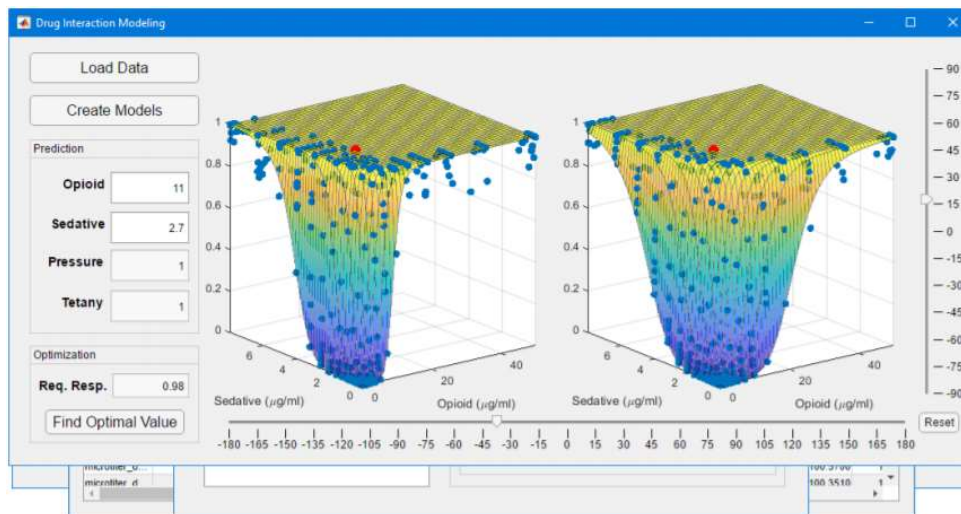


BÀI 2: - XÂY DỰNG ỨNG DỤNG XỬ LÝ ẢNH TRONG MATLAB VỚI APP DESIGNER - CÁC PHÉP XỬ LÝ ĐIỂM

I. App Designer

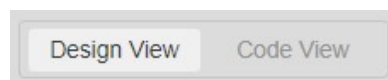
1. App Designer là gì?

- Là một môi trường phát triển tương tác mới cho việc xây dựng ứng dụng trong MATLAB.
- Cung cấp sự tích hợp hoàn chỉnh của hai tác vụ chính trong xây dựng ứng dụng đó là: thiết kế giao diện người dùng (bố cục các thành phần trực quan) và lập trình hành vi ứng dụng.
- Cung cấp trình quản lý bố cục lưới để tổ chức giao diện người dùng ứng dụng và các tùy chọn điều chỉnh tự động để làm cho ứng dụng phát hiện và phản hồi lại những thay đổi kích thước màn hình.
- Cho phép phân phối các ứng dụng bằng cách đóng gói chúng thành các trình cài đặt trực tiếp từ thành công cụ App Designer, hoặc bằng cách tạo ứng dụng web hoặc desktop hoạt động độc lập (yêu cầu có MATLAB Compiler).



2. Design view và Code view

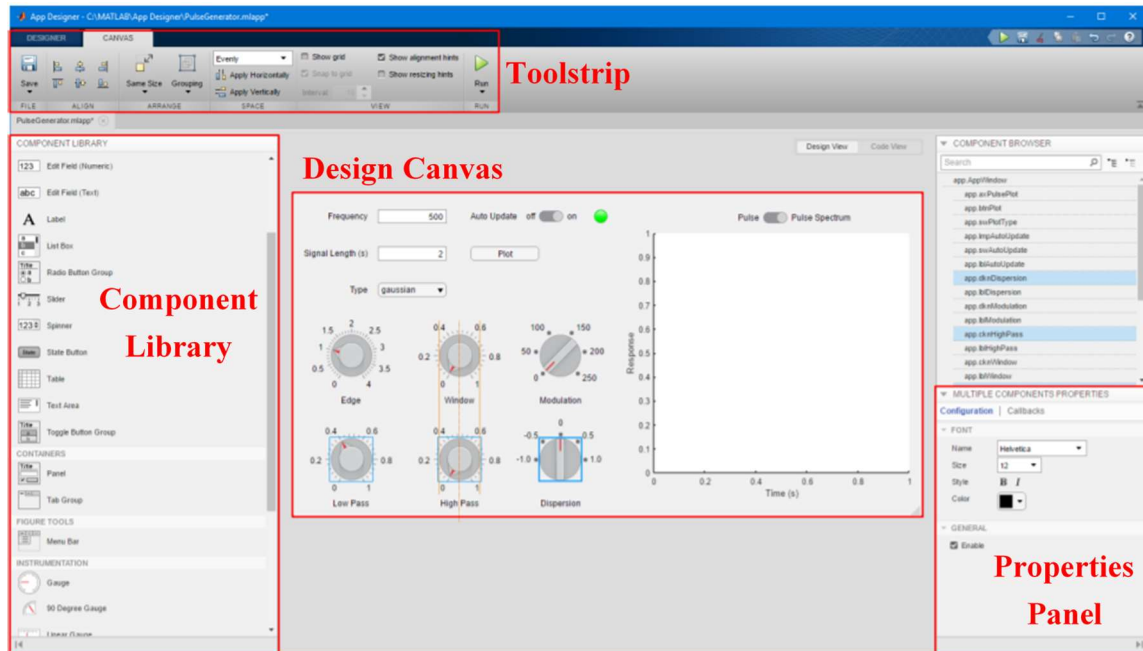
App Designer hỗ trợ hai khung nhìn làm việc: Design view và Code view. Chúng ta có thể thay đổi giữa hai khung nhìn bằng cách chọn chúng từ Design Canvas.



2.1 Design view

- Bố cục và thiết kế giao diện người dùng.
- Bao gồm:
 - + **Component Library:** cho phép chọn các thành phần và thêm nó vào Canvas.
 - + **Design Canvas:** cho phép bố cục các thành phần tạo giao diện người dùng.
 - + **Toolstrip:** căn chỉnh, tạo khoảng cách và gom nhóm các thành phần.

+ **Properties panel:** thiết lập các thuộc tính thành phần.



2.2 Code view

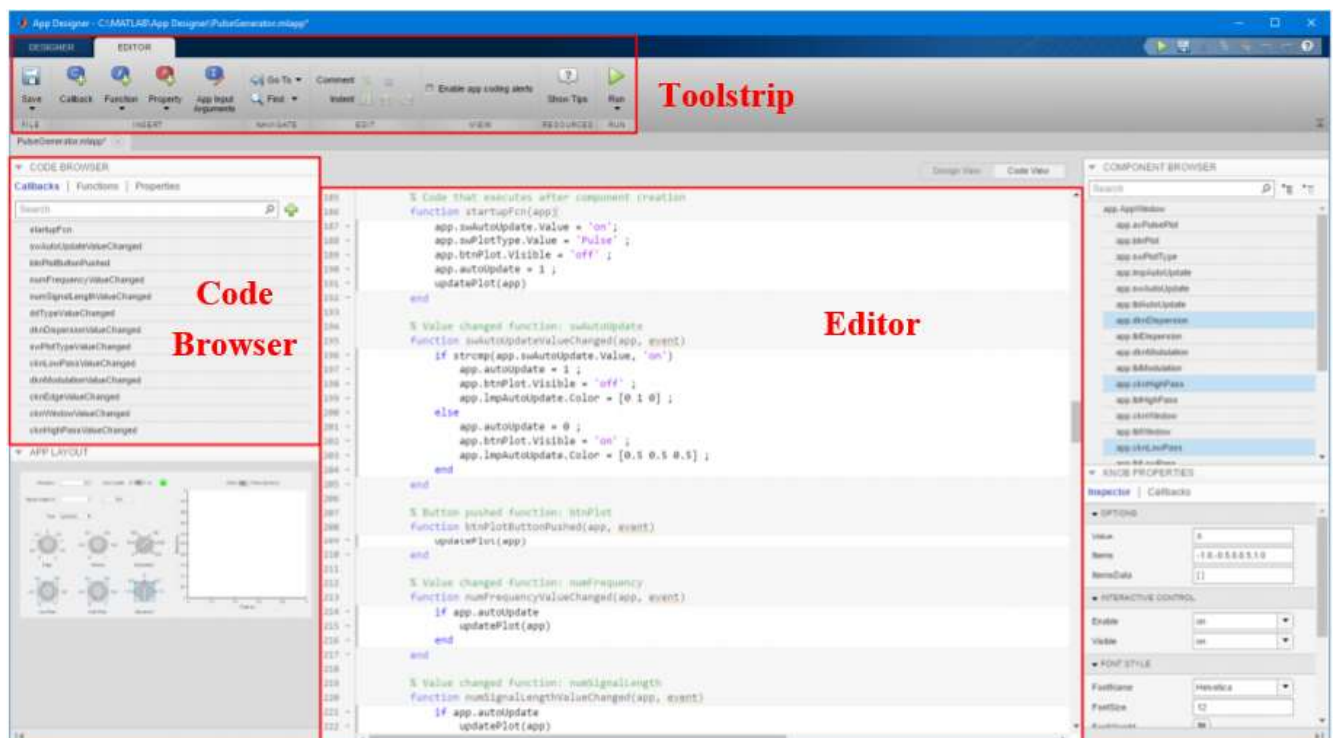
- Viết code để điều khiển hành vi của ứng dụng

- Bao gồm:

+ **Editor:** viết code cho các callback và các hàm khác

+ **Code Browser:** điều hướng đến các callback và các thuộc tính của ứng dụng

+ **Toolstrip:** thêm các thành phần code mới – các thuộc tính, callback và các hàm



3. Các thành phần của App Designer liên quan đến phát triển ứng dụng xử lý ảnh

3.1 Thành phần `uifigure`

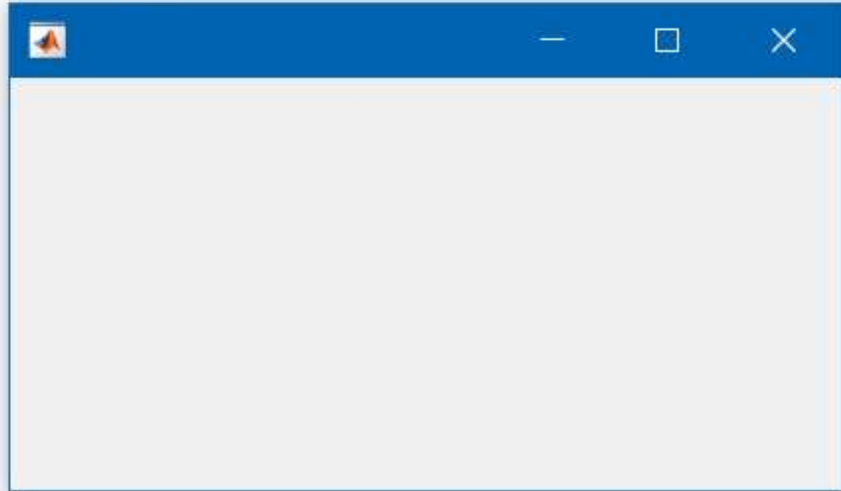
Cú pháp: `fig = uifigure`

`fig = uifigure(Name, Value)`

- `fig = uifigure`: tạo figure cho việc thiết kế giao diện người dùng và trả về một đối tượng Figure. Đây là kiểu figure App Designer sử dụng.

Ví dụ: tạo một figure mặc định:

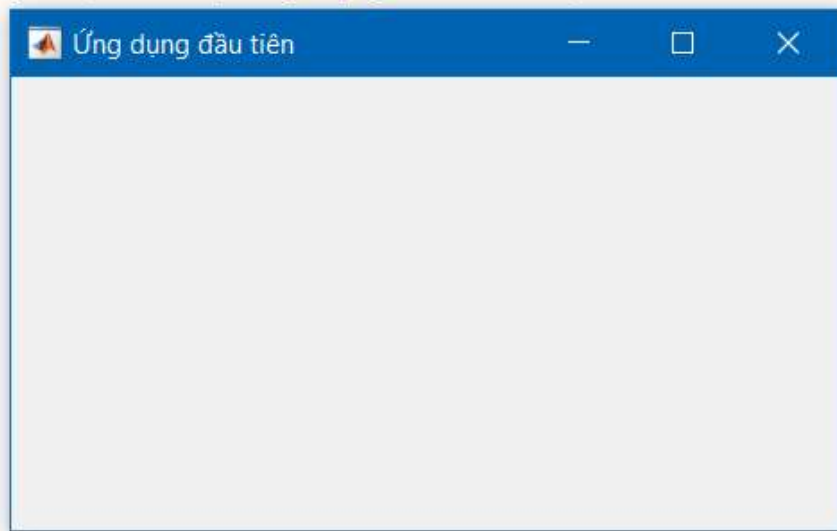
```
>> fig = uifigure;  
>>
```



- `fig = uifigure(Name, Value)`: chỉ định các thuộc tính của figure bằng cách sử dụng một hoặc nhiều cặp đối số `Name - Value`


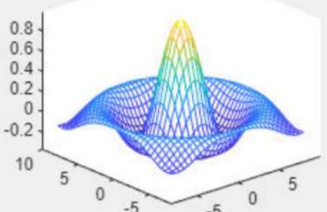


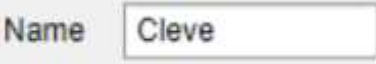


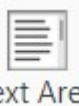
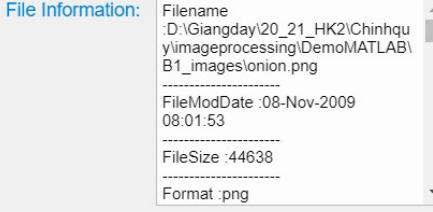

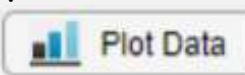

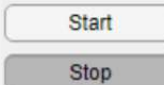
Ví dụ: tạo figure với tiêu đề được chỉ định là “Ứng dụng đầu tiên”


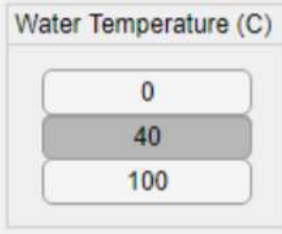



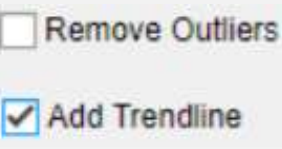

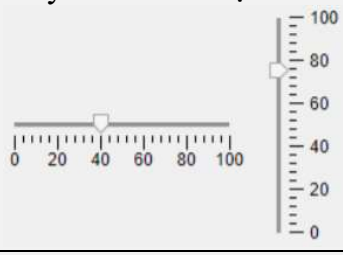


```
>> fig = uifigure('Name', 'Ứng dụng đầu tiên');  
>>
```

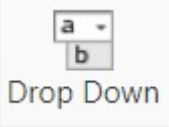



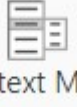


Link tham khảo: <https://www.mathworks.com/help/matlab/ref/uifigure.html>

3.2 Các thành phần chung

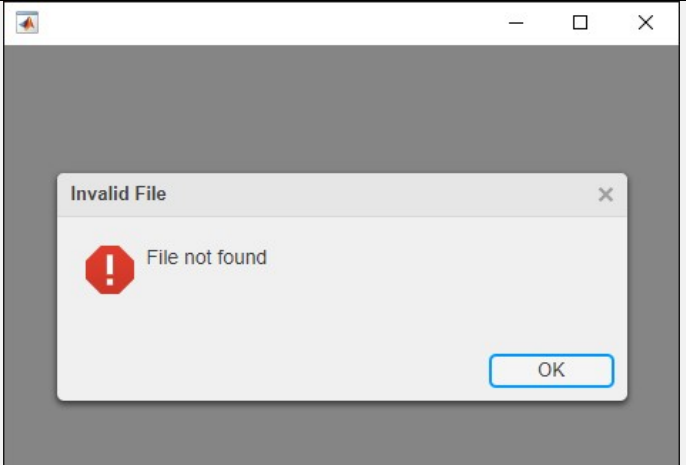
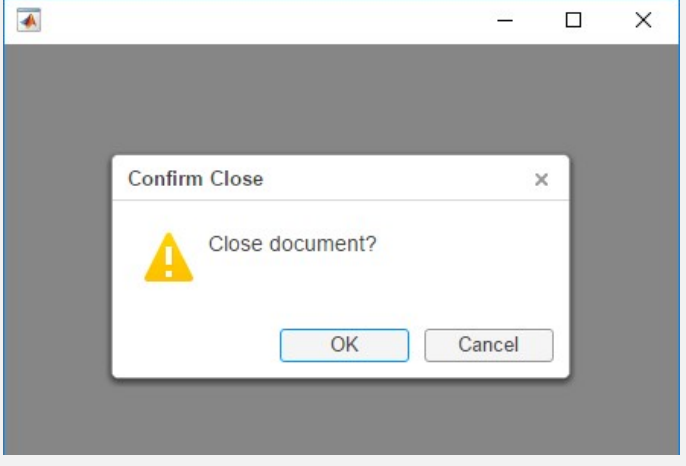
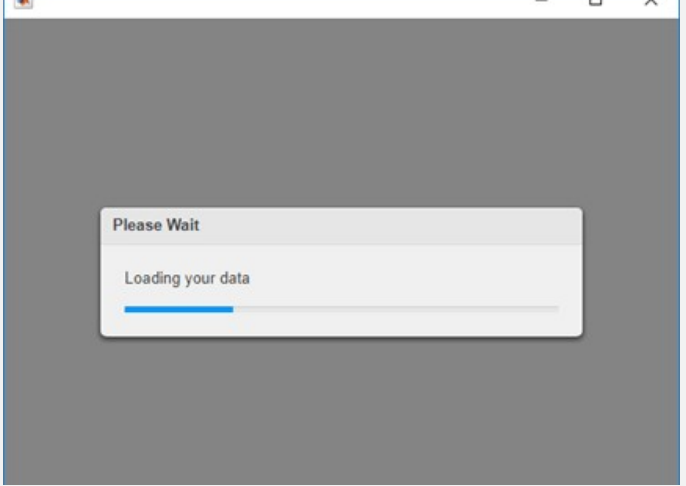
Thành phần (MATLAB class)	Icon	Mô tả và ví dụ
Axes <code>uiaxes</code>		<ul style="list-style-type: none"> - Cho phép hiển thị các đồ họa trên giao diện như đồ thị hay hình ảnh. - Giống tất cả các đối tượng đồ họa, axes có các thuộc tính cho phép chúng ta kiểm soát hành vi và vẻ ngoài của nó. 
Label <code>uicontrol</code>		<ul style="list-style-type: none"> - Hiển thị các dòng văn bản và người dùng không thể thay đổi nội dung của nó. - Được dùng cho việc gắn nhãn cho các thành phần khác. <p>Select an Option</p>
Edit Field <code>uicontrol</code>		Cho phép người dùng nhập hoặc điều chỉnh các chuỗi văn bản như dữ liệu đầu vào. 
Numeric Edit Field <code>uicontrol</code>		Cho phép người dùng nhập hoặc điều chỉnh dữ liệu đầu vào là các giá trị số. 
Text Area <code>uicontrol</code>		Cho phép người dùng nhập hoặc điều chỉnh dữ liệu đầu vào trên nhiều dòng. 
Button <code>uibutton</code>		Thực hiện một chức năng cụ thể khi được nhấn. 
State Button <code>uibutton</code>		<ul style="list-style-type: none"> - Tạo một hành động và cho biết chúng đang ở trạng thái bật hay tắt (2 trạng thái). Người dùng cần nhấn vào State Button để chuyển trạng thái của nó. 


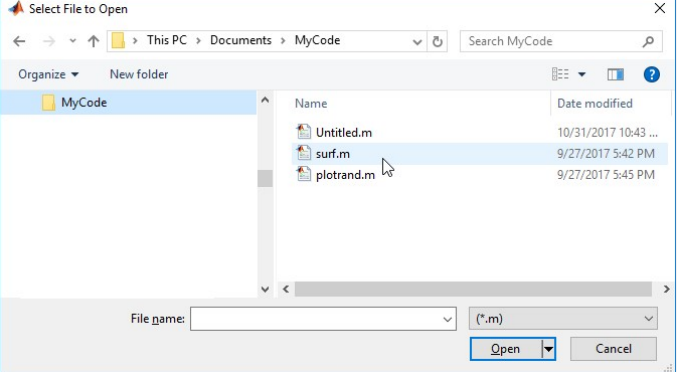
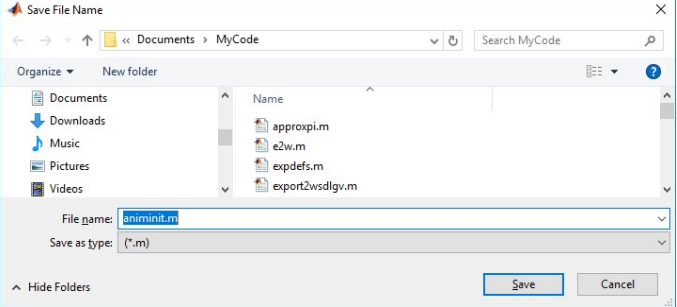
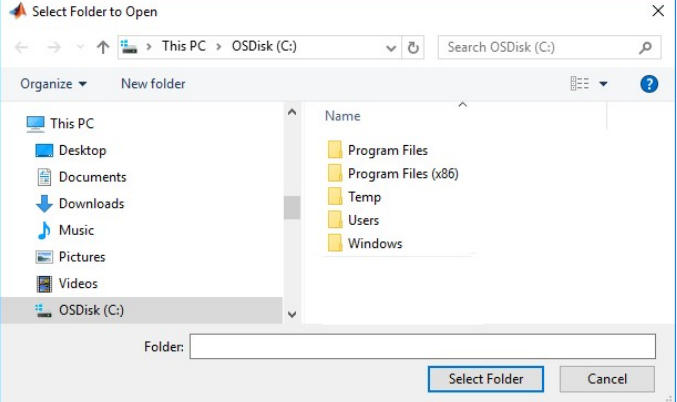
Thành phần (MATLAB class)	Icon	Mô tả và ví dụ
Toggle Button Group uibbuttongroup	 Toggle Button Group	- Chọn 1 trong các tùy chọn của các Toggle Button 
Radio Button Group uibbuttongroup	 Radio Button Group	- Cho phép chọn lựa các giá trị mang tính loại trừ nhau trong cùng một nhóm. Khi một giá trị được chọn, các giá trị trong cùng một nhóm sẽ tự động bị bỏ chọn. - Chọn: checked, bỏ chọn unchecked 
Check Box uicheckbox	 Check Box	- Cho phép chọn lựa nhiều giá trị cùng lúc (các giá trị độc lập với nhau) - Chọn: checked, bỏ chọn unchecked 
Slider uislider	 Slider	Cho phép người dùng thiết lập dữ liệu số trong một phạm vi được định trước bằng cách di chuyển thanh trượt. 
List Box uilistbox	 List Box	Hiển thị danh sách các item và cho phép người dùng chọn một hoặc nhiều item cùng lúc. 

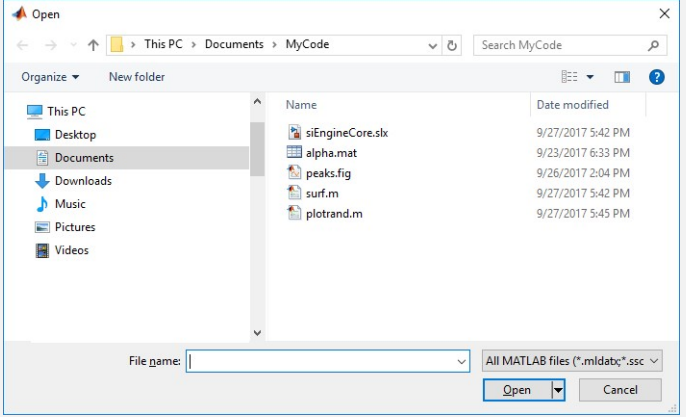
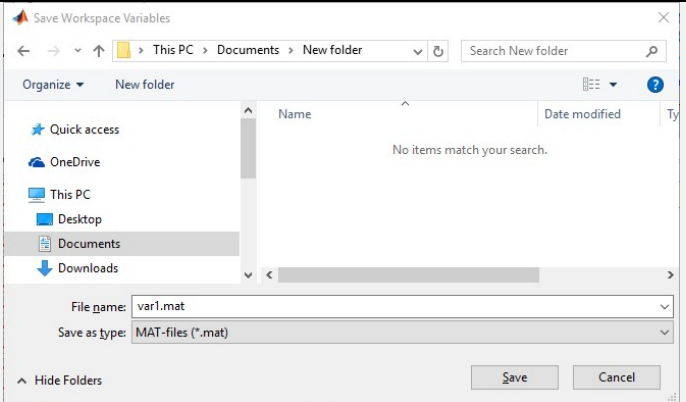
Thành phần (MATLAB class)	Icon	Mô tả và ví dụ
Drop Down uidropdown		Mở ra và hiển thị danh sách lựa chọn khi người dùng nhấp vào mũi tên. Chỉ một lựa chọn được thiết lập tại một thời điểm. <div> <div>Editable Drop Down</div> <div>Option 1</div> <div>Drop Down</div> <div>Red</div> <div>Red</div> <div>Green</div> <div>Blue</div> </div>
Panel uipanel		<ul style="list-style-type: none"> Sắp xếp các thành phần giao diện vào các nhóm. Giúp cho giao diện người dùng trở nên dễ hiểu hơn. Một Panel có thể có tiêu đề và đa dạng các kiểu đường viền. <div>Data</div>
Tab Group uitabgroup		<ul style="list-style-type: none"> Cho phép gom nhóm các thành phần và quản lý chúng trên các Tab riêng biệt <div>Data Plots</div>
Menu uimenu		<ul style="list-style-type: none"> Cho phép gom nhóm và hiển thị các lệnh cũng như các tùy chọn của ứng dụng bằng các chức năng. <div>File Edit Find Project</div> <div>Open</div> <div>Save</div> <div>Export</div>
Context Menu uicontextmenu		<ul style="list-style-type: none"> Hiển thị Context Menu khi Right-click trên thành phần liên kết <div>dd-mm-yyyy</div> <div>Change Format</div> <div>Restore Defaults</div>

3.3 Dialogs và Notification

Chúng ta không thể làm việc với các Dialog và Notification trong Design view. Thay vào đó, để thêm hoặc thao tác với chúng, chúng ta cần làm việc với Code view. Chi tiết từng Dialog được tham khảo từ các liên kết đến website của mathworks như trong bảng bên dưới.

Dialog + Link tham khảo	Ví dụ
uialert https://www.mathworks.com/help/matlab/ref/uialert.html	
uiconfirm https://www.mathworks.com/help/matlab/ref/uiconfirm.html	
uiprogressdlg https://www.mathworks.com/help/matlab/ref/uiprogressdlg.html	

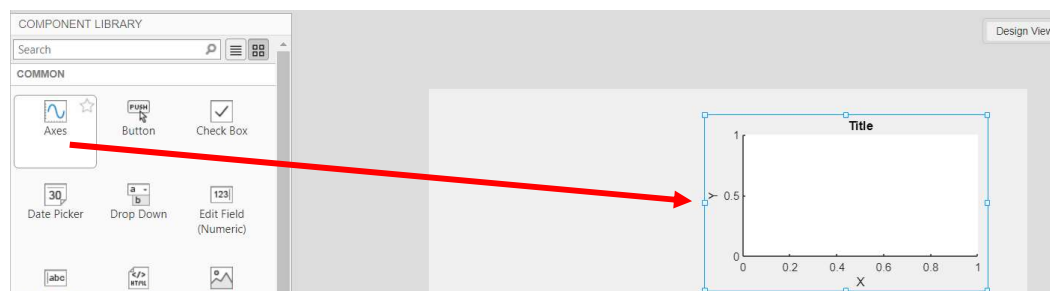
Dialog + Link tham khảo	Ví dụ
<p><code>uisetcolor</code></p> <p>https://www.mathworks.com/help/matlab/ref/uisetcolor.html</p>	
<p><code>uigetfile</code></p> <p>https://www.mathworks.com/help/matlab/ref/uigetfile.html</p>	
<p><code>uiputfile</code></p> <p>https://www.mathworks.com/help/matlab/ref/uiputfile.html</p>	
<p><code>uigetdir</code></p> <p>https://www.mathworks.com/help/matlab/ref/uigetdir.html</p>	

Dialog + Link tham khảo	Ví dụ
uiopen https://www.mathworks.com/help/matlab/ref/uiopen.html	
uisave https://www.mathworks.com/help/matlab/ref/uisave.html	

4. Các bước thực hiện cơ bản

4.1 Thiết kế giao diện người dùng với Design view

Bước 1: Chọn thành phần cần thêm từ Component Library và kéo thả nó vào Design Canvas.



Chúng ta có thể điều chỉnh kích thước thành phần bằng cách kéo chuột trên Design Canvas, hoặc click chuột trên Design Canvas để thêm thành phần ở kích thước mặc định của nó.

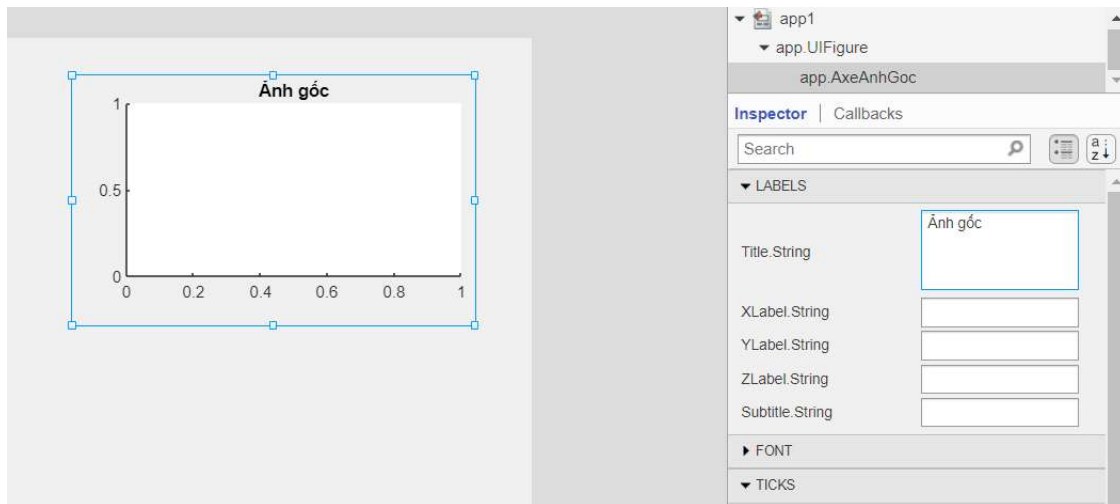
Bước 2: Đặt tên cho thành phần.

- Double Click vào tên thành phần trong **Component Browser** và đặt lại tên cho nó



Lưu ý: chúng ta có thể chọn thành phần bằng cách nhấp vào thành phần trong Design Canvas hoặc nhấp chọn thành phần từ Component Browser.

Bước 3: Thiết lập các thuộc tính (nếu cần) của thành phần trong Properties Panel

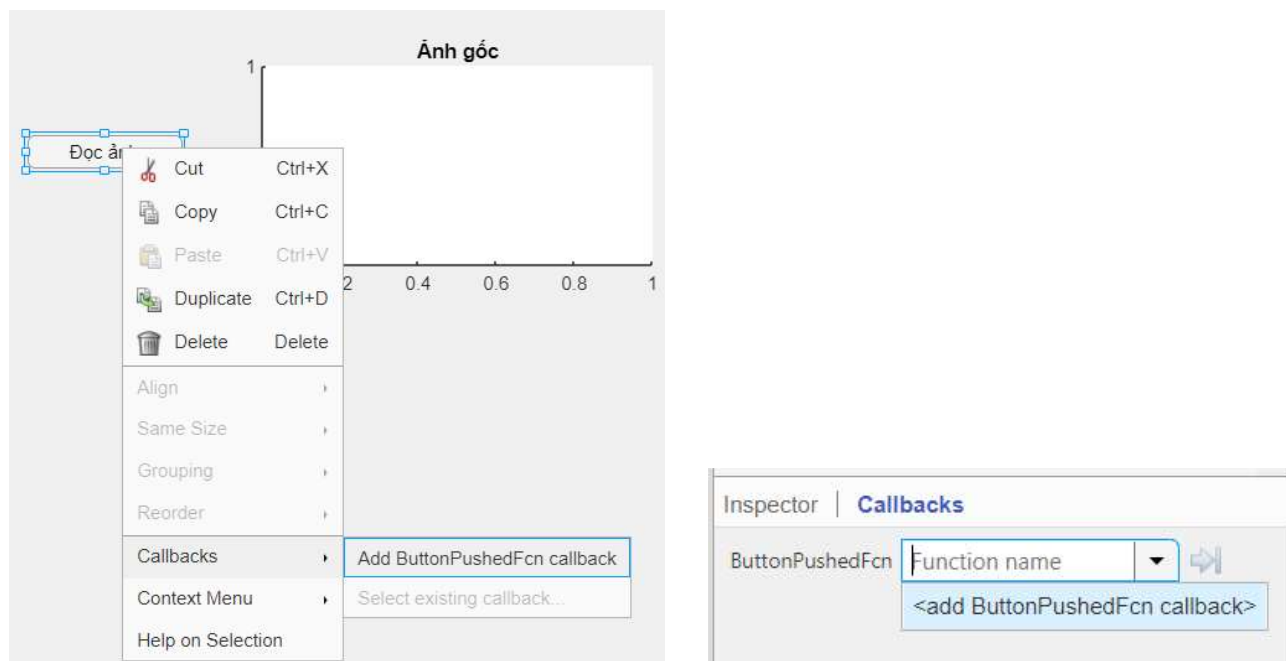


Bước 4: Điều chỉnh vị trí bằng tay hoặc căn chỉnh nó với những thành phần khác.

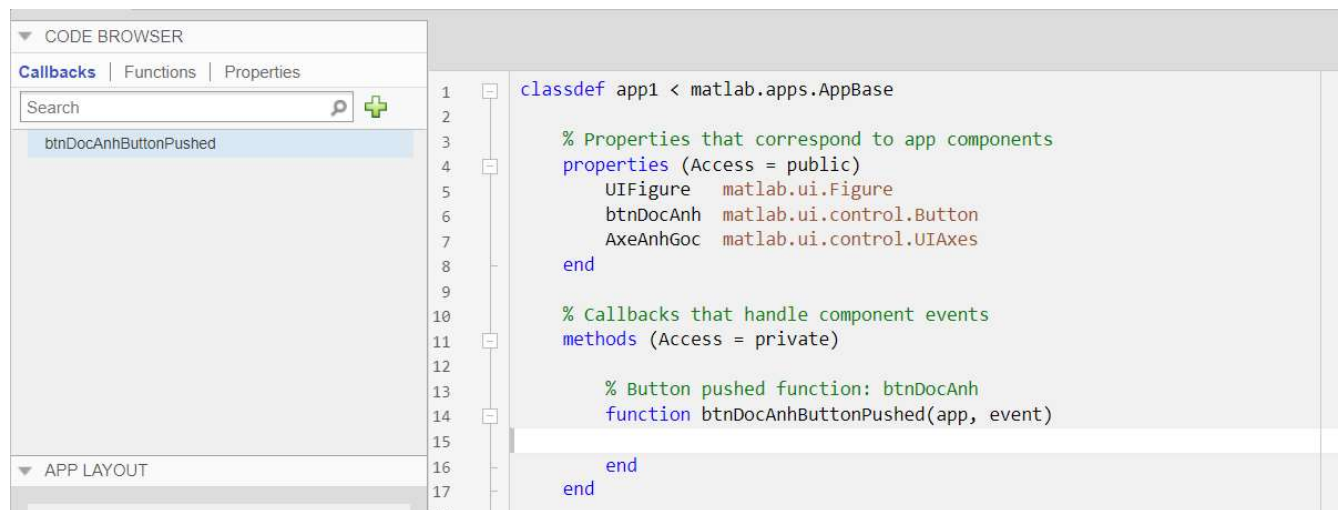
4.2 Viết mã lệnh cho hành vi của ứng dụng với Code view

Bước 1: Chọn một thành phần

Bước 2: Tạo một Callback từ Design view theo 2 cách như sau



- Sau khi tạo, Callback sẽ được thêm vào **Code Browser** và code tương ứng với callback được tự động sinh ra trong **Editor**



- Lưu ý: chúng ta chỉ có thể xóa Callback, Functions và Properties từ Code Browser, không thể xóa trong Editor.

Bước 3: Thêm mã lệnh Callback

```

% Button pushed function: btnDocAnh
function btnDocAnhButtonPushed(app, event)
    [filename, path]=uigetfile({'*.png'; '*.jpg'; '*.tif'}, "Chọn ảnh");
    if isequal(filename,0)||isequal(path,0)
        uialert(app.UIFigure, "Chưa đọc ảnh", "Lỗi đọc ảnh");
        cla(app.AxeAnhGoc);
    else
        file=strcat(path,filename);
        img=imread(file);
        imshow(img, 'Parent', app.AxeAnhGoc);
    end
end
  
```

Lưu ý: nên sử dụng các gợi ý để tránh các lỗi lập trình phổ biến

4.3 Cấu trúc mã lệnh ứng dụng

- Mã lệnh được tạo cho ứng dụng là một lớp MATLAB.
- Các điều khiển và dữ liệu chia sẻ được lưu trữ như những thuộc tính (properties) của lớp.
- Các Callback và các hàm trợ giúp được lưu trữ như các phương thức (methods) của lớp.
- App Designer tạo mã lệnh cho tất cả các thành phần của ứng dụng.

```

classdef FirstIPApp < matlab.apps.AppBase

    % Properties that correspond to app components
    properties (Access = public)
        UIFigure          matlab.ui.Figure
        btnDocAnh          matlab.ui.control.Button
        CVHINTHNHLabel     matlab.ui.control.Label
        AxesAnhGoc         matlab.ui.control.UIAxes
    end

    % Callbacks that handle component events
    methods (Access = private)

        % Button pushed function: btnDocAnh
        function btnDocAnhButtonPushed(app, event)

    end

end

% Component initialization
methods (Access = private)

    % Create UIFigure and components
    function createComponents(app)

        % Create UIFigure and hide until all components are created
        app.UIFigure = uifigure('Visible', 'off');
        app.UIFigure.Position = [100 100 640 480];

```

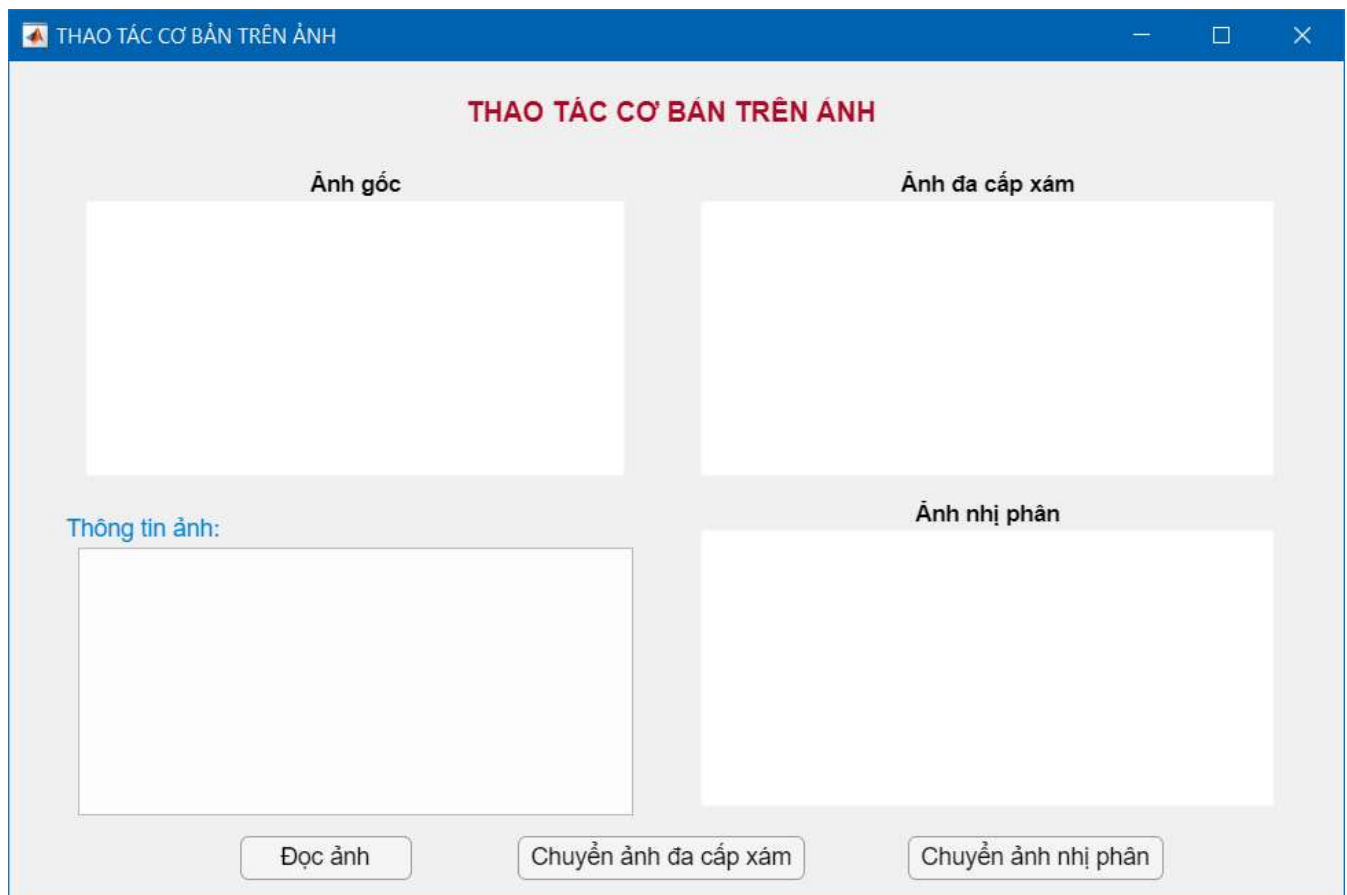
5. Xây dựng ứng dụng xử lý ảnh đầu tiên

Xây dựng ứng dụng “THAO TÁC CƠ BẢN TRÊN ẢNH” với các chức năng như sau:

- Đọc và hiển thị tập tin ảnh đơn
- Hiển thị thông tin ảnh
- Chuyển ảnh sang ảnh đa cấp xám và ảnh nhị phân.

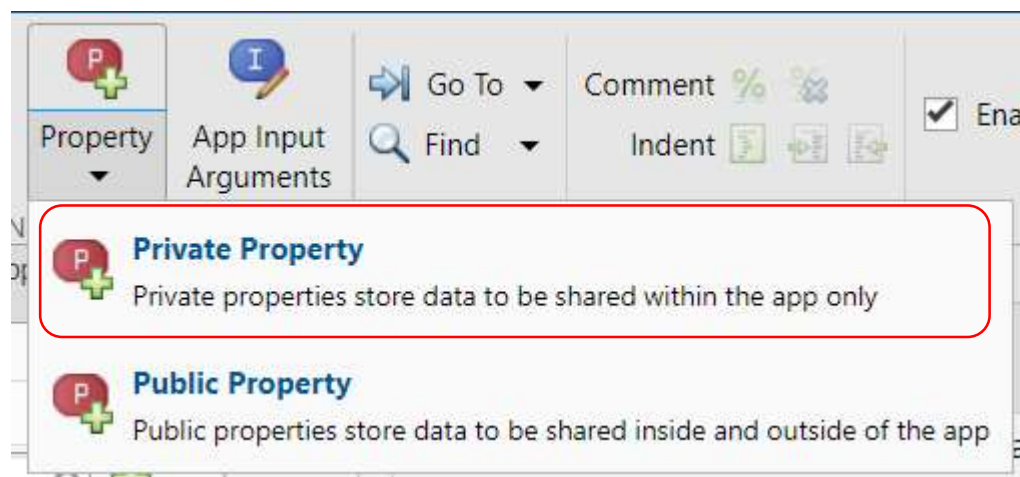
5.1 Thiết kế giao diện người dùng với các thành phần như sau:

- 3 uiaxes cho phép hiển thị ảnh gốc, ảnh đa cấp xám và ảnh nhị phân.
- 3 uibutton: đọc ảnh, chuyển ảnh đa cấp xám và chuyển ảnh nhị phân.
- 1 uitextarea: cho phép hiển thị thông tin ảnh.
- 1 uilabel: hiển thị tiêu đề chức năng.



5.2 Viết mã lệnh thực thi hành vi của ứng dụng:

- Thêm một property tạo một biến cho phép lưu trữ dữ liệu và chia sẻ giữa các Callback và functions. Ở đây chúng ta sẽ thêm ba property **img**, **map** và **colortype** có kiểu truy xuất private cho phép lưu trữ ảnh đọc vào, colormap của ảnh đã đọc và thuộc tính ColorType của tập tin ảnh, phục vụ cho các chức năng chuyển ảnh đa cấp xám và chuyển ảnh nhị phân.




```

properties (Access = private)
    img% Lưu trữ ảnh đọc vào
    map% Lưu trữ colormap của ảnh đọc vào
    colortype % Lưu trữ thông tin loại ảnh (RGB, grayscale, indexed...)
end

```

Đọc ảnh:

- Khi người dùng click vào nút đọc ảnh, một hộp thoại uigetfile được mở ra, cho phép người dùng chọn ảnh có định dạng **jpg**, **png** hoặc **tif**.

- Sau đó người dùng click OK, ảnh sẽ được hiển thị trong AxesAnhGoc, và thông tin của tập tin ảnh sẽ được hiển thị trong txtaThongTin.

- Trong trường hợp người dùng chọn Cancel, tương ứng với việc hủy bỏ thao tác đọc ảnh, ứng dụng cần hiển thị một dialogbox thông báo việc chưa chọn ảnh của người dùng.

```

% Button pushed function: btnDocAnh
function btnDocAnhButtonPushed(app, event)
    f=app.UIFigure;
    [filename, path]=uigetfile({'*.png'; '*.jpg'; '*.tif'}, 'Chọn ảnh');
    if isequal(filename,0)||isequal(path,0)
        uialert(f, 'chưa chọn ảnh', 'Lỗi đọc ảnh');
        cla(app.AxesAnhGoc);
    else
        file=strcat(path,filename);
        [app.img,app.map]=imread(file);
        imshow(app.img,app.map,'Parent',app.AxesAnhGoc);
        info=imfinfo(file);
        arr=[];
        if length(info)>1
            app.lblThongtinAnh.Text=strcat(strcat("Tập tin chứa: ",num2str(length(info))), " ảnh");
            arr=[arr;"THÔNG TIN ẢNH ĐẦU TIÊN"];
            info=info(1);
        else
            app.lblThongtinAnh.Text="";
        end
        app.colortype=info.ColorType;
        fields=fieldnames(info);
        for i=1:length(fields)
            if isnumeric(info.(char(fields(i))))
                value=num2str(info.(char(fields(i))));
            else
                value=info.(char(fields(i)));
            end
            str= strcat(fields(i),": ", value);
            arr=[arr;str];
            arr=[arr;'-----'];
        end
        app.txtaThongTin.Value=arr;
    end
end

```

Chuyển ảnh đa cấp xám:

- Khi người dùng click vào btnChuyenAnhXam, ứng dụng sẽ thực hiện việc chuyển đổi ảnh đầu vào thành ảnh đa cấp xám và hiển thị nó vào AxesAnhXam.

```
% Button pushed function: btnChuyenAnhXam
function btnChuyenAnhXamButtonPushed(app, event)
    if app.colortype=="truecolor"
        rgb=app.img;
        gray=rgb2gray(rgb);
        imshow(gray, 'Parent', app.AxesAnhXam);
    elseif app.colortype=="indexed"
        ind=app.img;
        gray=ind2gray(ind, app.map);
        imshow(gray, 'Parent', app.AxesAnhXam);
    else
        imshow(app.img, 'Parent', app.AxesAnhXam);
    end
end
```

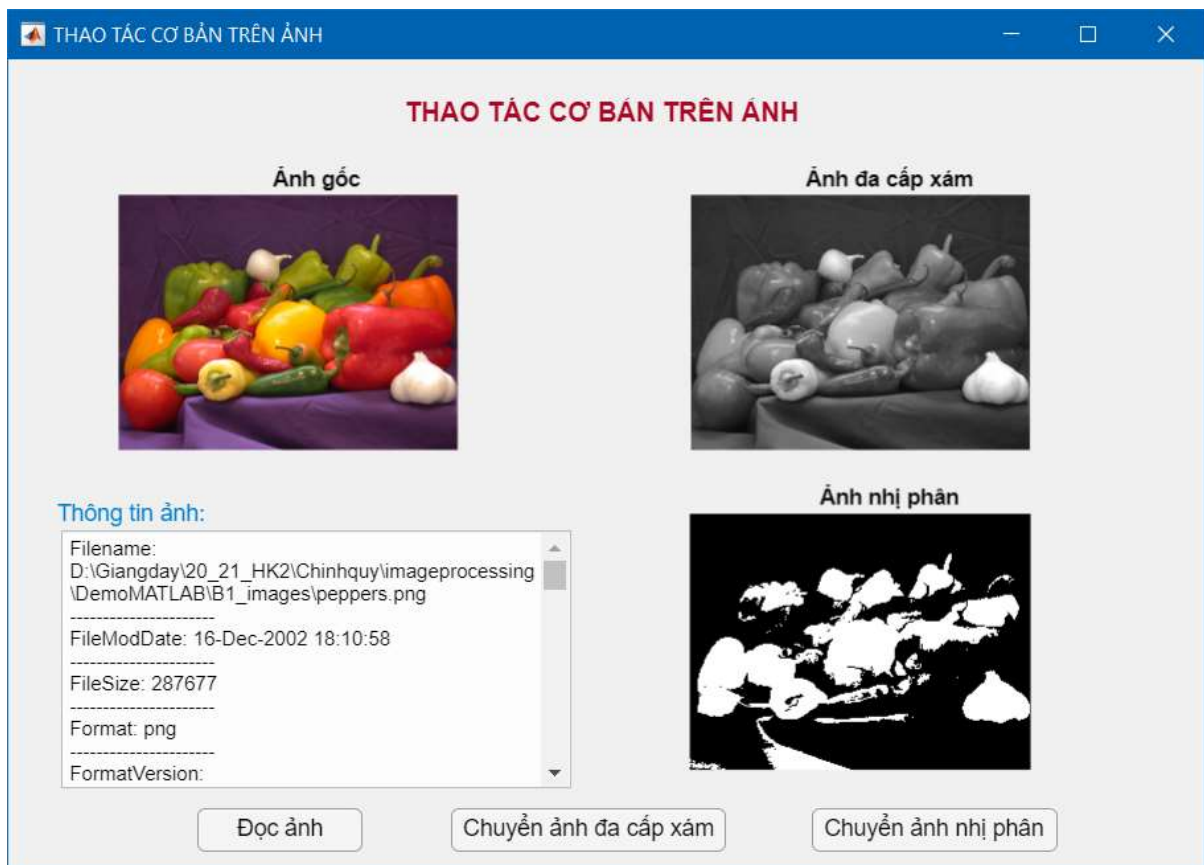
Chuyển ảnh nhị phân:

- Khi người dùng click vào btnChuyenAnhNhiPhan, ứng dụng sẽ thực hiện việc chuyển đổi ảnh đầu vào thành ảnh nhị phân và hiển thị nó vào AxesAnhNhiPhan

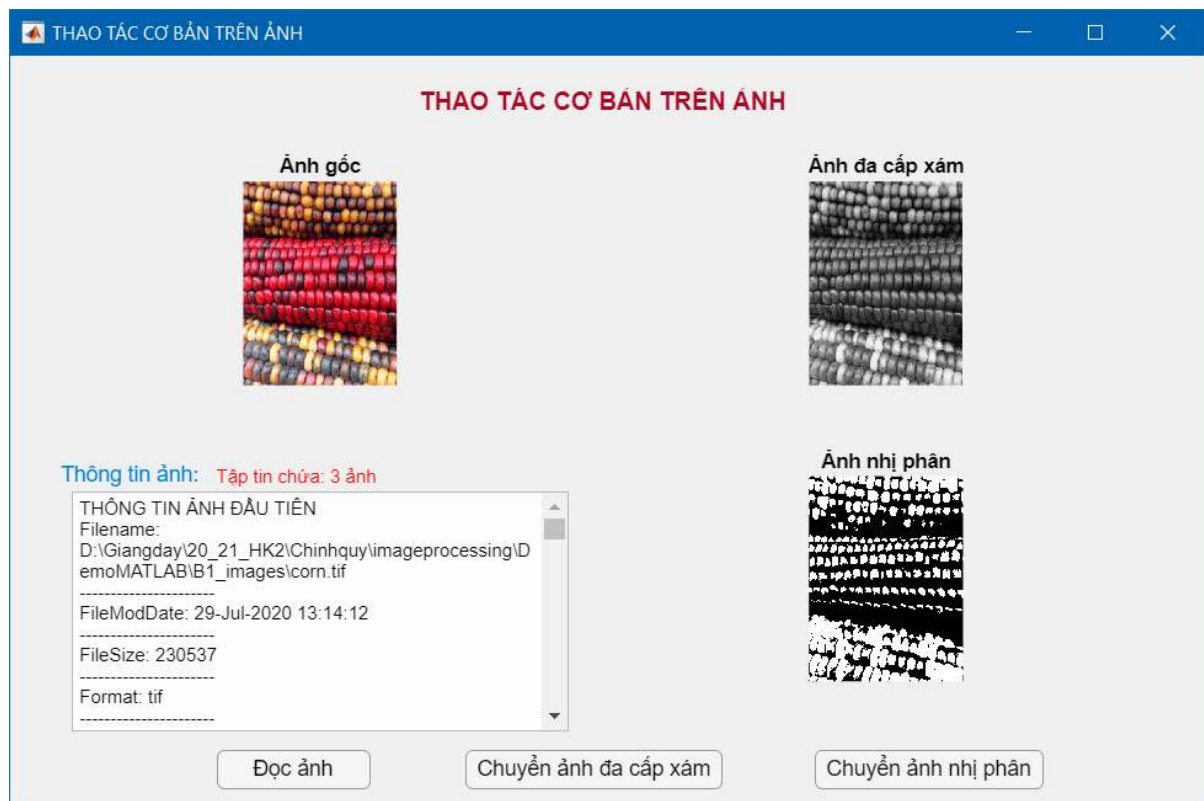
```
% Button pushed function: btnChuyenAnhNhiPhan
function btnChuyenAnhNhiPhanButtonPushed(app, event)
    if app.colortype=="truecolor"
        rgb=app.img;
        gray=rgb2gray(rgb);
        bw=imbinarize(gray);
        imshow(bw, 'Parent', app.AxesAnhNhiPhan);
    elseif app.colortype=="indexed"
        ind=app.img;
        gray=ind2gray(ind, app.map);
        bw=imbinarize(gray);
        imshow(bw, 'Parent', app.AxesAnhNhiPhan);
    else
        bw=imbinarize(app.img);
        imshow(bw, 'Parent', app.AxesAnhNhiPhan);
    end
end
```

Kết quả thu được như sau:

- Đọc tập tin có một ảnh đơn:



- Đọc tập tin có chứa nhiều ảnh:



II. Các phép xử lý điểm

1. Hiển thị Histogram của ảnh

- Sử dụng Hàm imhist của MATLAB IPT cho việc đọc histogram của ảnh.

- Cú pháp:

```
[counts,binLocations] = imhist(I)
```

```
[counts,binLocations] = imhist(I,n)
```

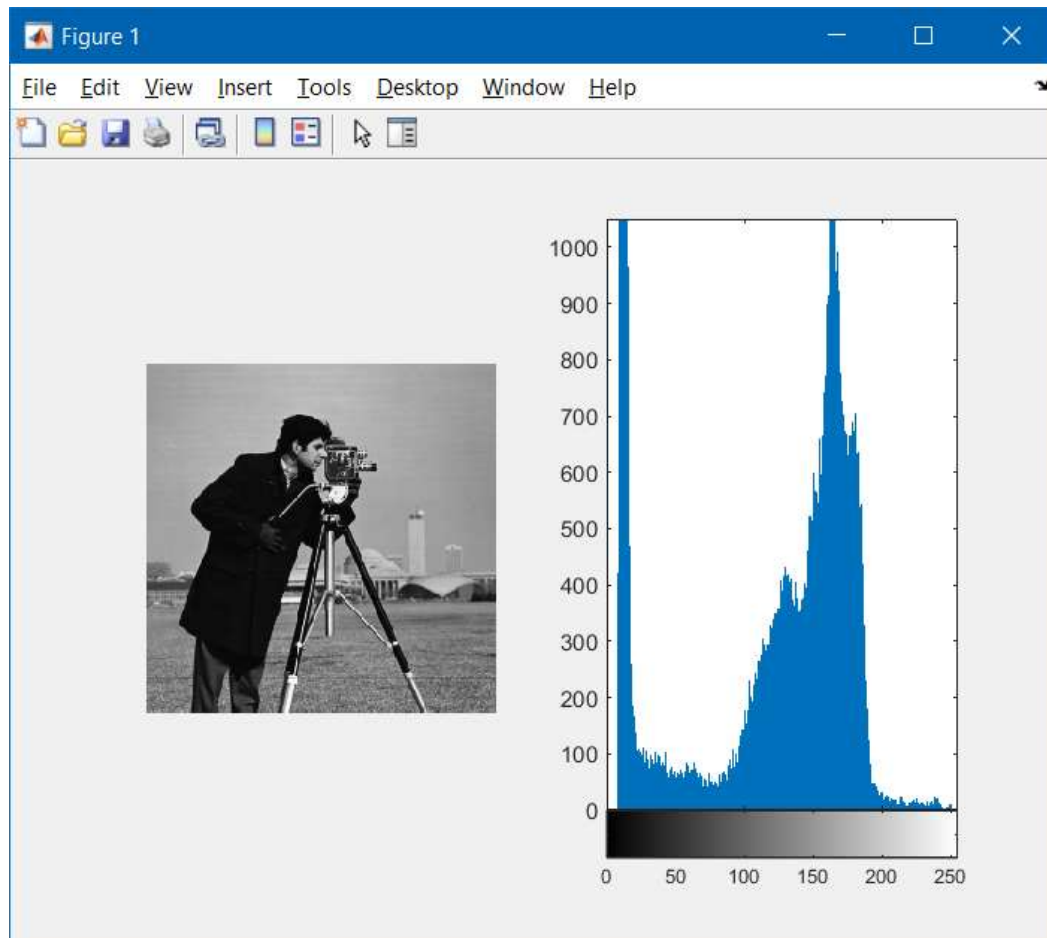
```
[counts,binLocations] = imhist(X,map)
```

- Mô tả: tham khảo liên kết <https://www.mathworks.com/help/images/ref/imhist.html>

- Ví dụ: Đọc ảnh bằng hàm imread, sau đó dùng hàm imhist cho việc hiển thị histogram của ảnh.

```
>> img=imread('B2_images\cameraman.tif');  
>> subplot(121);  
>> imshow(img);  
>> subplot(122);  
>> imhist(img);
```

- Kết quả:



Yêu cầu:

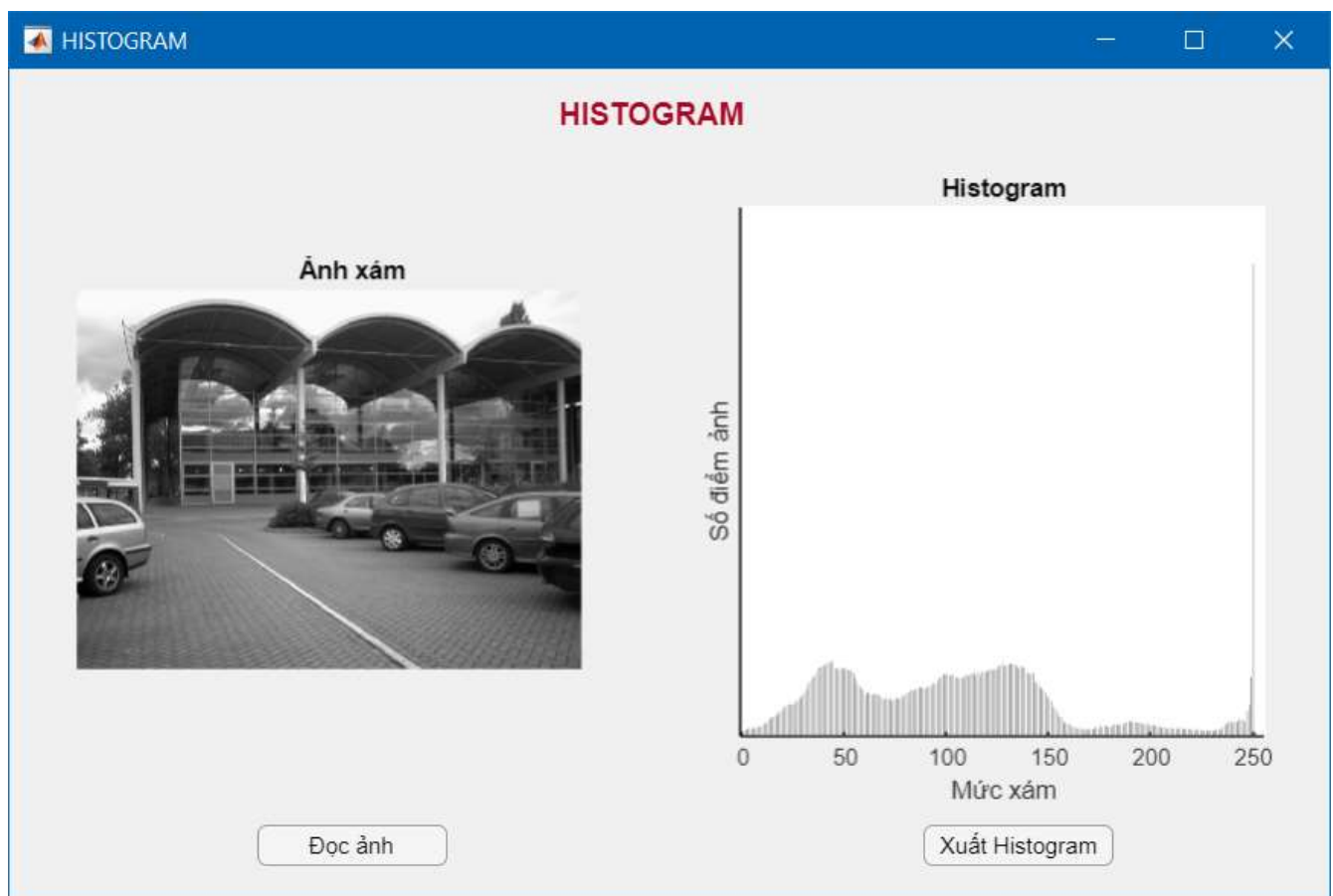
- Trong Command windows, sử dụng cấu trúc lệnh như ví dụ, đọc các ảnh trong B2_images và hiển thị histogram của nó.
- Xây dựng ứng dụng HISTOGRAM cho phép hiển thị histogram của ảnh được đọc vào với các yêu cầu cụ thể sau:

✓ Ảnh được đọc có thể là ảnh RGB, indexed, grayscale nhưng sẽ được chuyển sang ảnh grayscale trước khi hiển thị.

✓ Xử lý Callback của nút Xuất Histogram cho phép xuất histogram của ảnh xám đọc được. Gợi ý sử dụng hàm **bar** (<https://www.mathworks.com/help/matlab/ref/bar.html>) như sau cho việc vẽ biểu đồ cột trong uiaxes.

```
% Button pushed function: btnXuathis
function btnXuathisButtonPushed(app, event)
    [pixels, graylevels]=imhist(app.img);
    bar(app.AxeHistogram,graylevels,pixels,0.2,"stacked","black");
end
```

✓ Thay đổi các tham số của **bar** và ghi nhận những thay đổi trên histogram của ảnh.



c. Trong Command Window, gõ các lệnh sau, hãy nhận xét kết quả thu được. Biết rằng: red tương ứng với $\text{img}(:, :, 1)$, green tương ứng với $\text{img}(:, :, 2)$ và blue tương ứng với $\text{img}(:, :, 3)$

```
>> img=imread('B2_images\twins.tif');
>> red=img;
>> red(:, :, 2)=0;
>> red(:, :, 3)=0;
>> green=img;
>> green(:, :, 1)=0;
>> green(:, :, 3)=0;
>> blue=img;
>> blue(:, :, 1)=0;
>> blue(:, :, 2)=0;
>> subplot(221);
>> imshow(img);
>> subplot(222);
>> imshow(red);
>> subplot(223);
>> imshow(green);
>> subplot(224);
>> imshow(blue);
```

d. Xây dựng ứng dụng cho phép đọc vào một ảnh màu RGB xuất ra các ảnh trên từng băng màu tương ứng.

e. Xây dựng ứng dụng cho phép đọc ảnh màu RGB và xuất histogram tương ứng với 3 kênh màu Red, Green, Blue của ảnh.

2. Ảnh âm bản

- Đối với ảnh nhị phân: 1 thành 0, 0 thành 1
- Đối với ảnh đa cấp xám: $g(x,y) = L_{\max} - f(x,y)$ với:
 - + $f(x,y)$ là giá trị mức xám điểm ảnh trên ảnh input
 - + $g(x,y)$ là giá trị mức xám trên ảnh output
 - + L_{\max} giá trị mức xám cao nhất (thang mức xám phổ biến 256 mức, L_{\max} bằng 255)
- Đối với ảnh màu, red sẽ thành cyan, green sẽ thành magenta và blue sẽ thành yellow (tức màu bù của chúng) và ngược lại.
- Sử dụng hàm `imcomplement` của MATLAB IPT. Với cú pháp:

$$J = \text{imcomplement}(I)$$
- Mô tả: tính phân bù của I và trả về cho J. Tham khảo mô tả chi tiết và ví dụ về hàm `imcomplement` qua liên kết <https://www.mathworks.com/help/images/ref/imcomplement.html>
- Ví dụ 1: Trong Command Window gõ các lệnh sau:

```
>> img=imread('B2_images\carpark.png');
>> gray=rgb2gray(img);
>> com=imcomplement(gray);
>> subplot(131);
>> imshow(img);
>> subplot(132);
>> imshow(gray);
>> subplot(133);
>> imshow(com);
```

- Kết quả thu được:



- Ví dụ 2: Tạo một M-file mới chứa hàm **anhamban** với nội dung như sau:

```
function anhamban(file)
% Hàm trả về ảnh âm bản của một ảnh
% Đọc ảnh từ file
% Tính ảnh âm bản của ảnh đầu vào bằng hàm imcomplement
% Hiển thị ảnh đầu vào và ảnh âm bản
img=imread(file);
com=imcomplement(img);
subplot(121);
imshow(img);
subplot(122);
imshow(com);
```

Trong Command Window gõ lệnh sau:

```
>> anhamban('B2_images\carpark.png');
```

Kết quả thu được:



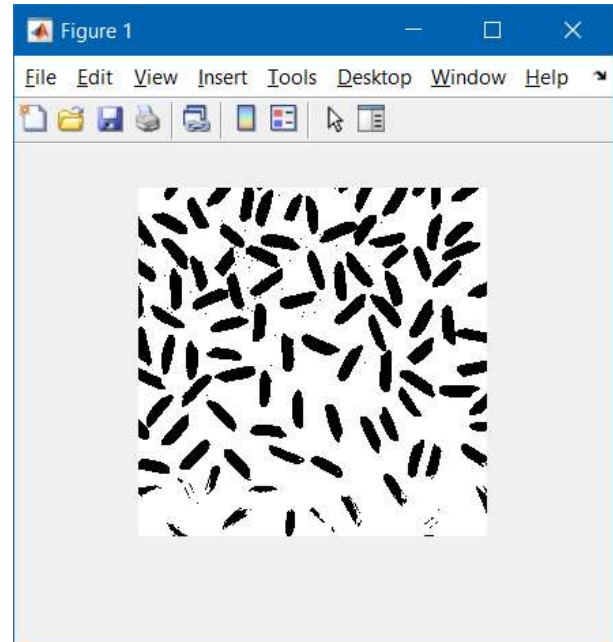
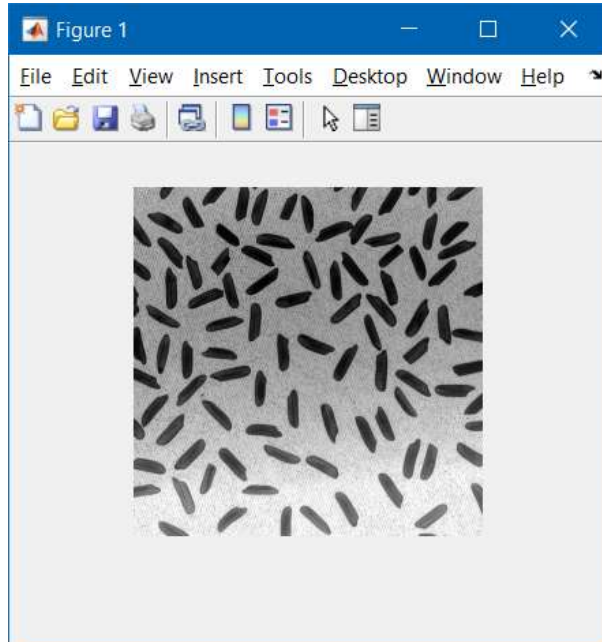
- Ví dụ 3: Xây dựng hàm **layanhamban** cho phép đọc một ảnh (nhị phân hoặc đa cấp xám) và trả về ảnh âm bản của nó bằng giải thuật như đã trình bày.

```
function [com,omap]=layanhamban(file)
% Hàm trả về ảnh âm bản của một ảnh
% Đọc ảnh từ file
% Tính ảnh âm bản của ảnh đầu vào
% Trả kết quả ảnh đầu ra cho com,
% và colormap của ảnh đầu vào cho omap cho việc hiển thị ảnh
[img,map]=imread(file);
info=imfinfo(file);
if length(info)>1
    info=info(1);
end
Lmax=2^info.BitDepth-1;
[y1,x1]=size(img);
com=zeros(y1,x1);
omap=map;
for i=1:y1
    for j=1:x1
        com(i,j)=Lmax-img(i,j);
    end
end
```

Kiểm tra tính đúng đắn của hàm **layanhamban** bằng cách gõ lần lượt các lệnh sau trong Command Window.

```
>> [com,map]=layanhamban('B2_images\rice.png');
>> imshow(com,map);
>> [com,map]=layanhamban('B2_images\ricebinary.png');
>> imshow(com,map);
```

Kết quả thu được:



Yêu cầu:

- Sử dụng hàm **anhamban**, đọc và hiển thị các ảnh trong gói dữ liệu ảnh B2_images. Kiểm tra kết quả thu được.
- Xây dựng hàm **layanhambanRGB** cho phép đọc một ảnh màu RGB và trả về ảnh âm bản của nó.
- Xây dựng ứng dụng cho phép đọc vào một ảnh và hiển thị ảnh âm bản của nó.



3. Các phép xử lý Tổ chức đồ (Histogram)

3.1 Trượt Tổ chức đồ, điều chỉnh độ sáng tối của ảnh

$$O(x,y) = I(x,y) + c$$

Nếu $c > 0$, trượt tổ chức đồ về bên phải, tăng độ sáng của ảnh (ảnh sáng hơn)

Nếu $c < 0$, trượt tổ chức đồ về bên trái, giảm độ sáng của ảnh (ảnh tối hơn)

Yêu cầu:

Xây dựng hàm) thực thi giải thuật trượt tổ chức đồ như mô tả

```
function [oimg,omap] = truotHistogram(file, c)
```

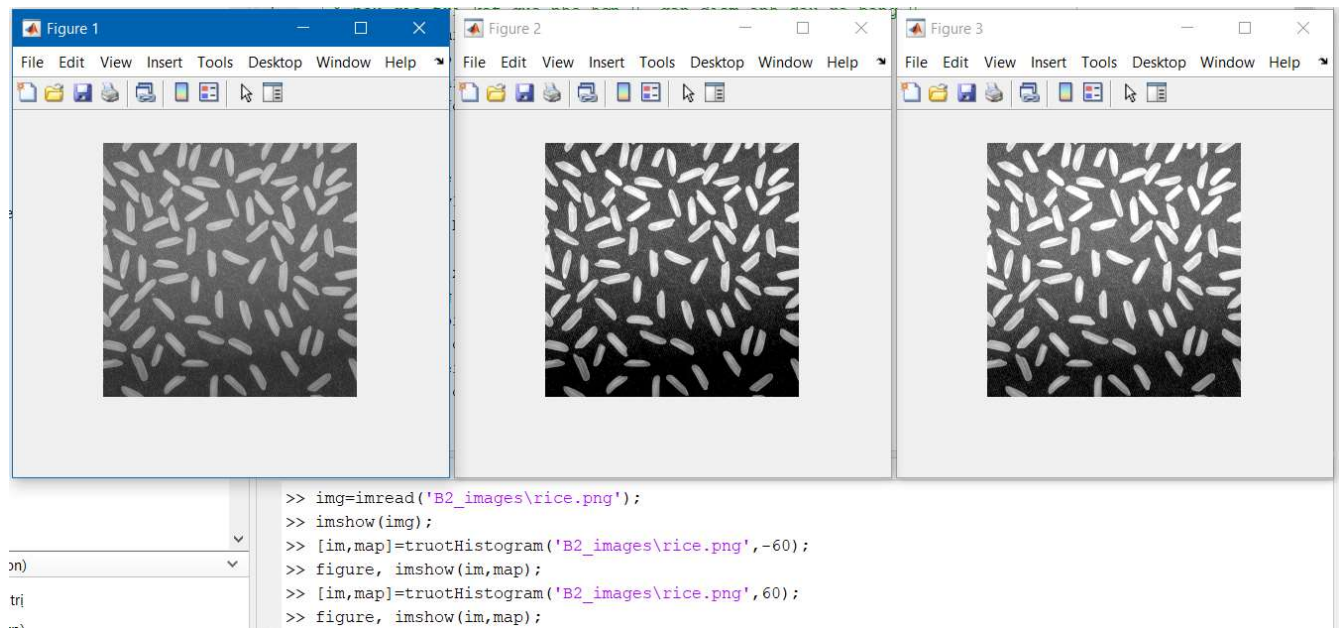
Điều kiện:

- + nếu giá trị mức xám thu được lớn hơn L_{\max} thì gán bằng L_{\max} ;
- + nếu giá trị mức xám thu được nhỏ hơn 0 thì gán bằng 0.

Kiểm tra tính đúng đắn của hàm truotHistogram từ Command Window.

Gợi ý: tham khảo hàm layanhamban, ở mỗi điểm ảnh đầu ra (sau khi cộng giá trị điểm ảnh đầu vào với hằng số c), kiểm tra giá trị điểm ảnh đầu ra thỏa điều kiện đã nêu.

Kết quả kiểm tra tính đúng đắn của hàm truotHistogram.



3.2 Căng tổ chức đồ, điều chỉnh độ tương phản của ảnh

$$O(x,y) = I(x,y) * c \text{ (với } c > 0 \text{)}$$

Nếu $c < 1$, thu hẹp chân tổ chức đồ, giảm độ tương phản

Nếu $c > 1$, mở rộng chân tổ chức đồ, tăng độ tương phản

Yêu cầu:

Xây dựng hàm thực thi giải thuật trượt tổ chức đồ như mô tả

function [oimg,omap] = **cangHistogram**(file, c)

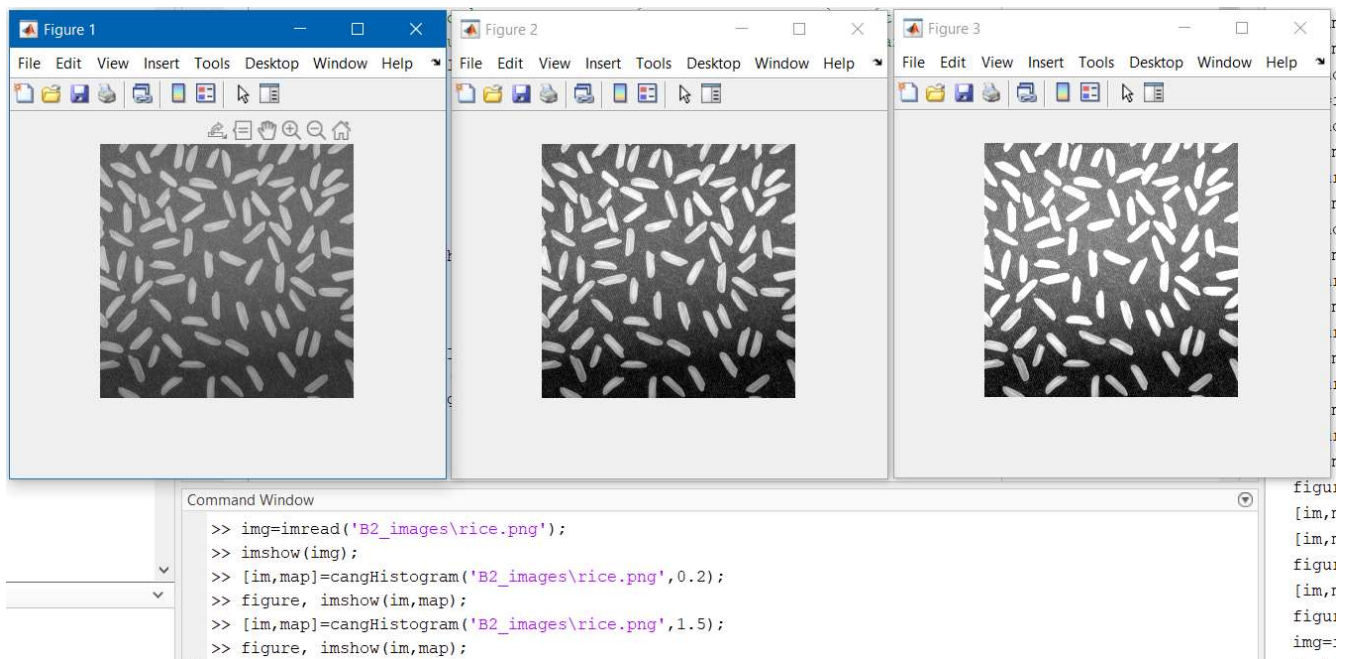
Điều kiện:

- + làm tròn giá trị mức xám về giá trị nguyên gần nhất (hàm round);
- + nếu giá trị mức xám thu được lớn hơn L_{\max} thì gán bằng L_{\max} ;

Kiểm tra tính đúng đắn của hàm cangHistogram từ Command Window.

Gợi ý: tham khảo hàm layanhamban, ở mỗi điểm ảnh đầu ra (sau khi nhân giá trị điểm ảnh đầu vào với hằng số c), làm tròn và kiểm tra giá trị điểm ảnh đầu ra thỏa điều kiện đã nêu.

Kết quả kiểm tra tính đúng đắn của hàm cangHistogram.



3.3 Biến đổi tuyến tính, nâng cao chất lượng ảnh

- Tham khảo hàm **imadjust**: <https://www.mathworks.com/help/images/ref/imadjust.html>

3.4 Cân bằng tổ chức đồ, tăng cường độ tương phản ảnh

- Tham khảo hàm **histeq**: <https://www.mathworks.com/help/images/ref/histeq.html>

3.5 Xây dựng ứng dụng cho phép thao tác với các phép xử lý tổ chức đồ

- Phát họa giao diện và hành vi của ứng dụng
- Thiết kế giao diện theo bản phát họa
- Viết code thực thi hành vi của ứng dụng

4. Các phép toán số học trên ảnh

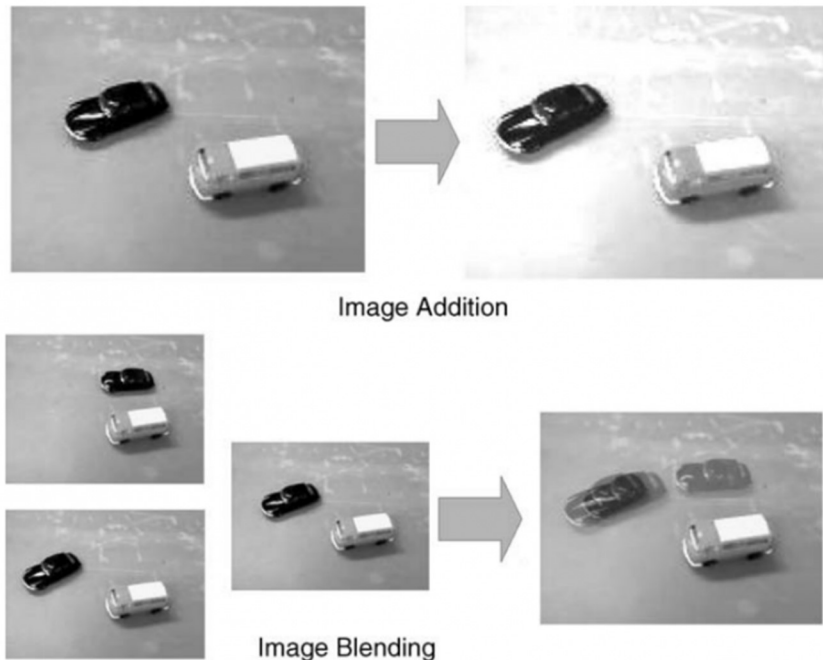
Các phép toán số học cơ bản có thể được thực hiện một cách nhanh chóng và dễ dàng trên các pixel hình ảnh cho một loạt các hiệu ứng và ứng dụng.

4.1 Cộng ảnh

- Cộng (hàm `imadd`: <https://www.mathworks.com/help/images/ref/imadd.html>) một giá trị cho mỗi điểm ảnh có thể được dùng để tạo những hiệu ứng sau:

+ *Brightness adjustment (Điều chỉnh độ sáng tối)*: Cộng một giá trị dương không đổi C làm tăng giá trị của nó từ đó tăng độ sáng tối.

+ *Blending (Pha trộn)*: Cộng 2 hình ảnh với nhau sẽ tạo ra một hình ảnh tổng hợp. Điều này có thể được sử dụng để tạo ra hiệu ứng pha trộn bằng cách dùng cộng trọng số.



- Ví dụ 1: Điều chỉnh độ sáng tối của ảnh bằng hàm `imadd`

```
>> A=imread('B2_images\cameraman.tif');
>> subplot(1,2,1), imshow(A);
>> B = imadd(A, 100);
>> subplot(1,2,2), imshow(B);
```

Kết quả thu được:



- Ví dụ 2: Pha trộn ảnh:

```
>> I = imread('B2_images\rice.png');  
>> J = imread('B2_images\cameraman.tif');  
>> K = imadd(I,J,'uint16');  
>> imshow(K,[]);
```

Kết quả thu được:



Lưu ý, nếu cộng ảnh với chính nó cũng làm tăng độ sáng của ảnh.

4.2 Trừ ảnh

- Trừ ảnh (hàm `imsubtract`: <https://www.mathworks.com/help/images/ref/imsubtract.html>): tương tự như cộng, chúng ta có thể trừ giá trị cho mỗi điểm ảnh cũng điều chỉnh độ sáng tối.

- Trừ ảnh cho chúng ta thấy phần khác biệt giữa các hình ảnh. Nếu chúng ta trừ hai hình ảnh trong một chuỗi video chúng ta sẽ có được một hình ảnh khác biệt trong đó cho thấy sự chuyển động hoặc thay đổi đã xảy ra giữa các frame trong bối cảnh đó. Điều này có thể được sử dụng như một hình thức cơ bản để phát hiện thay đổi / chuyển động trong chuỗi video.

- Ví dụ:

```
>> A=imread('B2_images\cola1.png');  
>> B=imread('B2_images\cola2.png');  
>> subplot(1,3,1), imshow(A);  
>> title('anh 1');  
>> subplot(1,3,2), imshow(B);  
>> title('anh 2');  
>> Output = imsubtract(A, B);  
>> subplot(1,3,3), imshow(Output);  
>> title('anh output');
```

Kết quả thu được:



- Một biến thể hữu ích về phép trừ là sự khác biệt tuyệt đối giữa các hình ảnh $I = |I1 - I2|$. Điều này tránh các vấn đề tiềm ẩn của tràn số nguyên. Sử dụng hàm **imabsdiff**, tham khảo theo liên kết sau: <https://www.mathworks.com/help/images/ref/imabsdiff.html>.

- Ví dụ:

```
>> A=imread('B2_images\cola1.png');
>> B=imread('B2_images\cola2.png');
>> subplot(1,3,1), imshow(A);
>> title('anh 1');
>> subplot(1,3,2), imshow(B);
>> title('anh 2');
>> Output = imabsdiff(A, B);
>> subplot(1,3,3), imshow(Output);
>> title('anh output');
```

Kết quả thu được:



4.3 Nhân và chia ảnh

- Phép nhân (hàm **immultiply**, <https://www.mathworks.com/help/images/ref/immultiply.html>) và chia (hàm **imdivide**, <https://www.mathworks.com/help/images/ref/imdivide.html>) được sử dụng như một phương thức đơn giản điều chỉnh độ tương phản và là phần mở rộng của phép cộng và trừ (ví dụ giảm độ tương phản đến 25% chia cho 4; tăng độ tương phản 50% nhân 1,5). Quá trình này đôi khi được gọi là chia tỉ lệ màu sắc hình ảnh.

- Phép chia có thể được sử dụng cho Differencing (vi phân) hình ảnh, ví dụ chia một hình ảnh với một hình ảnh khác cho kết quả là 1.0 nếu các giá trị điểm ảnh giống hệt nhau và giá trị khác 1.0 mà sự khác biệt xảy ra. Tuy nhiên, Differencing hình ảnh sử dụng phép trừ được tính

toán hiệu quả hơn. Chia và nhân 2 hình ảnh khác nhau thường không được sử dụng nhiều trong xử lý ảnh.

- Ví dụ:

```
>> A=imread('B2_images\peppers.png');  
>> subplot(1,3,1), imshow(A);  
>> title('anh goc');  
>> Output1 = immultiply(A,1.5);  
>> subplot(1,3,2), imshow(Output1);  
>> title('anh out1');  
>> Output2 = imdivide(A,4);  
>> subplot(1,3,3), imshow(Output2);  
>> title('anh out2');
```

Kết quả thu được:



- Đối với tất cả các phép tính số học giữa các hình ảnh chúng ta phải đảm bảo rằng các giá trị điểm ảnh có kết quả vẫn nằm trong phạm vi số nguyên hiện có sẵn của các kiểu dữ liệu / kích thước. Ví dụ, một hình ảnh 8-bit biểu diễn cho 256 giá trị trong mỗi vị trí pixel. Nếu giá trị điểm ảnh không nằm trong phạm vi cho phép từ 0-255 thì tràn số nguyên sẽ xảy ra và giá trị sẽ thường 'wrap around' một giá trị thấp. Điều này thường được gọi là bão hòa trong không gian ảnh: giá trị vượt quá khả năng biểu hiện của hình ảnh. Một giải pháp là để phát hiện tràn và tránh nó bằng cách thiết lập các giá trị với giá trị tối đa cho các biểu diễn hình ảnh (ví dụ cắt ngắn đến 255). Phương pháp xử lý tràn được thực hiện trong imadd, imsubtract, immultiply và imdivide.

- Đối với ba kênh hình ảnh RGB các phép toán toán học thường được thực hiện riêng cho từng kênh màu.

4.4 Xây dựng ứng dụng cho các phép toán số học trên ảnh

- Phát họa giao diện và hành vi của ứng dụng
- Thiết kế giao diện theo bản phát họa
- Viết code thực thi hành vi của ứng dụng
- Lưu ý, các phép toán số học đa phần được áp dụng trên nhiều ảnh.

5. Các phép toán luận lý trên ảnh

- Tìm hiểu và thao tác với các hàm: bitand, bitor, bitxor và bitcmp.
- Xây dựng ứng dụng cho các phép toán luận lý trên ảnh. Lưu ý, các phép toán luận lý đa phần được áp dụng trên nhiều ảnh.