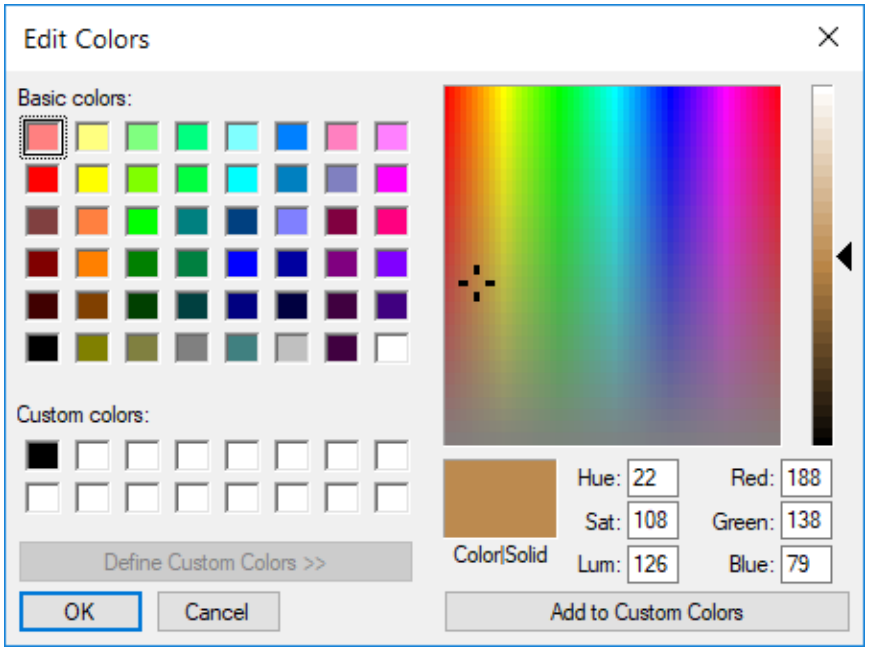


Matlab dùng hàm rgb2hsv và hsv2rgb chuyển đổi qua lại giữa hai hệ màu

Gía trị RGB và HSV

- Muốn tìm giá trị các thành phần của ảnh ta có thể dùng MS Paint. Nạp ảnh vào Ms Paint, dùng Pick Color và con chuột để chọn điểm ảnh. Chọn Edit Colors đọc giá trị HSL và RGB. Chú ý là giá trị của HSV trong Paint đi từ 0 đến 240 còn giá trị tương ứng trong Matlab là 0 đến 1cho S, V và 0..360 cho H, vậy cần dùng hệ số tỷ lệ phù hợp.





CHUYỂN ĐỔI RGB HSV

Max= MAX(R,G,B)

Min=MIN(R,G,B)

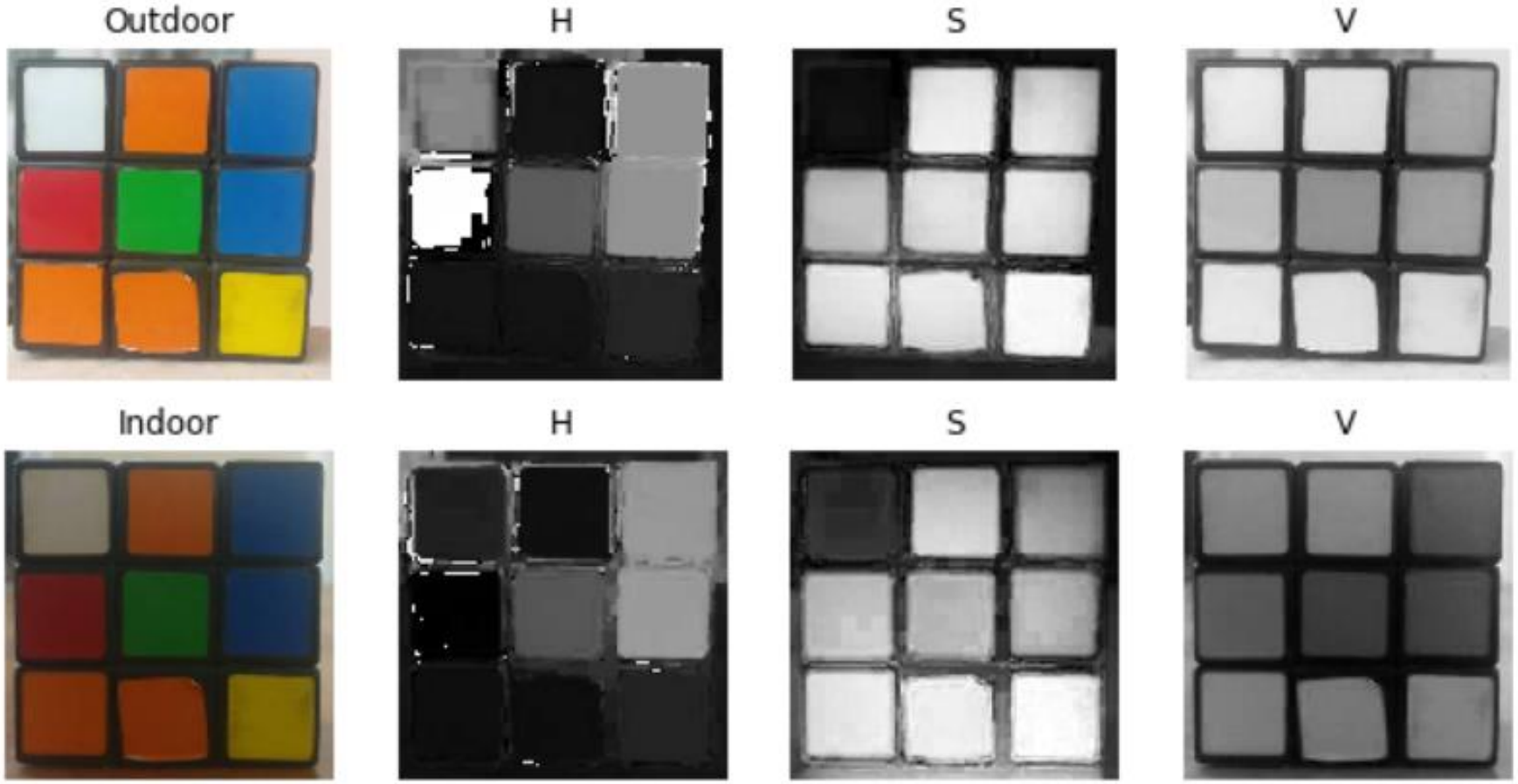
C=Max-Min

Nếu Max=Min(R,G,B) đó là ảnh xám, H=0, S=0

S=C/Max

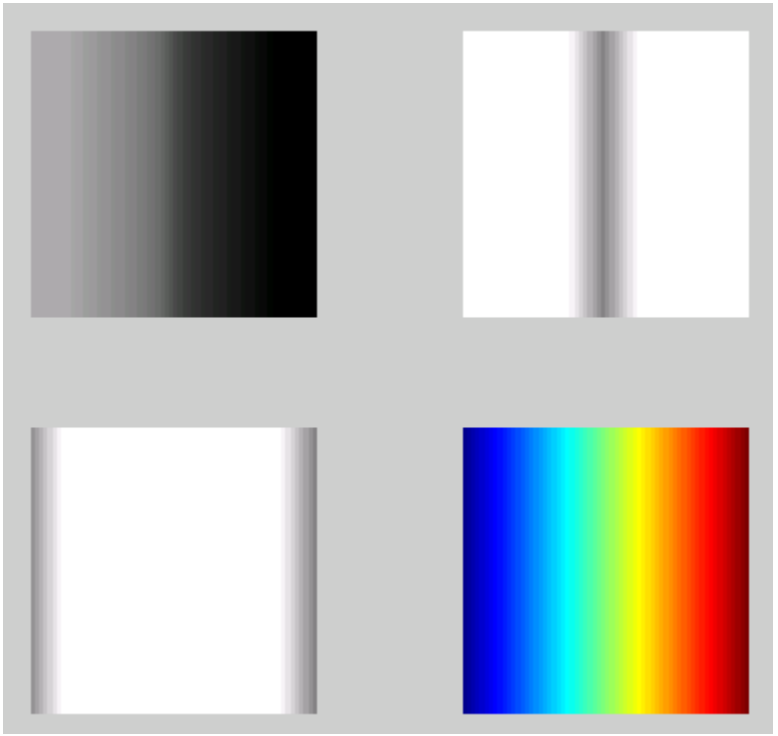
V=Max/255

```
If (R == Max)
    { H = (G - B) / C;
      if(H < 0.0)
      { H += 6.0; }
    }
else if(G == Max)
{ H = ((B - R) / C) + 2.0; }
 else //B == Max
 {H = ((R - G) / C) + 4.0;
 }
H *= 60.0; //H=0..360o
```


Outdoor
H
S
V
Indoor
H
S
V

KHÔNG GIAN MÀU HSV

```
RGB=reshape(ones(64,1)*reshape
    (jet(64),1,192),[64,64,3]);
HSV=rgb2hsv(RGB);
 H=HSV(:,:,1);
S=HSV(:,:,2);
V=HSV(:,:,3);
subplot(2,2,1), imshow(H)
subplot(2,2,2), imshow(S)
subplot(2,2,3), imshow(V)
subplot(2,2,4), imshow(RGB)
```

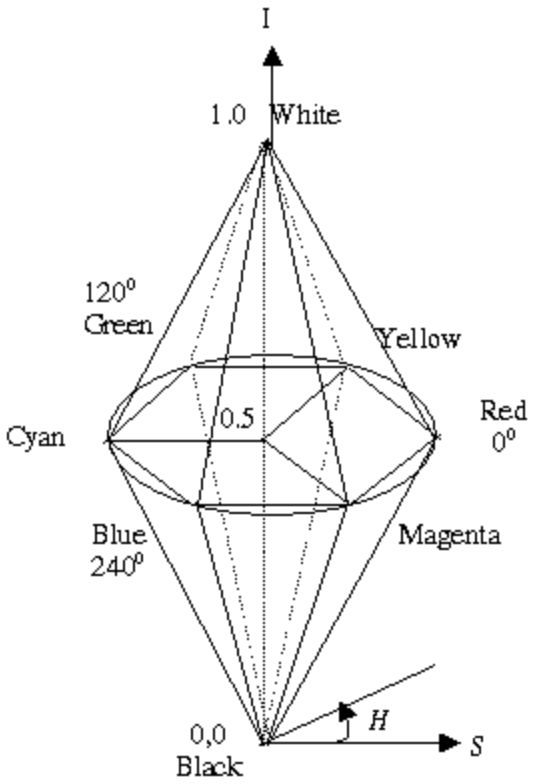


CHUYỂN ĐỔI RGB HSV

CHUYỂN ĐỔI RGB HSV

KHÔNG GIAN MÀU HSI

- Không gian màu HSI tương tự HSV trong đó giá trị I là cường độ sáng
- Không gian màu HIS giúp cho việc xử lý ảnh dễ dàng hơn, tương tự cách con người nhìn, không cần biết thành phần màu cơ bản



Mô hình hình nón kép không gian màu HSI

KHÔNG GIAN MÀU HSI

- Chuyển đổi RGB –HSI

$$I = \frac{1}{3}(R+G+B)$$

$$S = 1 - \frac{3}{R+G+B}[\min(R,G,B)]$$

$$H = \cos^{-1}\left[\frac{\frac{1}{2}[(R-G)+(R-B)]}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right]$$

Nếu B>G, $H=360^0 - H$

- Chuyển đổi HIS – RGB: tùy thuộc góc H

KHÔNG GIAN MÀU HSI

- Đoạn RG (0^0 <H< 120^0)

$$B = I(1 - S)$$

$$R = I\left[1 + \frac{S\cos H}{\cos(60^\circ - H)}\right]$$

$$G = 3I - (R + B)$$

- Đoạn GB (120^0 <H < 240^0)

$$H = H - 120^0$$

$$R = I(1 - S)$$

$$G = I\left[1 + \frac{S\cos H}{\cos(60^\circ - H)}\right]$$

$$B = 3I - (R + G)$$

- Đoạn BR

$$H = H - 240^\circ$$

$$G = I(1 - S)$$

$$B = I\left[1 + \frac{S\cos H}{\cos(60^\circ - H)}\right]$$

$$R = 3I - (G + B)$$

KHÔNG GIAN MÀU HSI

- Hàm RGB2HSI, HSI2RGB chuyển đổi giữa RGB và HSI

```
function hsi= rgb2hsi(rgb)
rgb=im2double(rgb);
r=rgb(:,:,1);
g=rgb(:,:,2);
b=rgb(:,:,3);
num = 0.5*((r-b)+r-g);
den=sqrt((r-g).^2+(r-b).*(g-b));
theta=acos(num./(den+eps));
H=theta;
H(b>g)=2*pi-H(b>g);
H=H/2*pi;
num = min(min(r,g),b);
den = r+g+b;
den(den==0) = eps;
S=1-3.*num./den;
H(S==0) = 0;
I = (r+g+b)/3;
hsi = cat(3,H,S,I);
```

KHÔNG GIAN MÀU HSI

KHÔNG GIAN MÀU HSI

YCrCb Color-Space

Hàm rgb2ycbcr ycbcr2rgb

Y là độ sáng (ảnh xám), Cr Cb là màu, thường dùng khi tách màu không phụ thuộc độ chiếu sáng

$$Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B$$

$$Cr \leftarrow (R - Y) \cdot 0.713 + delta$$

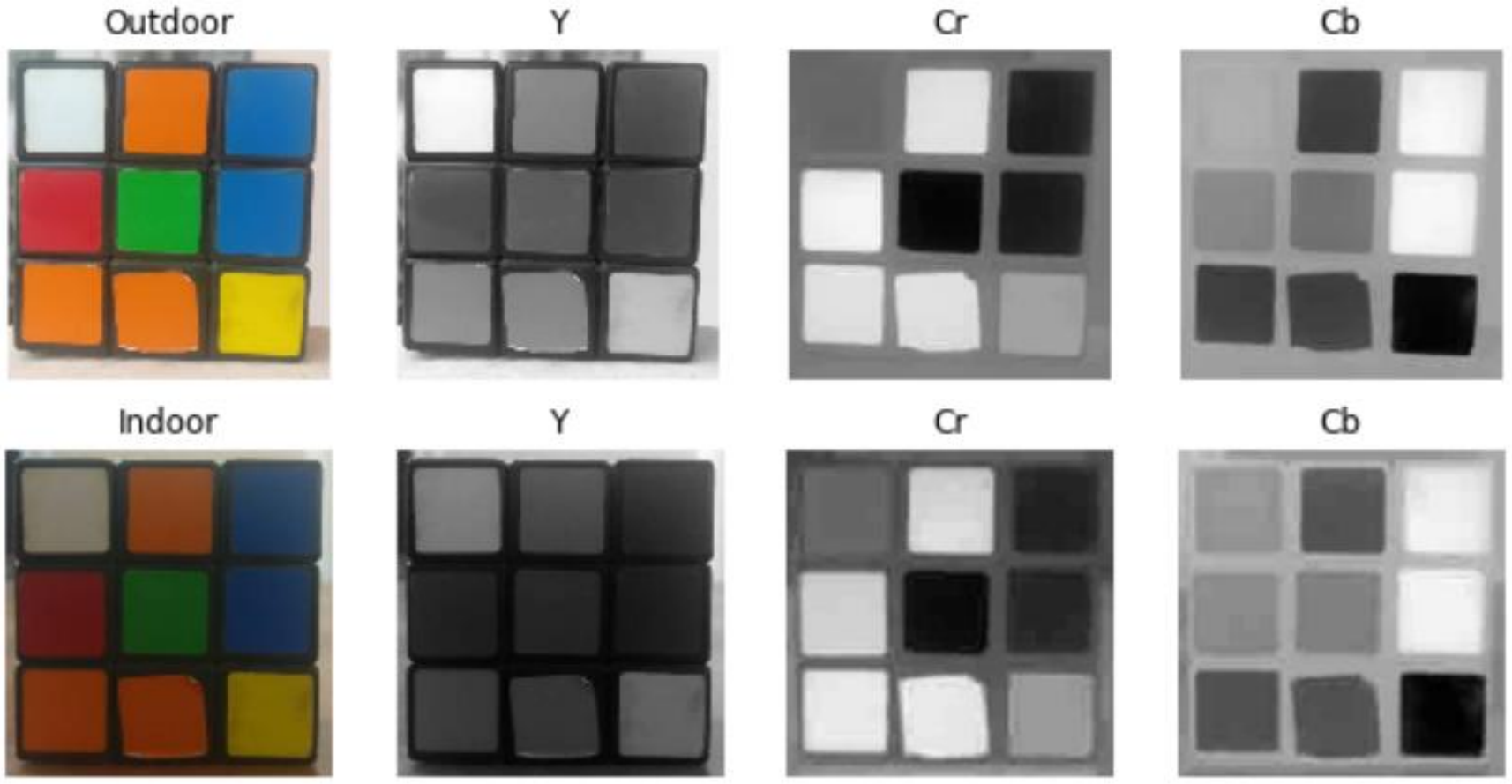
$$Cb \leftarrow (B - Y) \cdot 0.564 + delta$$

$$R \leftarrow Y + 1.403 \cdot (Cr - delta)$$

$$G \leftarrow Y - 0.714 \cdot (Cr - delta) - 0.344 \cdot (Cb - delta)$$

$$B \leftarrow Y + 1.773 \cdot (Cb - delta)$$

$$delta = \begin{cases} 128 & \text{for 8-bit images} \\ 32768 & \text{for 16-bit images} \\ 0.5 & \text{for floating-point images} \end{cases}$$


Outdoor
Y
Cr
Cb
Indoor
Y
Cr
Cb

CÁC HÀM CHUYỂN ĐỔI KHÔNG GIAN MÀU

- rgb2gray: đổi ảnh màu rgb sang ảnh xám
- Im2bw(src,level): đổi ảnh màu hay xám ra ảnh nhị phân, level là mức ngưỡng, nếu không có tham số level thì mức ngưỡng là 0.5. Ngưỡng được tính bằng hàm graythresh

CÁC HÀM CHUYỂN ĐỔI LOẠI ẢNH

gray2ind	Convert grayscale or binary image to indexed image
ind2gray	Convert indexed image to grayscale image
mat2gray	Convert matrix to grayscale image
rgb2gray	Convert RGB image or colormap to grayscale
ind2rgb	Convert indexed image to RGB image
label2rgb	Convert label matrix into RGB image
demosaic	Convert Bayer pattern encoded image to truecolor image
imquantize	Quantize image using specified quantization levels and output values
multithresh	Multilevel image thresholds using Otsu ´s method
im2bw	Convert image to binary image, based on threshold
graythresh	Global image threshold using Otsu's method
grayslice	Convert grayscale image to indexed image using multilevel thresholding
im2double	Convert image to double precision
im2int16	Convert image to 16-bit signed integers
im2java2d	Convert image to Java buffered image
im2single	Convert image to single precision
im2uint16	Convert image to 16-bit unsigned integers
im2uint8	Convert image to 8-bit unsigned integers

MATLAB FILE VIDEO

- Play video: implay(‘đường dẫn và tên file’)

```
implay
implay(filename)
implay(I)
implay(___,fps)
```

- OBJ = VideoReader (‘đường dẫn và tên file’)
- Muốn xử lý một video ta đọc từng frame, xử lý rồi phát lại

Methods:

readFrame - Read the next available frame from a video file.

hasFrame - Determine if there is a frame available to read from a video file.

getFileFormats - List of known supported video file formats.

Properties:

Name - Name of the file to be read.

Path - Path of the file to be read.

Duration - Total length of file in seconds.

CurrentTime - Location from the start of the file of the current frame to be read in seconds.

Tag - Generic string for the user to set.

UserData - Generic field for any user-defined data.

Height - Height of the video frame in pixels.

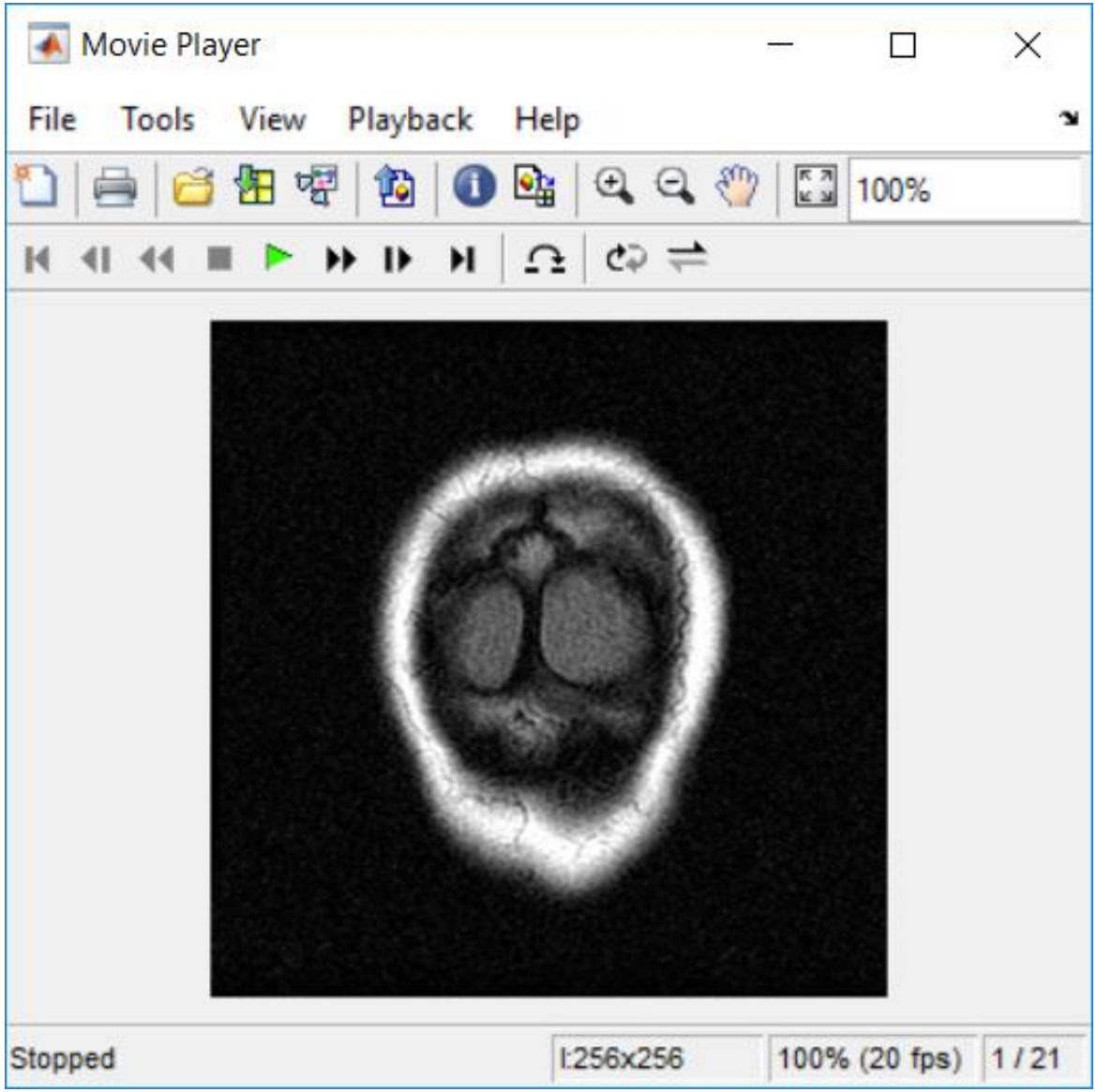
Width - Width of the video frame in pixels.

BitsPerPixel - Bits per pixel of the video data.

VideoFormat - Video format as it is represented in MATLAB.

FrameRate - Frame rate of the video in frames per second.

```
>>implay('rhinos.avi');
>>load cellsequence; implay(cellsequence,10);
>> load mristack; implay(mristack);
```


Movie Player
File
Tools
View
Playback
Help
100%
Stopped
I:256x256
100% (20 fps)
1 / 21

ĐỌC FRAME

```
% Construct a multimedia reader object
vidObj = VideoReader('xylophone.mp4');
 % Specify that reading should start at 0.5 seconds from the beginning.
       vidObj.CurrentTime = 0.5;
 % Create an axes
       currAxes = axes;
  % Read video frames until available
       while hasFrame(vidObj)
          vidFrame = readFrame(vidObj);
          image(vidFrame, 'Parent', currAxes);
          currAxes.Visible = 'off';
          pause(1/vidObj.FrameRate);
       end
```

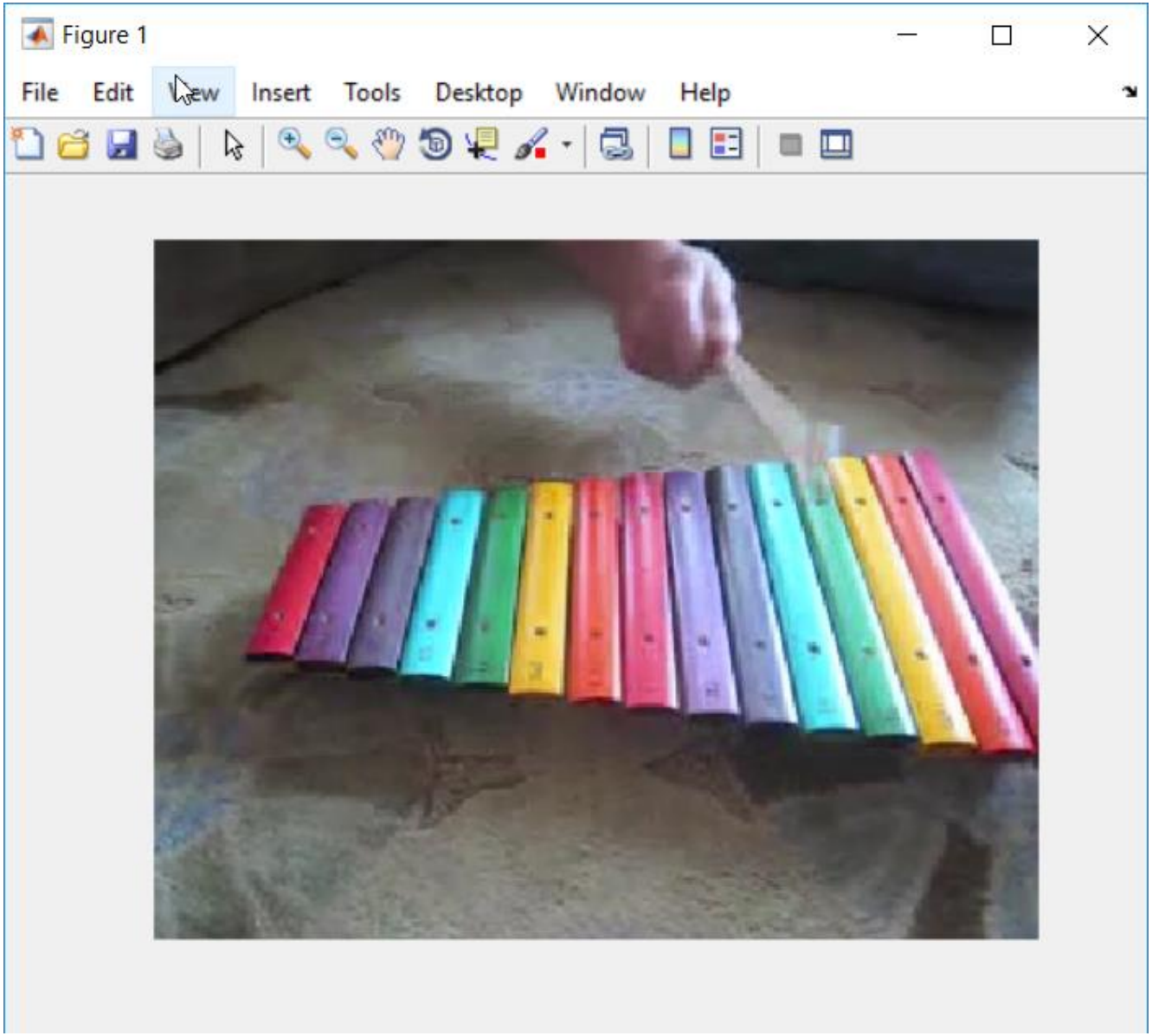
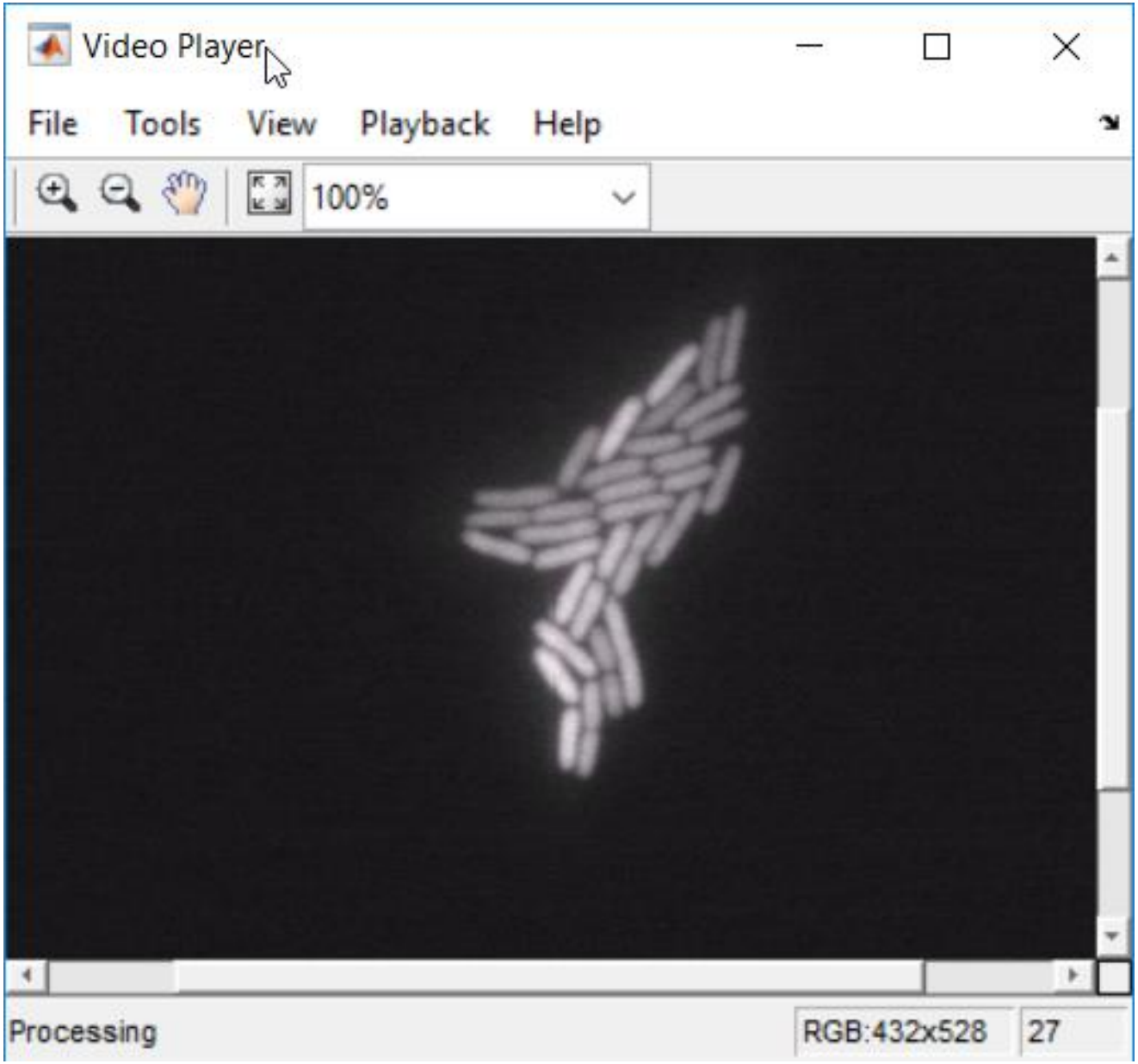


Figure 1
File
Edit
View
Insert
Tools
Desktop
Window
Help

DÙNG COMPUTERVISION TOOLBOX

```
%Load the video using a video reader object.
videoFReader = vision.VideoFileReader('ecolicells.avi');
%Create a video player object to play the video file.
videoPlayer = vision.VideoPlayer;
%Use a while loop to read and play the video frames.
while ~isDone(videoFReader)
videoFrame = videoFReader();
videoPlayer(videoFrame);
end
```


Video Player
File
Tools
View
Playback
Help
100%
Processing
RGB:432x528
27

GHI VIDEO VÀO ĐĨA

- Dùng hàm videowriter ghi file avi vào đĩa
- OBJ = VideoWriter(FILENAME) constructs a VideoWriter object to

  write video data to an AVI file that uses Motion JPEG compression.

  FILENAME is a string enclosed in single quotation marks that specifies the name of the file to create. If filename does not include the extension '.avi', the VideoWriter constructor appends the extension.
- OBJ = VideoWriter(FILENAME, PROFILE) applies a set of properties tailored to a specific file format (such as 'Uncompressed AVI') to a VideoWriter object.

PROFILE is a string enclosed in single quotation marks that describes the type of file to create.

GHI VIDEO VÀO ĐĨA

Methods:

open - Open file for writing video data.

close - Close file after writing video data.

writeVideo - Write video data to file.

getProfiles - List profiles and file format supported by VideoWriter.

Properties:

ColorChannels - Number of color channels in each output video frame.

Colormap - Numeric matrix having dimensions Px3 that contains color information about the video file. The colormap can have a maximum of

GHI VIDEO VÀO ĐĨA

256 entries of type 'uint8' or 'double'.

The entries of the colormap must integers. Each row of Colormap specifies the red, green and blue components of a single color. The colormap can be set:

- Explicitly before the call to open OR

- Using the colormap field of the FRAME struct at the time of writing the first frame. Only applies to objects associated with Indexed AVI files.

GHI VIDEO VÀO ĐĨA

CompressionRatio - Number greater than 1 indicating the tarrget ratio between the number of bytes in the input image and compressed image. Only applies to objects associated with Motion JPEG 2000 files.

Duration Scalar value specifying the duration of the file in seconds.

FileFormat - String specifying the type of file to write.

Filename - String specifying the name of the file.

FrameCount - Number of frames written to the video file.

FrameRate Rate of playback for the video in frames per second

Height Height of each video frame in pixels.The writeVideo method sets values for Height and Width based on the dimensions of the first frame.

GHI VIDEO VÀO ĐĨA

LosslessCompression - Boolean value indicating whether lossy or lossless compression is to be used. If true, any specified value for the CompressionRatio property is ignored.

MJ2BitDepth - Number of least significant bits in the input image data, from 1 to 16.

GHI VIDEO VÀO ĐĨA

Path - String specifying the fully qualified file path.

Quality - Integer from 0 through 100.

VideoBitsPerPixel - Number of bits per pixel in each output video frame.

VideoCompressionMethod - String indicating the type of video compression.

VideoFormat - String indicating the MATLAB representation of the video format.

Width - Width of each video frame in pixels.

GHI VIDEO VÀO ĐĨA

```
% Prepare the new file.
    vidObj = VideoWriter('peaks.avi');
    open(vidObj);
     % Create an animation.
    Z = peaks; surf(Z);
    axis tight
    set(gca,'nextplot','replacechildren');
     for k = 1:20
       surf(sin(2*pi*k/20)*Z,Z)
        % Write each frame to the file.
       currFrame = getframe;
       writeVideo(vidObj,currFrame);
    end
      % Close the file.
    close(vidObj);
```

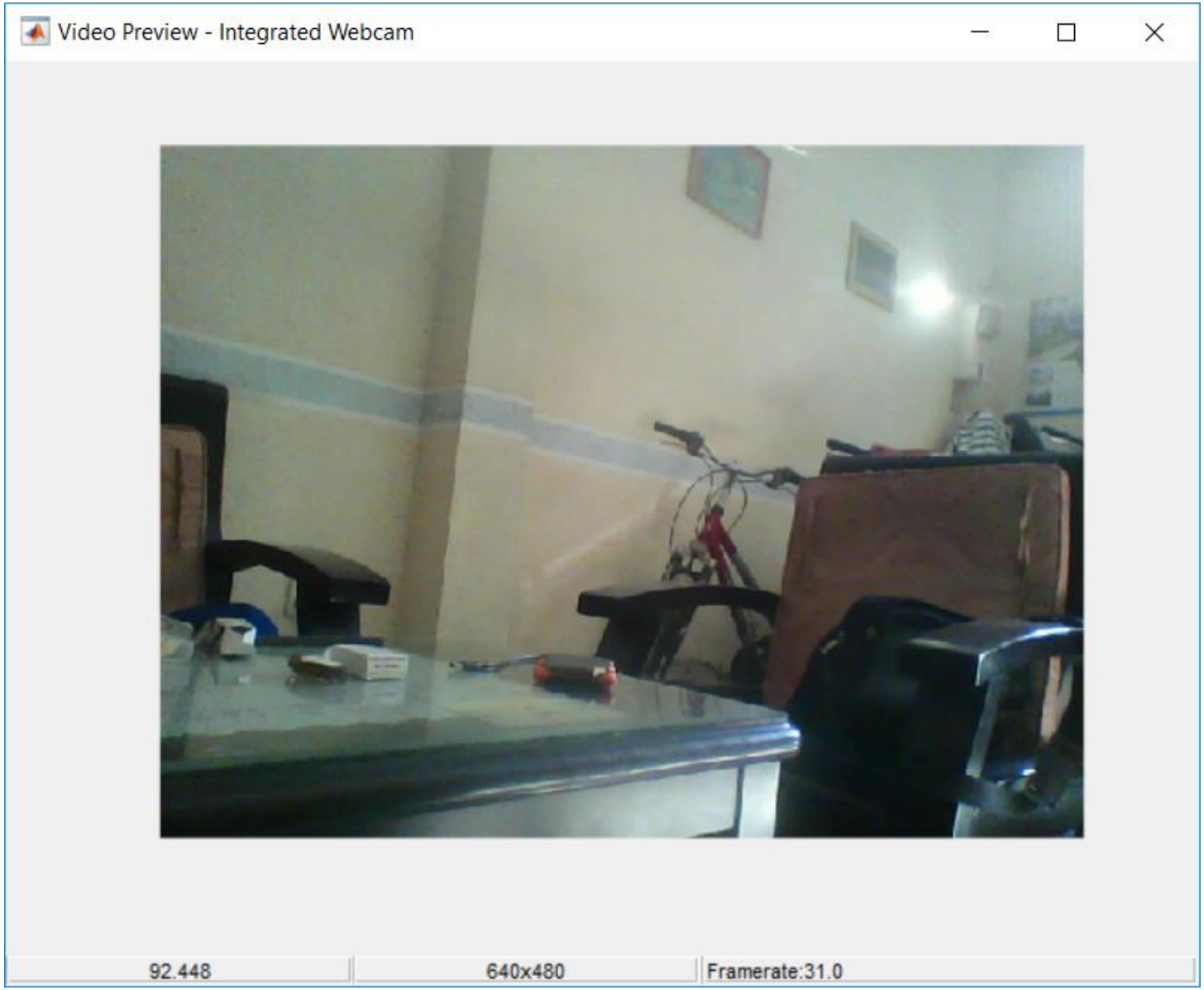
Dùng CV TOOLBOX

```
videoFReader =
    vision.VideoFileReader('viplanedeparture.mp4');
videoFWriter =
    vision.VideoFileWriter('myFile.avi','FrameRate',...
    videoFReader.info.VideoFrameRate);
%Write the first 50 frames from original file into a newly
    created AVI file.
for i=1:50 videoFrame = step(videoFReader);
    step(videoFWriter,videoFrame);
end
%Close the input and output files.
release(videoFReader);
release(videoFWriter);
```

GIAO TiẾP CAMERA DÙNG MATLAB

- Dùng hàm webcamlist tìm danh sách camera nối với máy tính, sau đó hàm webcam(chỉ số) để kết nối và hàm preview để quan sát. Kết thúc dùng closePreview

```
>> webcamlist
ans =
  2×1 cell array
    'WebcamMax Capture'
    'Integrated Webcam‘
>> cam = webcam(2); %Chọn webcam số 2
>>preview (cam) % đọc video từ camera
>> cam.AvailableResolutions
ans =
  1×8 cell array
  Columns 1 through 6
    '640x480'    '640x360'    '352x288'    '320x240'    '424x240'    '176x144‘
```


Video Preview - Integrated Webcam
92.448
640x480
Framerate:31.0

GIAO TiẾP CAMERA DÙNG MATLAB

Columns 7 through 8

'160x120' '1280x720‘

>> cam.Resolution = '320x240';

>> img = snapshot(cam); //Lấy một ảnh và hiển thị

>> imshow(img)

>> closePreview(cam) // ngừng camera

>> clear('cam'); ngắt kết nối

WEBCAM RGB TO GRAY

```
cam = webcam(2);
%preview(cam)
for idx = 1:100 % acquire 100 image
 % Acquire a single image.
rgbImage = snapshot(cam);
% Convert RGB to grayscale.
grayImage = rgb2gray(rgbImage);
% Display the image.
 imshow(grayImage);
end
clear('cam');
close
```

LOGGING WEBCAM TO DISK

```
% Connect to the webcam.
cam = webcam (1);
%Create the VideoWriter object to open an AVI file for writing.
vidWriter = VideoWriter('frames.avi');% video file to save dir
    documents/matlab
open(vidWriter);
%The following loop writes the acquired frames to the specified AVI file
    for future processing.
for index = 1:20 %number of frame
img = snapshot(cam); % Acquire frame for processing
    writeVideo(vidWriter, img); % Write frame to video
end
close(vidWriter); %Once the connection is no longer needed, clear the
    associated variable.
clear cam
```