MATLAB PROJECT ON

PHOTO EDITING SOFTWARE

INDEX

INTRODUCTION TO IMAGE PROCESSING USING MATLAB R2011A

Digital image processing is the use of computer algorithms to create, process, communicate, and display digital images. Digital image processing algorithms can be used to:

- Convert signals from an image sensor into digital images
- Improve clarity, and remove noise and other artifacts
- Extract the size, scale, or number of objects in a scene
- Prepare images for display or printing
- Compress images for communication across a network

Effective techniques for processing digital images include using algorithms and tools that provide a comprehensive environment for data analysis, visualization, and algorithm development.

Photo Editing Using MATLAB

There are several MATLAB functions present in image processing toolbox of MATLAB and using the same to create a basic image processor having different features like,

- Viewing the red, green and blue components of a color image separately
- Color detection
- Noise addition and removal
- Edge detection
- Cropping
- Resizing
- Rotation
- Histogram adjust
- Brightness control, etc.

which are used in a basic image editor along with object detection and tracking.

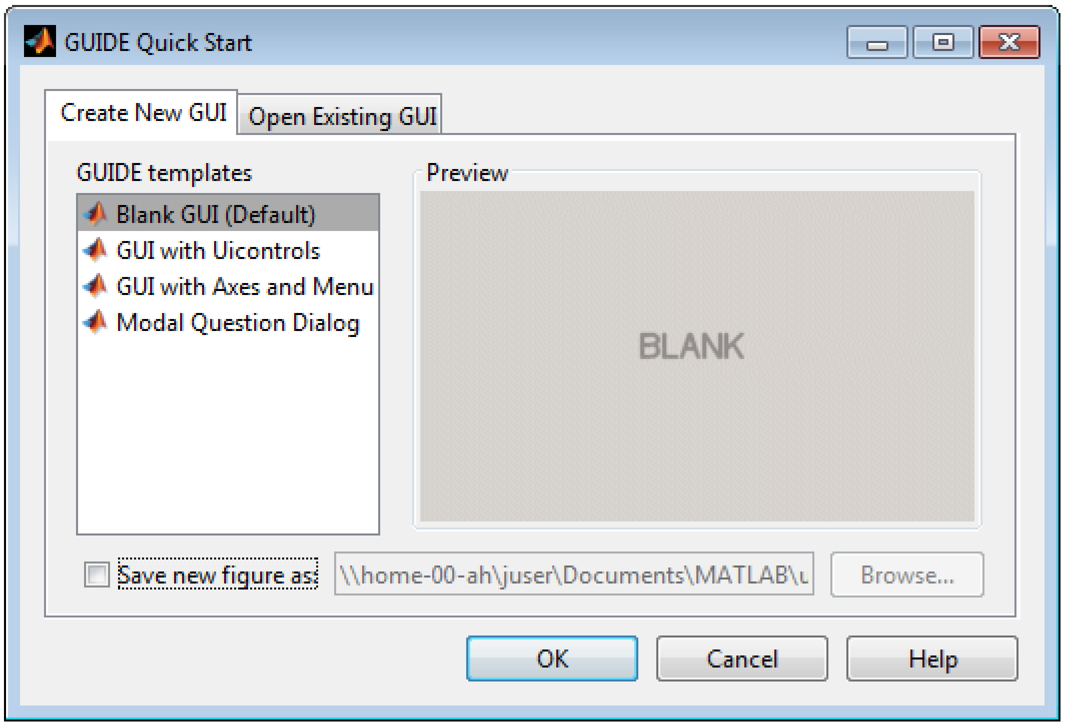
CREATING GUI USING MATLAB GUIDE

MATLAB apps are self-contained MATLAB programs with GUI front ends that automate a task or calculation. The GUI typically contains controls such as menus, toolbars, buttons, and sliders. Many MATLAB products, such as Curve Fitting Toolbox, Signal Processing Toolbox, and Control System Toolbox, include apps with custom user interfaces. You can also create your own custom apps, including their corresponding UIs, for others to use.

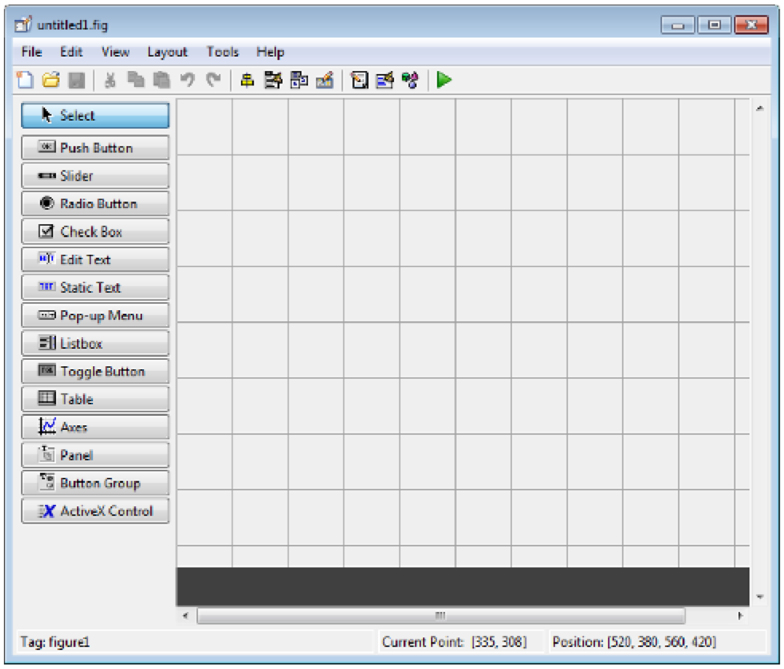
GUIDE (graphical user interface design environment) provides tools for designing user interfaces for custom apps. Using the GUIDE Layout Editor, you can graphically design your UI. GUIDE then automatically generates the MATLAB code for constructing the UI, which you can modify to program the behavior of your app.

Steps to create GUI using MATLAB GUIDE :

1. Start GUIDE by typing guide at the MATLAB prompt.

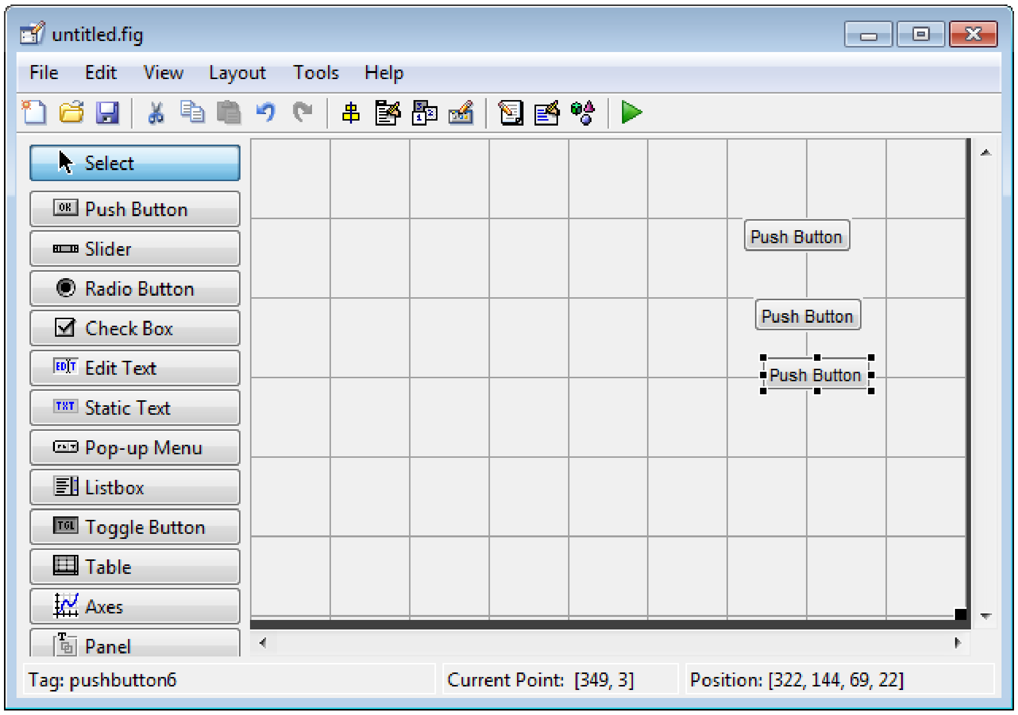


2. In the GUIDE Quick Start dialog box, select the **Blank (Default)** template, and then click **OK**.

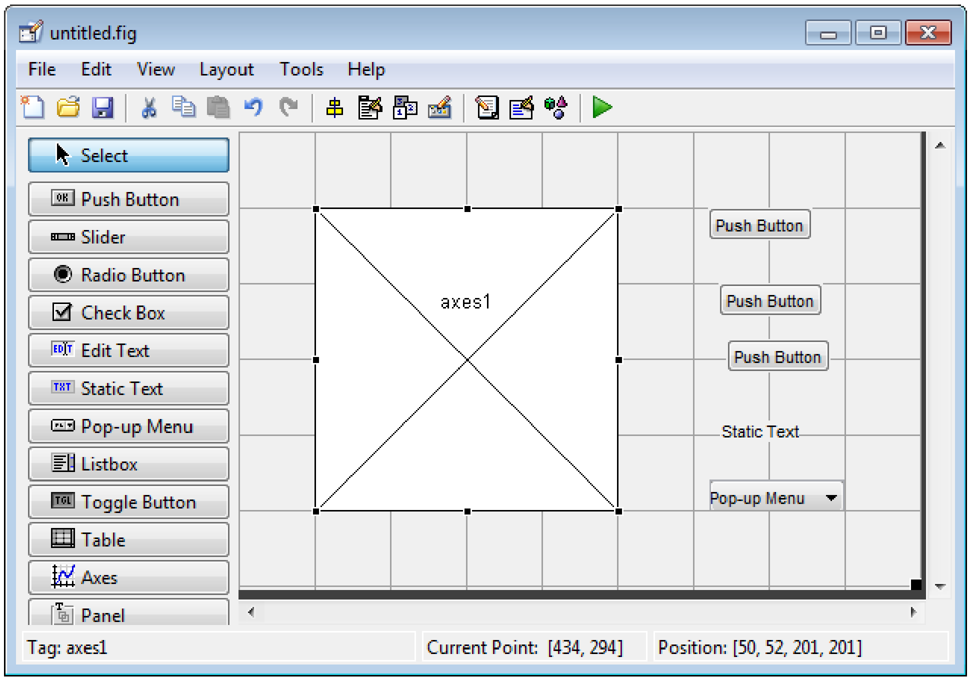


Tag: figure1 Current Point: [335, 308] Position: [520, 380, 560, 420]

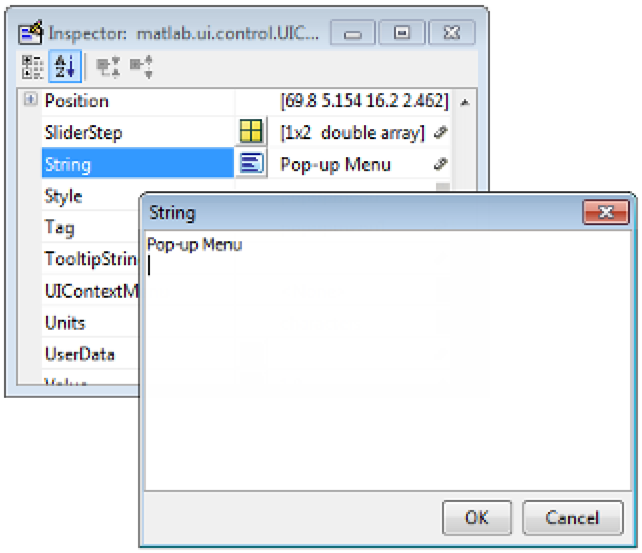
3. Select the **Push button** tool from the component palette at the left side of the Layout Editor and drag it into the layout area. Similarly create the axes using **Axes** tool and a Text Box using **Edit Text** tool



Tag: pushbutton6 Current Point: [349, 3] Position: [322, 144, 69, 22]



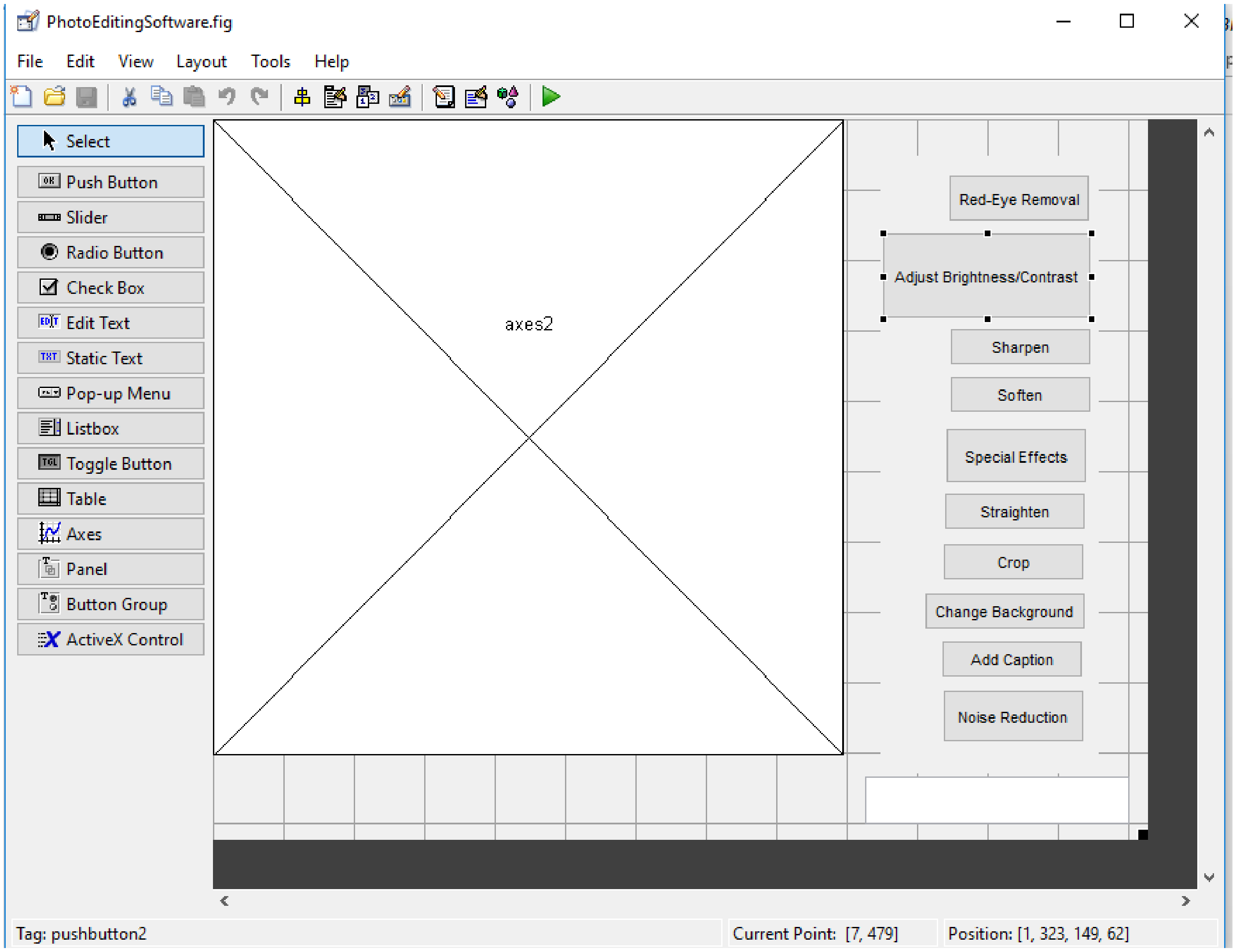
4. To change the labels of Push Buttons, double click on them to open **Property Inspector.** Scroll down to **String** property, click on the button beside it and write the required label.

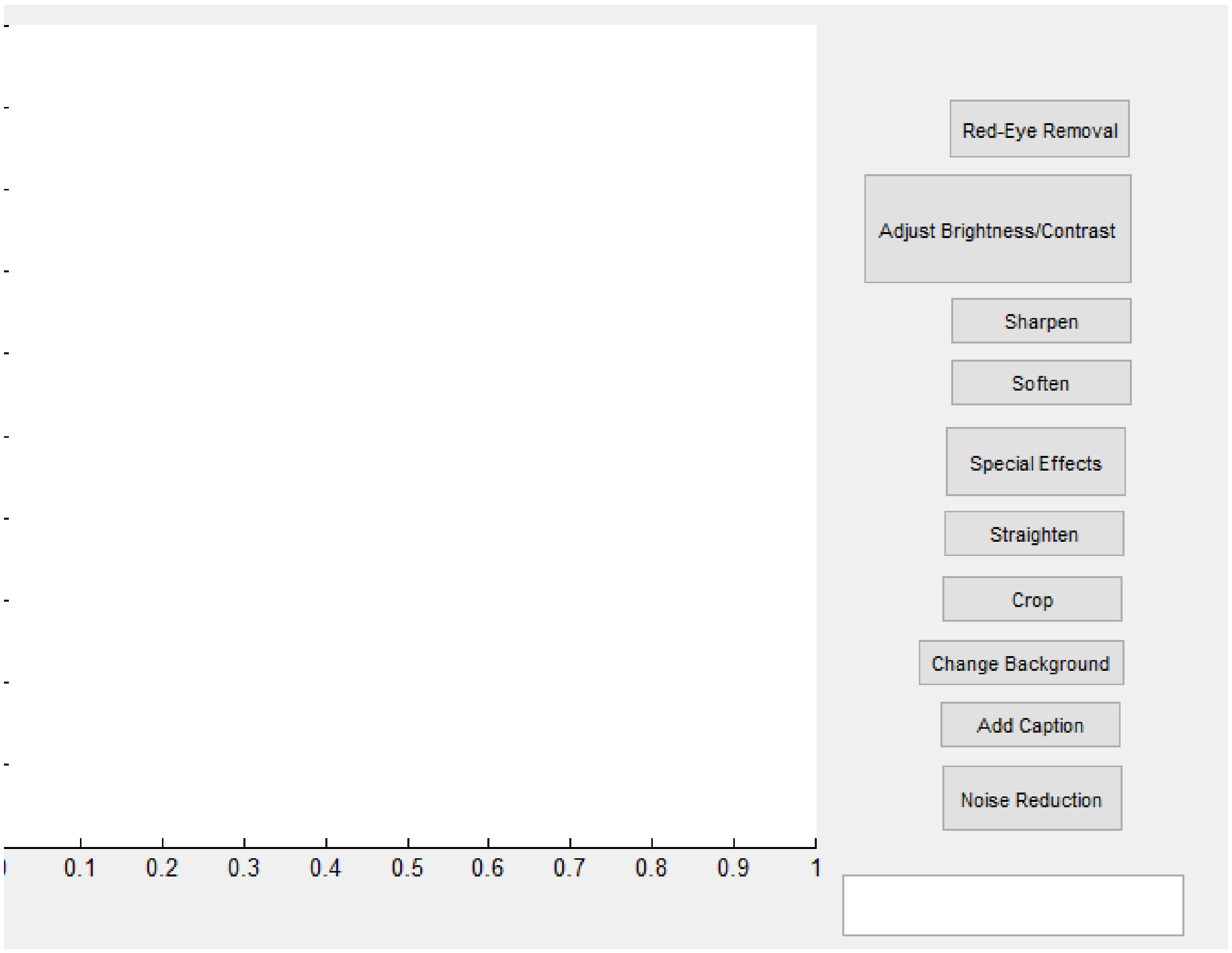

Inspector: matlab.ui.control.UIC...
Position [69.8 5.154 16.2 2.462]
SliderStep [1x2 double array]
String Pop-up Menu
Style
Tag
TooltipStrin
UIContextM
Units
UserData
String
Pop-up Menu
OK
Cancel

5. To run your GUI, save and run your program by selecting **Tools** > **Run**.
   a. To run an app created with GUIDE without opening GUIDE, execute its code file by typing its name.
   simple_gui
   b. You can also use the run command with the code file, for example,
   run simple_gui

6. If the file `simple_gui.m` is not already open in the editor, open from the Layout Editor by selecting **View** > **Editor**.

By following the above specified steps, I created the following GUI for my Photo Editing Software :





Various Push Buttons on the right, implement various features of the Photo Editing Software. The implementation and output of the software is described under :

- **Photo Editing Software Implementation**
- **Photo Editing Software : Output of various features**

MATLAB FUNCTIONS USED IN THIS PROJECT

1. **x = input('Any String');**
   Displays the text in `prompt` and waits for the user to input a value and press the **Return** key.

2. **Image(I)**
   Where I = image read using imread() function.
   Displays the data in array `I` as an image

3. **P = ginput(num)';**
   Raises crosshairs in the current axes to for you to identify points in the figure, positioning the cursor with the mouse. The figure must have focus before `ginput` can receive input. If it has no axes, one is created upon the first click or keypress.

4. **A = imadjust(I);**
   Maps the intensity values in grayscale image `I` to new values in `J` such that 1% of data is saturated at low and high intensities of `I`.
   - **J = imadjust(I,[low_in; high_in],[low_out; high_out]);**
     Maps the values in `I` to new values in `J` such that values between `low_in` and `high_in` map to values between `low_out` and `high_out`.

5. **J= wiener2(I,[m n]);**
   Filters the image `I` using pixelwise adaptive Wiener filtering, using neighborhoods of size `m`-by-`n` to estimate the local image mean and standard deviation. If you omit the `[m n]` argument, `m` and `n` default to 3.

6. **J = flipdim(I,dim);**
   When the value of dim is 1, the array is flipped row-wise down. When dim is 2, the array is flipped columnwise left to right. flipdim(A,1) is the same as flipud(A), and flipdim(A,2) is the same as fliplr(A).

7. **J = imrotate(I,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.

8. **BW = im2bw(I);**
   Converts the grayscale image `I` to a binary image.

9. **J = imclearborder(I);**
   Suppresses structures that are lighter than their surroundings and that are connected to the image border. Use this function to clear the image border. `IM` can be a grayscale or binary image.

10. **T = cp2tform(movingPoints,fixedPoints, transformtype);**
    Infers a spatial transformation from control point pairs and returns this transformation as a `T` structure.

11. **I2 = imcrop(I);**
    Displays the image `I` in a figure window and creates a cropping tool associated with that image. `I` can be a grayscale image, a truecolor image, or a logical array. `imcrop` returns the cropped image, `I2`.
    - **I2 = imcrop(I,rect);**
      Crops the image `I`. `rect` is a four-element position vector of the form `[xmin ymin width height]` that specifies the size and position of the crop rectangle. `imcrop` returns the cropped image, `I2`.

12. **I2 = im2double(I);**
    Converts the intensity image `I` to double precision, rescaling the data if necessary. `I` can be a grayscale intensity image, a truecolor image, or a binary image.

13. **Package: vision**
    Draw text on image or video stream.
    - **txtInserter = vision.TextInserter('string') ;**
      Returns a text inserter object, txtInserter with the input text provided in quotes added to the image.

14. **J = imnoise(I,*type*);**
    Adds noise of a given type to the intensity image `I`. *type* is a string that specifies any of the following types of noise.
    - **J = imnoise(I,'gaussian',M,V);**
      Adds Gaussian white noise of mean `m` and variance `v` to the image `I`. The default is zero mean noise with 0.01 variance.

There are various types of noises as described in the table below :

Value	Description
`'gaussian'`	Gaussian white noise with constant mean and variance
`'localvar'`	Zero-mean Gaussian white noise with an intensity-dependent variance
`'poisson'`	Poisson noise
`'salt & pepper'`	On and off pixels
`'speckle'`	Multiplicative noise

PHOTO EDITING SOFTWARE IMPLEMENTATION

Following is the code has been implemented using MATLAB R2011a :

```
function varargout = PhotoEditingSoftware(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @PhotoEditingSoftware_OpeningFcn, ...
                   'gui_OutputFcn',  @PhotoEditingSoftware_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
end

% --- Executes just before PhotoEditingSoftware is made visible.
function PhotoEditingSoftware_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
end

% --- Outputs from this function are returned to the command line.
function varargout = PhotoEditingSoftware_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
end



% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
    % Red-Eye Removal Feature
    plot(3,3);
    I=imread('C:/Users/Parul/Desktop/redeye.jpg');
    plot(3,3);
```

```
   image(I);
   P = ginput(4)';
   Q = ginput(4)';
   red_threshold = 100;
   J=I;
   for i = int16(min(P(2,:))):1:int16(max(P(2,:)))
   for j = int16(min(P(1,:))):1:int16(max(P(1,:)))
   c = I(i,j,:);
   red=c(1);
   if red > red_threshold
   J(i,j,:)=0;
   end
   end
   end
   for i = int16(min(Q(2,:))):1:int16(max(Q(2,:)))
   for j = int16(min(Q(1,:))):1:int16(max(Q(1,:)))
   c = I(i,j,:);
   red=c(1);
   if red > red_threshold
   J(i,j,:)=0;
   end
   end
   end
   subplot(2,1,1)
   imshow(I);title('Red Eye');
   subplot(2,1,2)
   imshow(J);title('No Red Eye');
end


% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
   % Adjust Brightness and Contrast
   plot(3,3);
   C = get(handles.edit1,'String');
   C = str2num(C);
    if isempty(C)
       h = msgbox('Specify adjustment values (Eg1:[.2,.3]  Eg2:[.2 .3 0; .6 .7 1])');
       return
    end
   A =imread('C:/Users/Parul/Desktop/bird.jpg');
   I = rgb2gray(A);
   I = imadjust(I);
```

```
    subplot(3,3,[1,2])
    imshow(A);title('Original Image');
    subplot(3,3,[4,5])
    imshow(I);title('Adjusted Image 1 (B/W)');
    I2 =  imadjust(A, C, []);
    subplot(3,3,[7,8])
    imshow(I2);title('Adjusted Image 2 (Colored)');
end



% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
    % Sharpening
    plot(3,3);
    T = get(handles.edit1,'String');

     if isempty(T)
         h = msgbox('Specify type of sharpening you want to apply (Eg:average, disk, gaussian,
laplacian, log, motion, prewitt, sobel)');
         return
     end
    f1 =imread('C:/Users/Parul/Desktop/bird.jpg');
    f1 = rgb2gray(f1);
    subplot(2,5,[1,1.99])
    imshow(f1); title('Original image');
    subplot(2,5,[3.21,4])
    imhist(f1); title('Original image histogram');
    w4=fspecial(T);
    f2=im2double(f1);
    g4=f2-imfilter(f2, w4, 'replicate');
    subplot(2,5,[6,6.99])
    imshow(g4); title('Sharpened Image');
    subplot(2,5,[8.21,9])
    imhist(g4); title('Sharpened image histogram');
end



% --- Executes on button press in pushbutton4.
function pushbutton4_Callback(hObject, eventdata, handles)
    %Softening
    plot(3,3);
    V = get(handles.edit1,'String');
    V = str2num(V);
     if isempty(V)
         h = msgbox('Specify values for softening (Eg1:[10,10])  (Eg2: [30,30])');
         return
     end
    f1 =imread('C:/Users/Parul/Desktop/bird.jpg');
    f1 = rgb2gray(f1);
    subplot(2,5,[1,1.99])
    imshow(f1); title('Original image');
    subplot(2,5,[3.21,4])
    imhist(f1); title('Original image histogram');
    K = wiener2(f1,V);
    subplot(2,5,[6,6.99])
    imshow(K); title('Softened Image');
    subplot(2,5,[8.21,9])
    imhist(K); title('Softened image histogram');
end



% --- Executes on button press in pushbutton5.
function pushbutton5_Callback(hObject, eventdata, handles)
    % Special Effects
    plot(3,3);
     option = get(handles.edit1,'String');
    n = str2num(option);
     if isempty(option)
         h = msgbox({'1. Flip' '2. Rotate|AOR' ''});
         return
     end
    I =imread('C:/Users/Parul/Desktop/bird.jpg');
    if strcmp(option,'Flip') | n==1
        I2 = flipdim(I,2);
        I3 = flipdim(I,1);
        I4 = flipdim(I3,2);
        subplot(2,3,1)
        imshow(I); title('Original Image');
        subplot(2,3,2)
        imshow(I2); title('Flipped Image Horizontally');
        subplot(2,3,4)
        imshow(I3); title('Flipped Image Vertically');
        subplot(2,3,5)
        imshow(I4); title('Flipped Image Vertically/Horizontally');
    end

    if strfind(option, 'Rotate')
        str = strtrim(option,'|');
        angle = str2num(str(2));
        I2 = imrotate(I,angle);
        subplot(2,3,[1,2])
        imshow(I); title('Original Image');
        subplot(2,3,[4,5])
        imshow(I2); title('Rotated image');
    end

if strcmp(option,'Negative') | n==3
    A =imread('C:/Users/Parul/Desktop/bird.jpg');
    negimg = imcomplement(A);
    subplot(2,3,[1,2])
    imshow(A); title('Original Image');
    subplot(2,3,[4,5])
    imshow(negimg); title('Negative image');
end

if strcmp(option,'Collage') | n==4
    %I1 =imread('C:/Users/Parul/Desktop/bird.jpg');
    I2 =imread('C:/Users/Parul/Desktop/bird2.jpg');
    I3 =imread('C:/Users/Parul/Desktop/bird3.jpg');
    I4 =imread('C:/Users/Parul/Desktop/bird4.jpg');
    plotSize = [256,256];
    %A = plotSize./[2,1];
    B = plotSize./[2,1];
    C = plotSize./[2,2];
    D = plotSize./[2,2];
    collImg = [imresize(I2,B);imresize(I3,C),imresize(I4,D)];
    imshow(collImg);title('Collage');
end

if strcmp(option,'Glassy Effect') | n==5
     A =imread('C:/Users/Parul/Desktop/bird.jpg');
    m=6;
    n=7;
    Image=uint8(zeros([size(A,1)-m,size(A,2)-n,3]));
    for i=1:size(A,1)-m
        for j=1:size(A,2)-n
```

```
            mymask=A(i:i+m-1,j:j+n-1,:);
            %Select a pixel value from the neighborhood.
            x2=ceil(rand(1)*m);
            y2=ceil(rand(1)*n);
            Image(i,j,:)=mymask(x2,y2,:);
        end
    end
    subplot(2,3,[1,2])
    imshow(A); title('Original Image');
    subplot(2,3,[4,5])
    imshow(Image); title('Glassy image');
 end
end


% --- Executes on button press in pushbutton6.
function pushbutton6_Callback(hObject, eventdata, handles)
    % Straighten a picture

    %Straightening 1st image
    plot(3,3);
    I = imread('C:/Users/Parul/Desktop/rect.jpg');
    I=rgb2gray(I);
    subplot(2,3,1)
    imshow(I); title('Tilted Rectangle 1');
    I2 = imclearborder(im2bw(I));
    [y,x] = find(I2);
    [~,loc] = min(y+x);
    C = [x(loc),y(loc)];
    [~,loc] = min(y-x);
    C(2,:) = [x(loc),y(loc)];
    [~,loc] = max(y+x);
    C(3,:) = [x(loc),y(loc)];
    [~,loc] = max(y-x);
    C(4,:) = [x(loc),y(loc)];
    L = mean(C([1 4],1));
    R = mean(C([2 3],1));
    U = mean(C([1 2],2));
    D = mean(C([3 4],2));
    C2 = [L U; R U; R D; L D];
    T = cp2tform(C ,C2,'projective');
    IT = imtransform(im2bw(I),T);
    subplot(2,3,4)
    imshow(IT);title('Straight Rectangle 1');

    %Straightening 2nd image
    I = imread('C:/Users/Parul/Desktop/rect2.jpg');
    I=rgb2gray(I);
    subplot(2,3,2)
    imshow(I); title('Tilted Rectangle 2');
    I2 = imclearborder(im2bw(I));
    [y,x] = find(I2);
    [~,loc] = min(y+x);
    C = [x(loc),y(loc)];
    [~,loc] = min(y-x);
    C(2,:) = [x(loc),y(loc)];
    [~,loc] = max(y+x);
    C(3,:) = [x(loc),y(loc)];
    [~,loc] = max(y-x);
    C(4,:) = [x(loc),y(loc)];
    L = mean(C([1 4],1));
    R = mean(C([2 3],1));
    U = mean(C([1 2],2));
    D = mean(C([3 4],2));
    C2 = [L U; R U; R D; L D];
    T = cp2tform(C ,C2,'projective');
    IT = imtransform(im2bw(I),T); %IM2BW is not necessary
    subplot(2,3,5)
    imshow(IT); title('Straight Rectangle 2');
end



% --- Executes on button press in pushbutton7.
function pushbutton7_Callback(hObject, eventdata, handles)
    % Crop a picture
    plot(3,3);
    C = get(handles.edit1,'String');
    C = str2num(C);
     if isempty(C)
        h = msgbox('Enter the coordinates for cropping   (Eg: [75 68 130 112])');
        return
     end
    i=imread('C:/Users/Parul/Desktop/bird.jpg');
    subplot(2,3,[1,2])
    imshow(i);title('Original image');
    [I2, rect] = imcrop(i);
    subplot(2,3,4)
    imshow(I2);title('Cropped Image 1');
    I3 = imcrop(i,C);
    subplot(2,3,5)
    imshow(I3);title('Cropped Image 2');

end



% --- Executes on button press in pushbutton8.
function pushbutton8_Callback(hObject, eventdata, handles)
    % Change the background of image
    plot(3,3);
    color = get(handles.edit1,'String');
    color = str2num(color);
     if isempty(color)
         h = msgbox('Enter the coordinates of color you want to change background color to  (Eg:
[255 128 127])');
         return
     end

    A = imread('C:/Users/Parul/Desktop/black.jpg');
    subplot(2,3,1)
    imshow(A); title('Black background');
    [nRow,nCol,nColor] = size(A);
    for i = 1:nRow
    for j = 1:nCol
    c = A(i,j,:);
    if c==0  | c==255 %Background color is originally black or white
    A(i,j,1)=color(1);
    A(i,j,2)=color(2);
    A(i,j,3)=color(3);
    end
    end
    end
    subplot(2,3,4)
    imshow(A); title('Background Change');

    A = imread('C:/Users/Parul/Desktop/white.jpg');
    subplot(2,3,2)
    imshow(A); title('White background');
```

```
   [nRow,nCol,nColor] = size(A);
   for i = 1:nRow
   for j = 1:nCol
   c = A(i,j,:);
   if c==0  | c==255 %Background color is originally black or white
   A(i,j,1)=color(1);
   A(i,j,2)=color(2);
   A(i,j,3)=color(3);
   end
   end
   end
   subplot(2,3,5)
   imshow(A); title('Background Change');
end



% --- Executes on button press in pushbutton9.
function pushbutton9_Callback(hObject, eventdata, handles)
   % Add a Caption
   plot(3,3);
   str = get(handles.edit1,'String');
   str = strtrim(str,'|');
    if isempty(str)
       h = msgbox('Enter Caption|Color|Size|Position  (Eg: [Title|1.0 0.9 0.8|127|45 34])');
       return
    end
   A = im2double((imread('C:/Users/Parul/Desktop/bird.jpg')));
   subplot(2,3,[1,2])
   imshow(A); title('Image without Caption');
   Text = str(1);
   H = vision.TextInserter(Text);
   H.Color = str2double(str(2));
   H.FontSize = str2double(str(3));
   H.Location = str2double(str(4));
   %Birdieeee|0.9 0.7 0.4|123|45 67
   %H = vision.TextInserter('Birdieeee');
   %H.Color = [0.9 0.7 0.4];
   %H.FontSize = 123;
   %H.Location = [45 67];
   I = im2double((imread('C:/Users/Parul/Desktop/bird.jpg')));
   CI = step(H, I);
   subplot(2,3,[4,5])
```

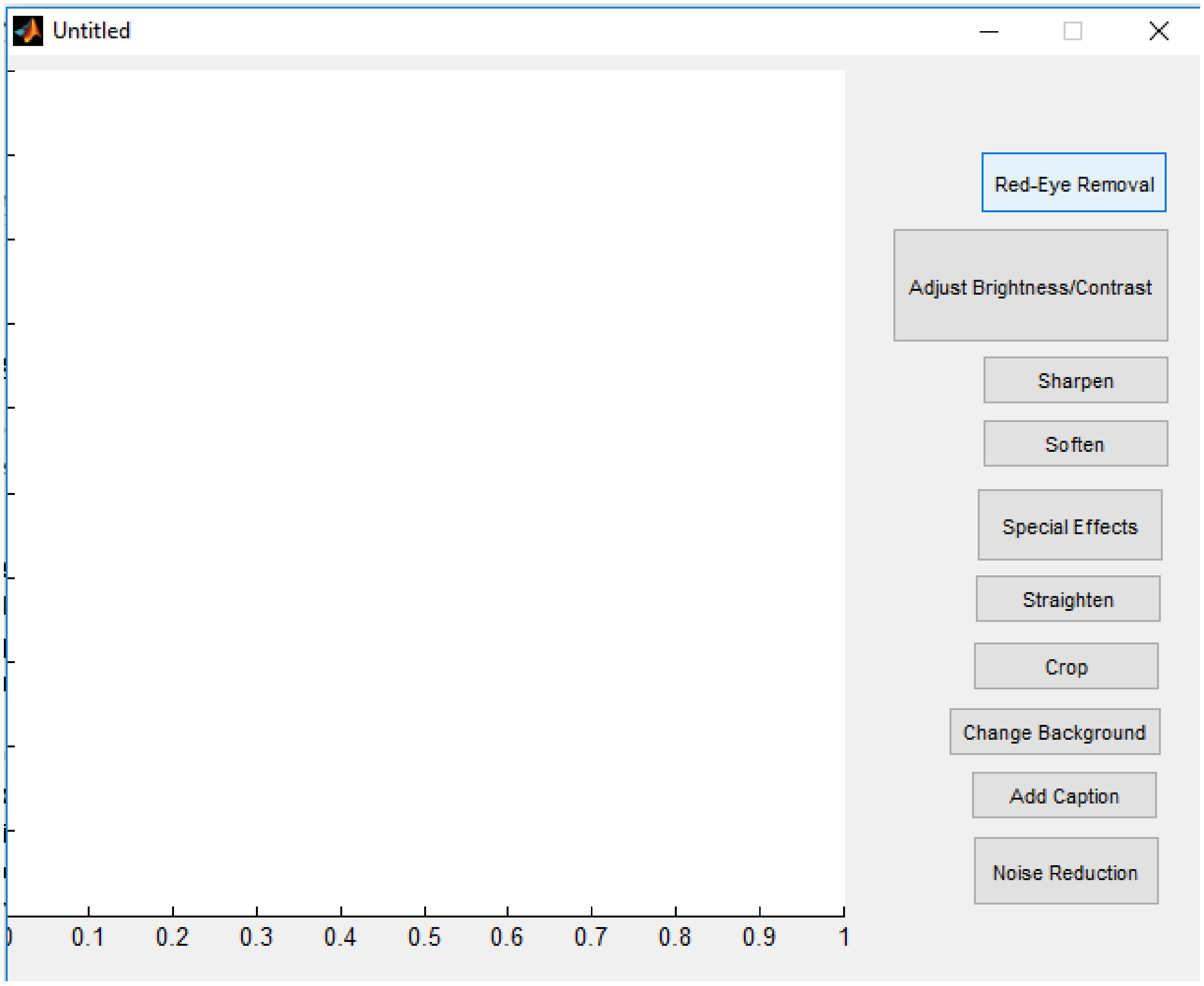
```
    imshow(CI); title('Image with Caption');

end



% --- Executes on button press in pushbutton10.
function pushbutton10_Callback(hObject, eventdata, handles)
    % Noise Reduction
    plot(3,3);
    fprintf('\n\nNoise Reduction\n\n')
    A =imread('C:/Users/Parul/Desktop/bird.jpg');
    A = rgb2gray(A);
    subplot(2,3,[1.56,2])
    imshow(A);title('Original Image');
    N = imnoise(A,'gaussian',0,0.01);
    subplot(2,3,[4,4.30])
    imshow(N);title('Image with noise');
    NR = wiener2(N,[5,5]);
    subplot(2,3,[5.22,5.51])
    imshow(NR); title('Image after Noise Reduction');
end



function edit1_Callback(hObject, eventdata, handles)
end


% --- Executes during object creation, after setting all properties.
function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'), get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
end
```

PHOTO EDITING SOFTWARE – OUTPUT OF VARIOUS FEATURES

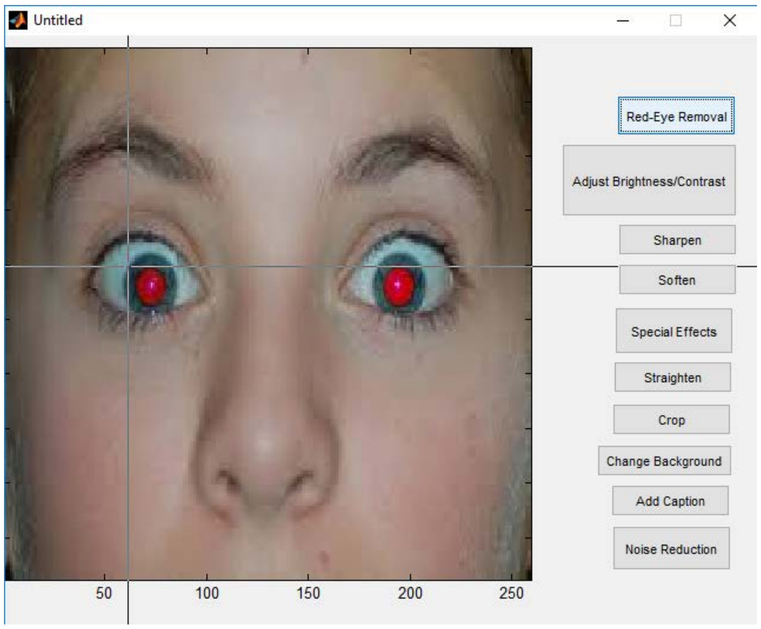
1. Red-Eye Removal

This feature enables the user to mark rectangular regions around eyes, in order to remove the red color from them. By executing the provided code we get the following output :

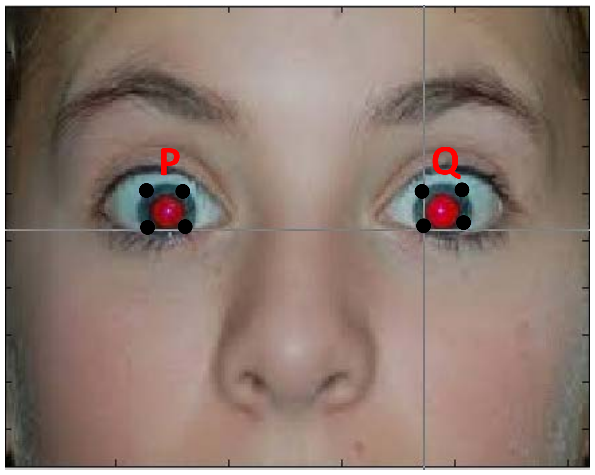
1. Click on Red-Eye Removal Push Button. The image appears on which are going to mark the two rectangles around the eyes.



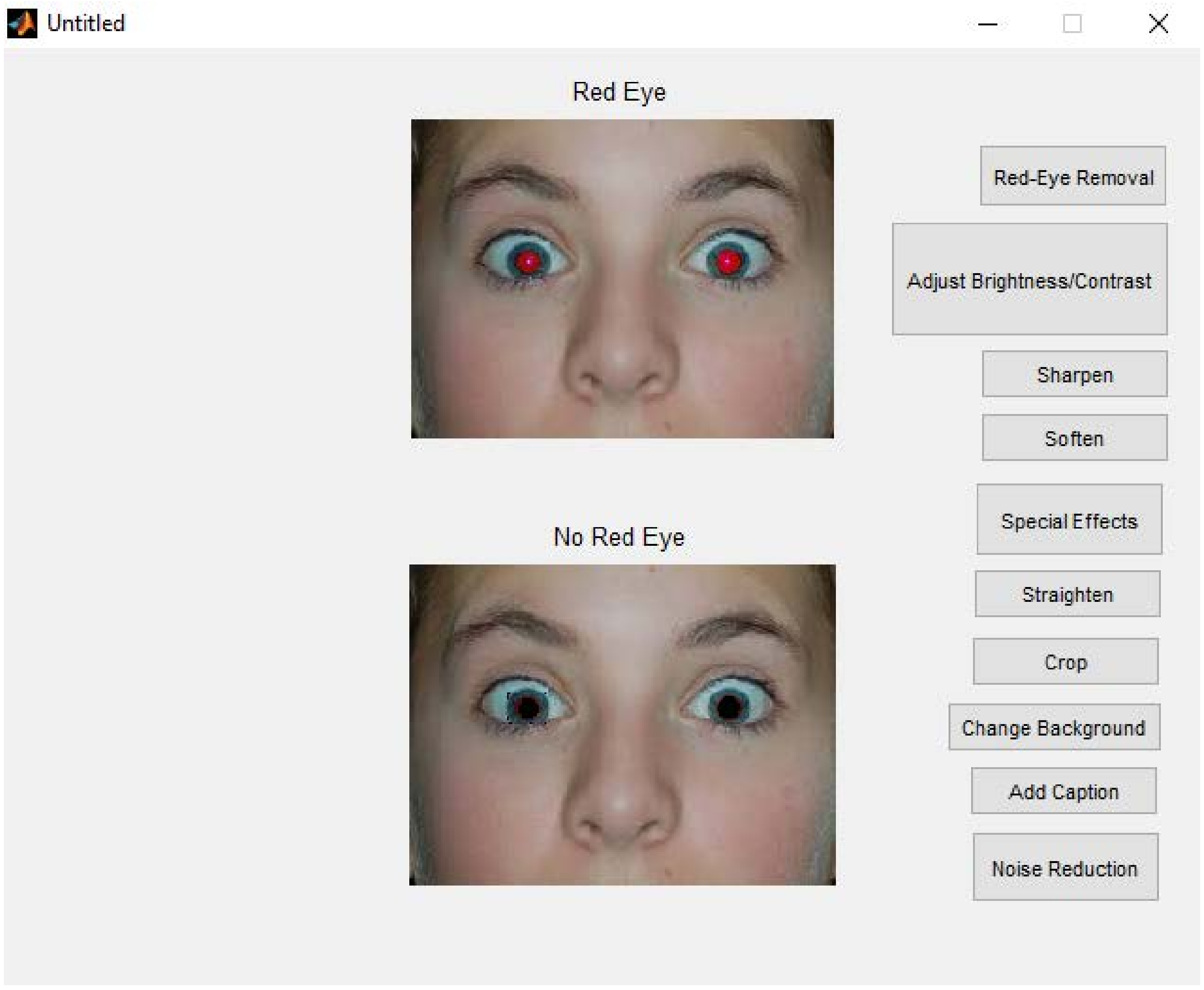
We mark the two rectangles P and Q around the red eyes, by clicking with the mouse on 4 corners of the rectangles to be formed. Below is the figure describing how to form the rectangles :



Using above action, the two rectangles P and Q, with four coordinates each, will be formed as following :



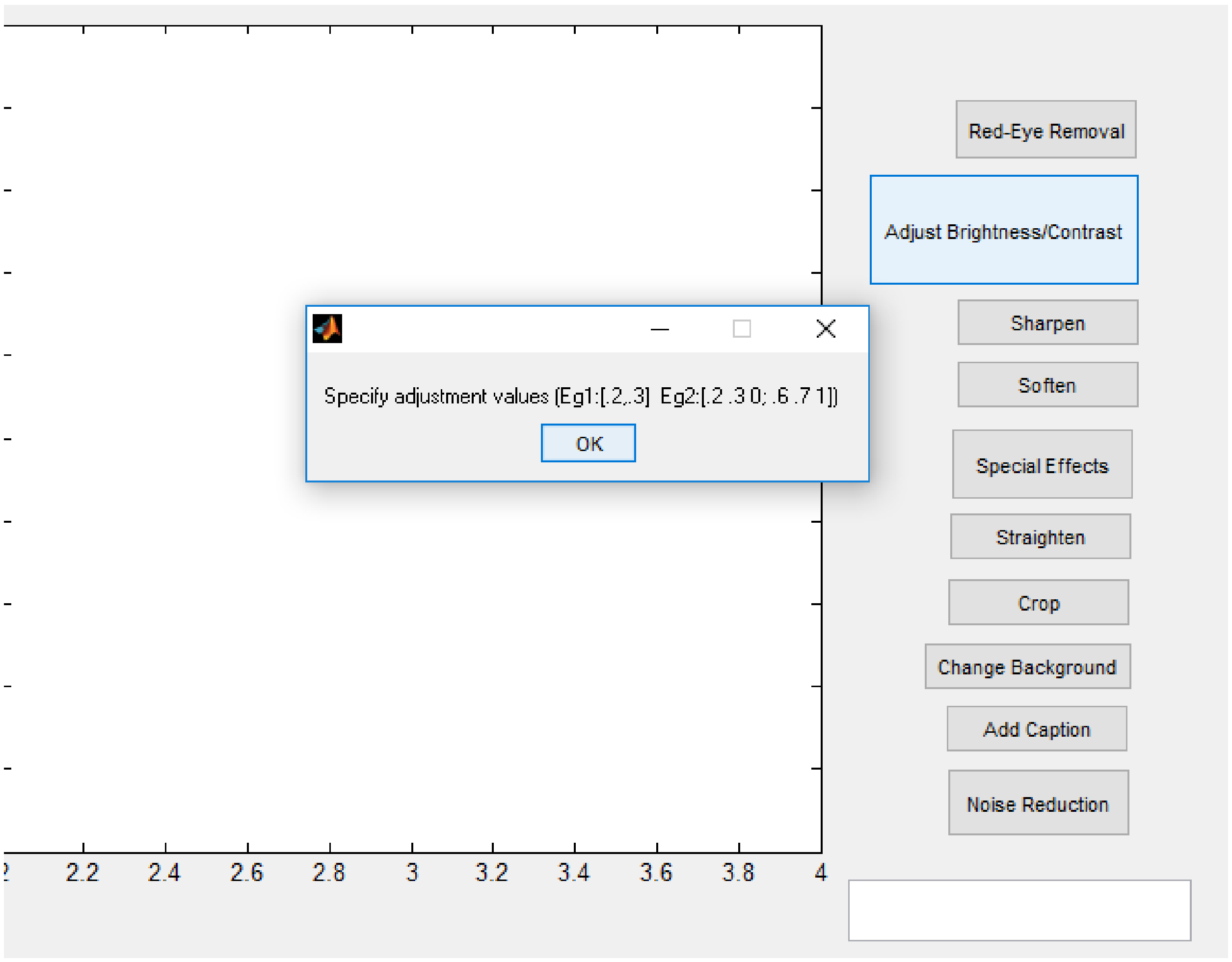
After done marking the two rectangles, the following figure is generated as the result of 'Red Eye Removal' feature :



2. Adjust Brightness and Contrast

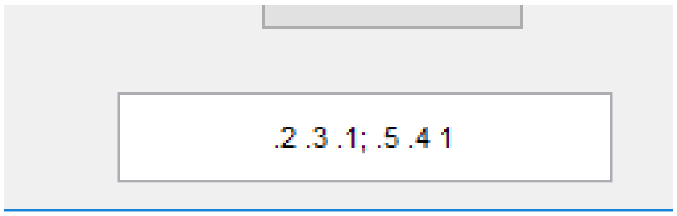
This feature enables the user to adjust brightness and contrast of the provided image by specifying the adjustment values for **imadjust()** function of MATLAB. By executing the provided code we get the following output :

When we click on **Adjust Brightness/Contrast**, we get the following pop up :



Specify adjustment values [Eg1:[.2,.3] Eg2:[.2 .3 0; .6 .7 1]]

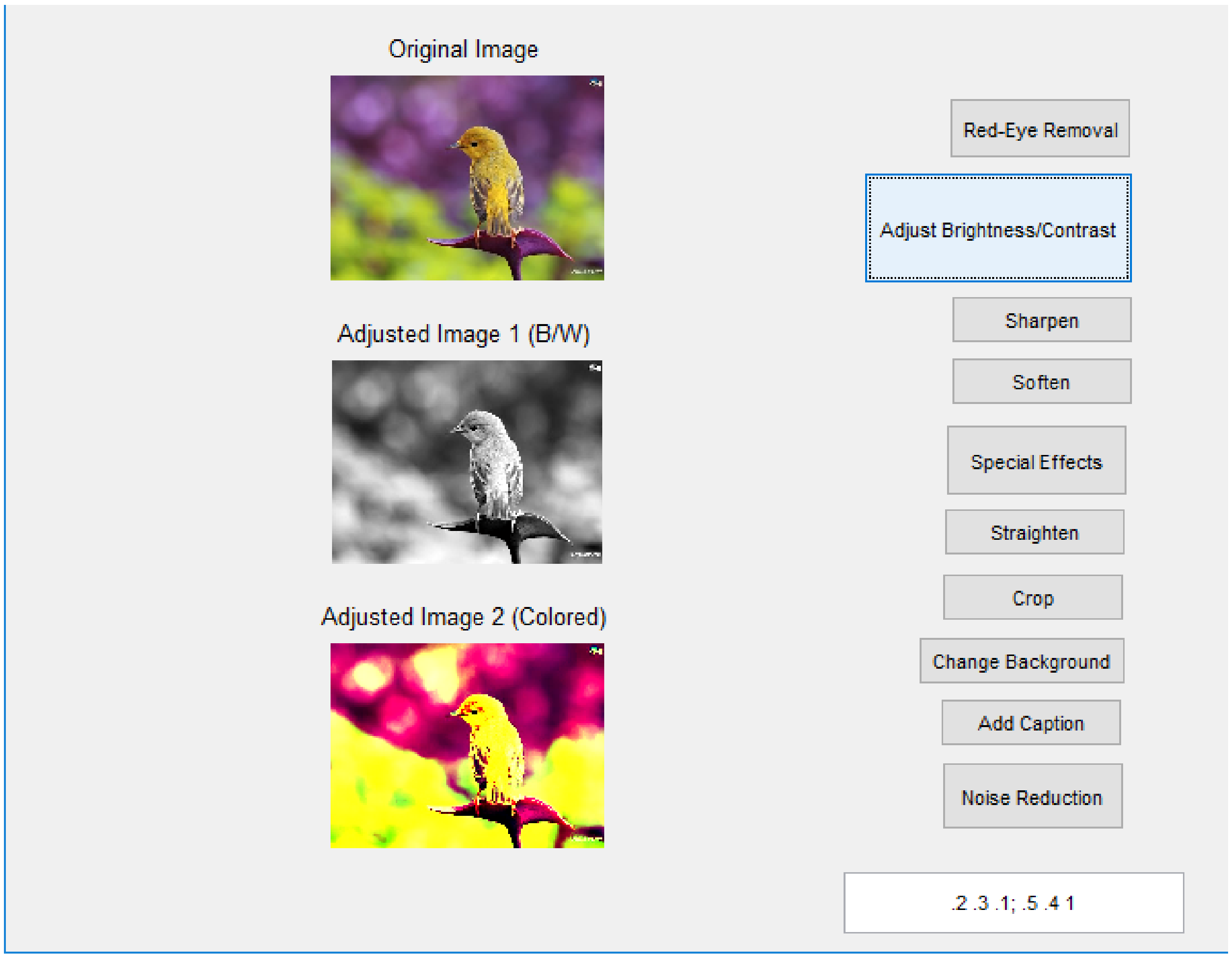
We specify the adjustment values in the textbox present in lower-right corner.



.2 .3 .1; .5 .4 1

The black and white figure is generated upon using **imadjust()** function without any parameters specified. The imadjust() function with no parameters works only for Grayscale images.

The other image is generated when user entered the adjustment values of `[low_in; high_in]`



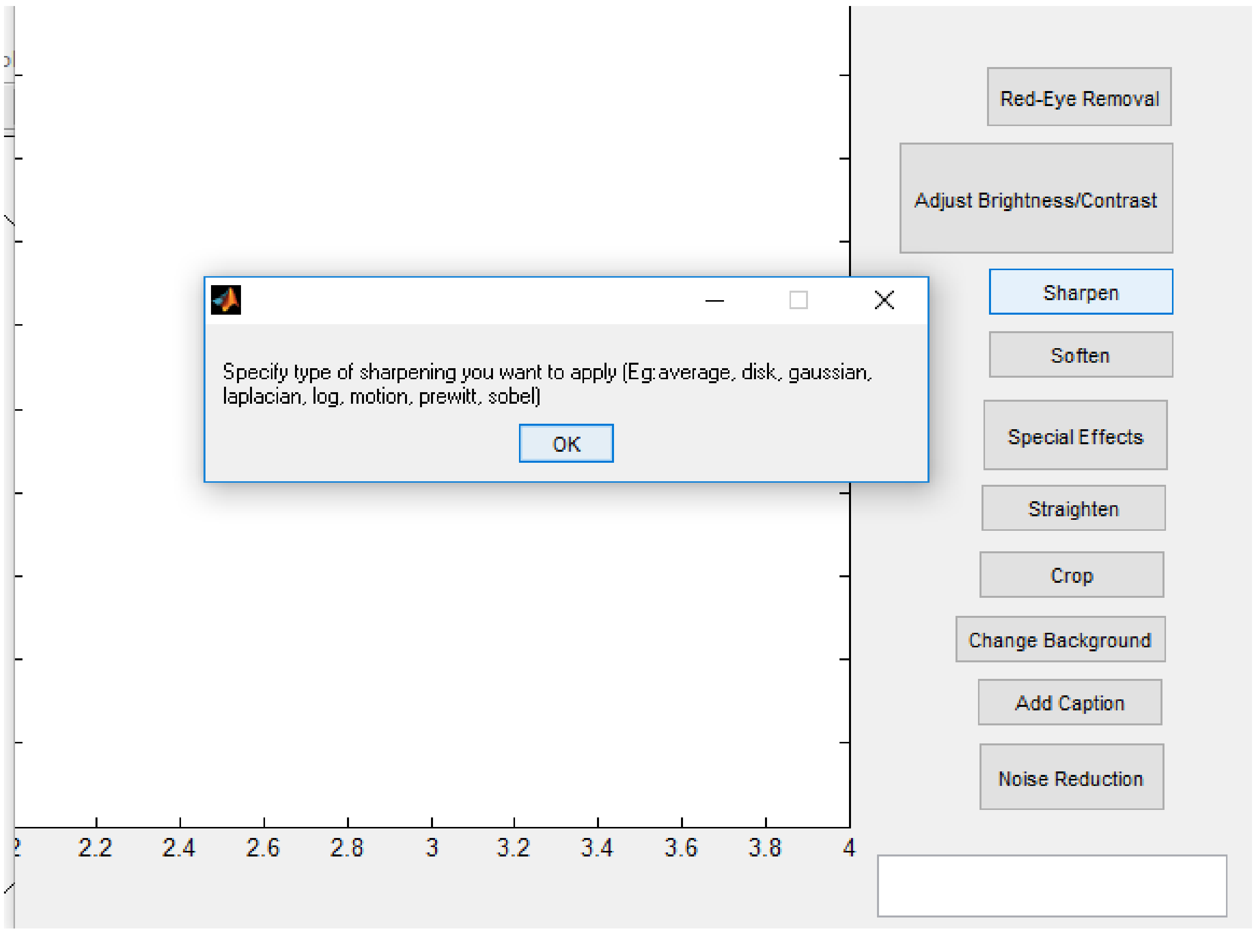
.2 .3 .1; .5 .4 1

3. Sharpening and Softening Effects on Images

This feature enables the user to sharpen an image using fspecial() function and its type parameter and soften it using weiner2() function by specifying the integer array [m,n]. By executing the provided code we get the following output :

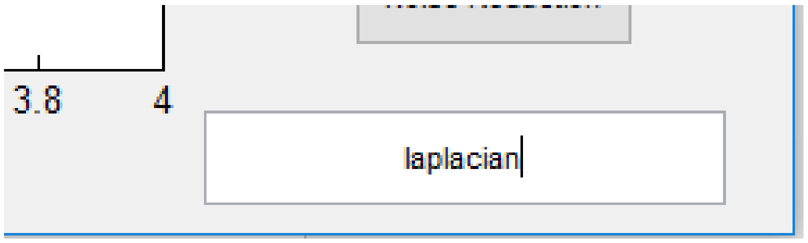
Sharpening Effect

When we click on **Sharpen**, we get the following pop up :



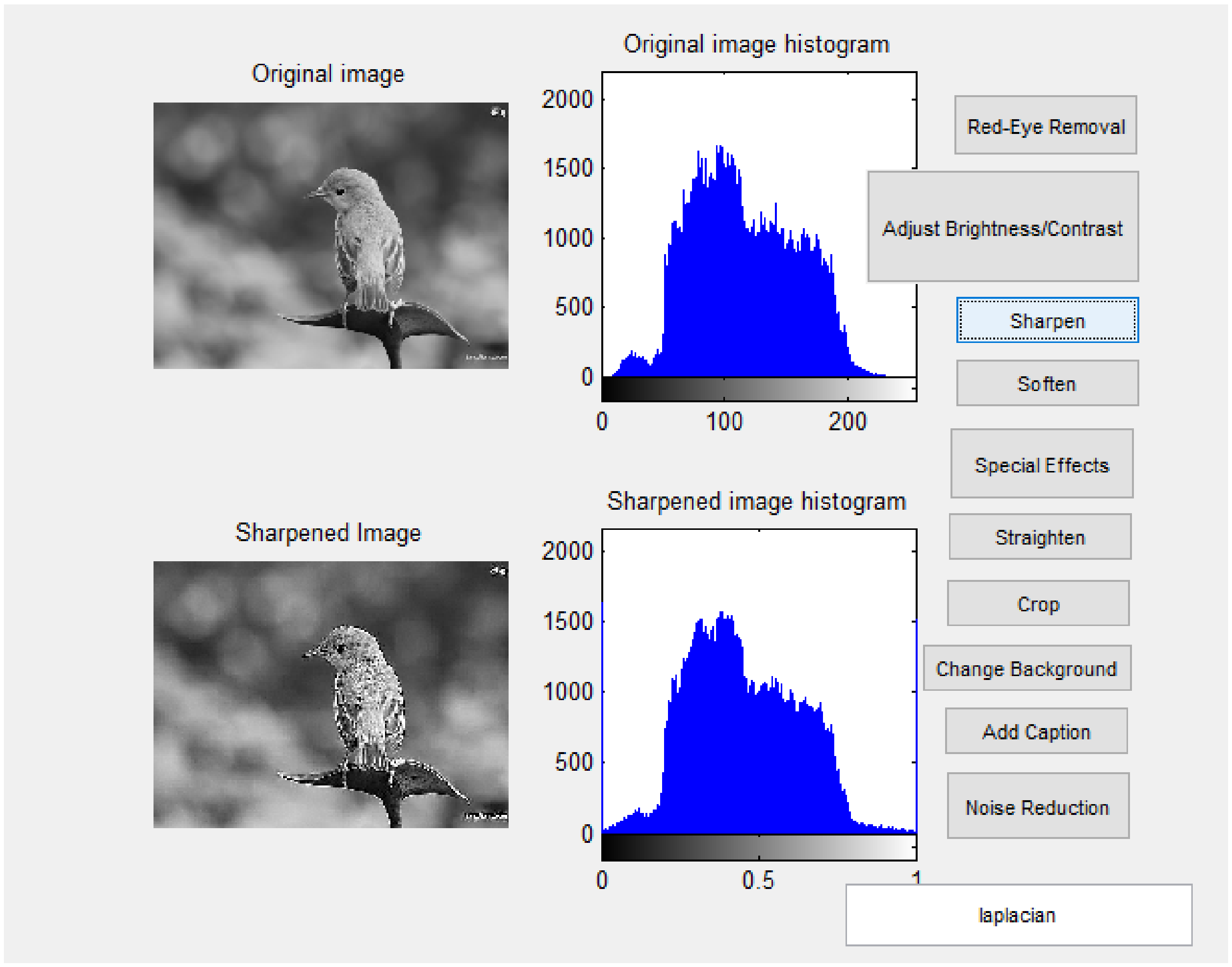
```
Specify type of sharpening you want to apply (Eg:average, disk, gaussian,
laplacian, log, motion, prewitt, sobel)
```

We specify the type of sharpening in the textbox present in lower-right corner.



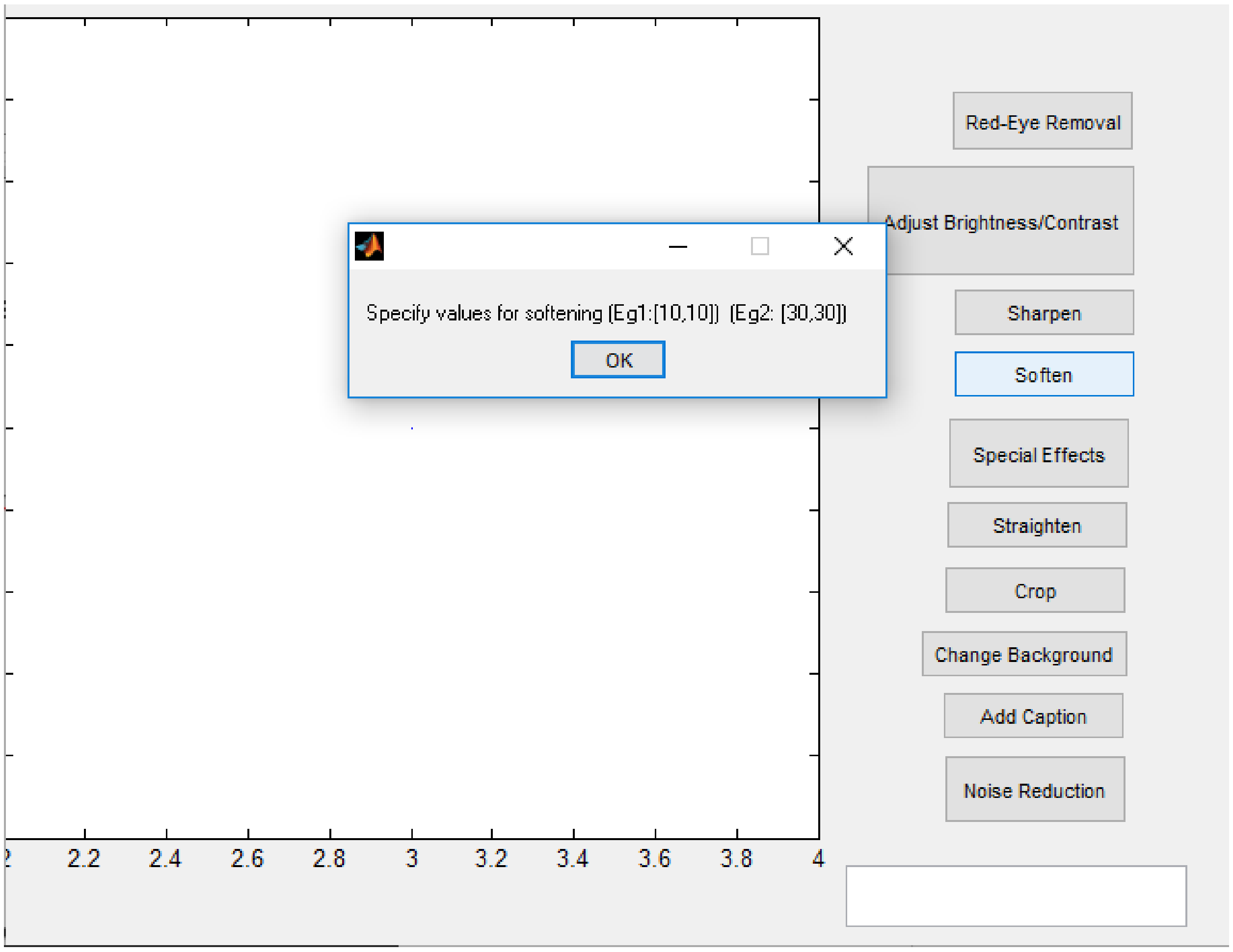
```
laplacian
```

The **type** parameter of **fpspecial(type)** function in matlab is used to provide various types of sharpening effects.



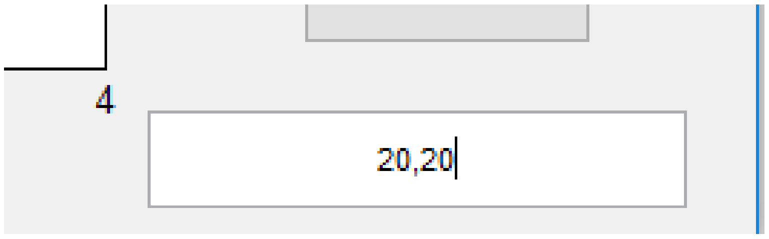
Softening Effect

When we click on **Soften**, we get the following pop up :



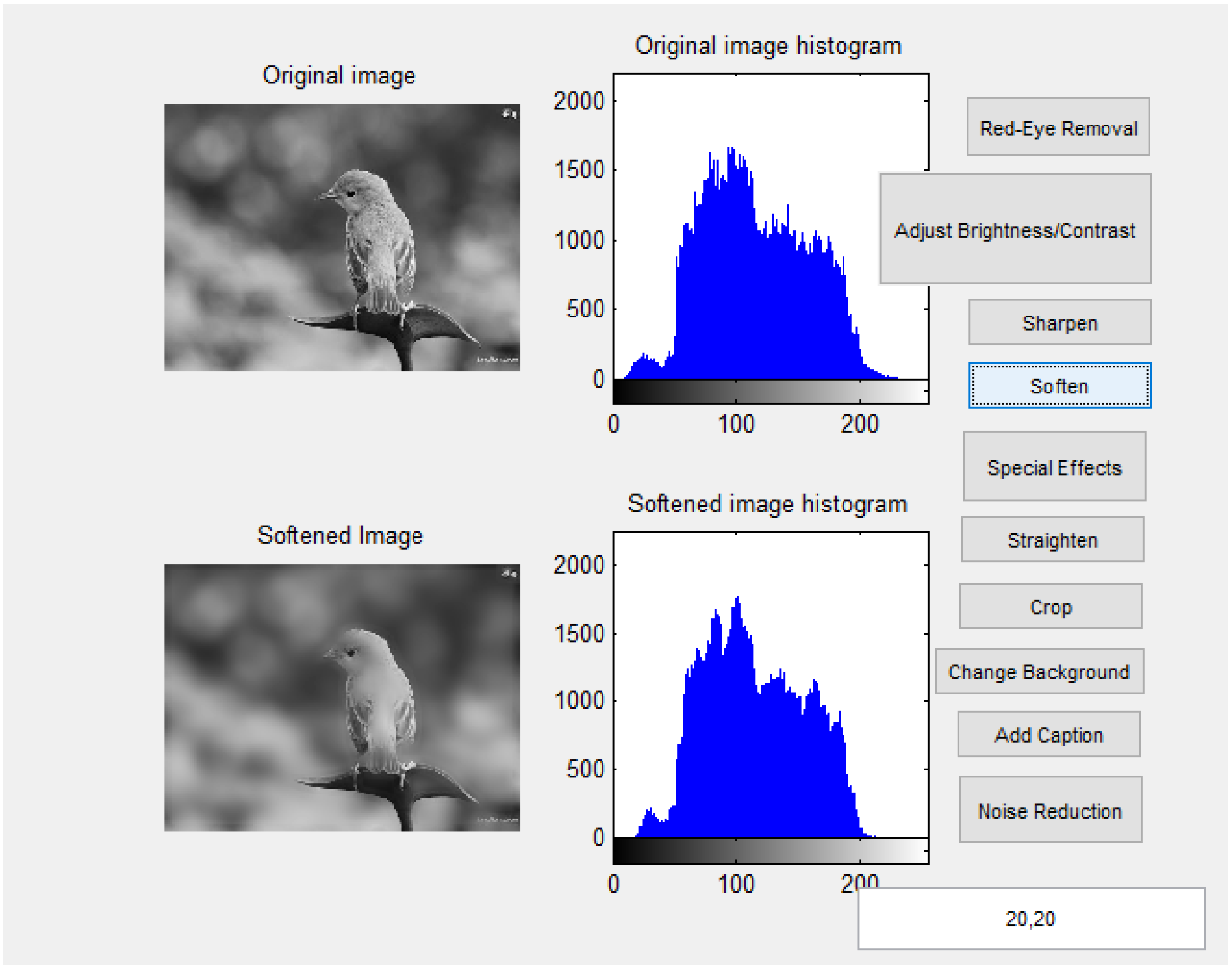
Specify values for softening [Eg1:[10,10]] [Eg2: [30,30]]

We specify the [m,n] values for softening in the textbox present in lower-right corner.



20,20

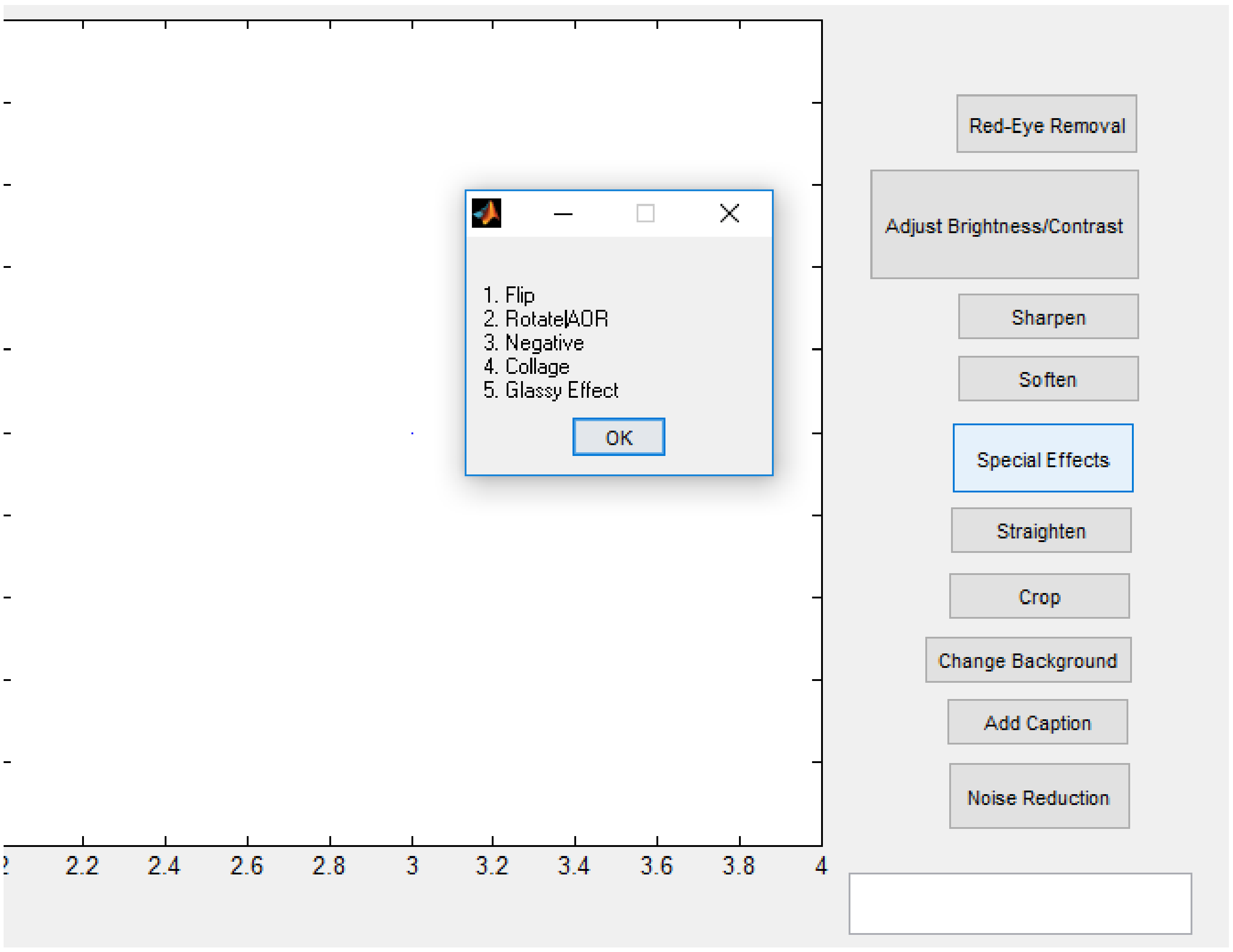
Integer array **[m,n]** used in **weiner2()** function filters the image using pixelwise adaptive Wiener filtering, using neighborhoods of size m-by-n to estimate the local image mean and standard deviation.



4. Applying Special Effects to an image

This feature enables the user to flip the image vertically, horizontally or both or rotate the image to the provided angle. By executing the provided code we get the following output :

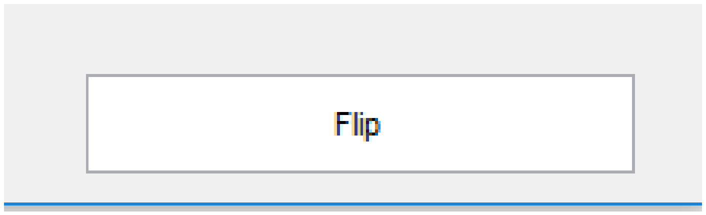
When we click on **Special Effects**, we get the following pop up :



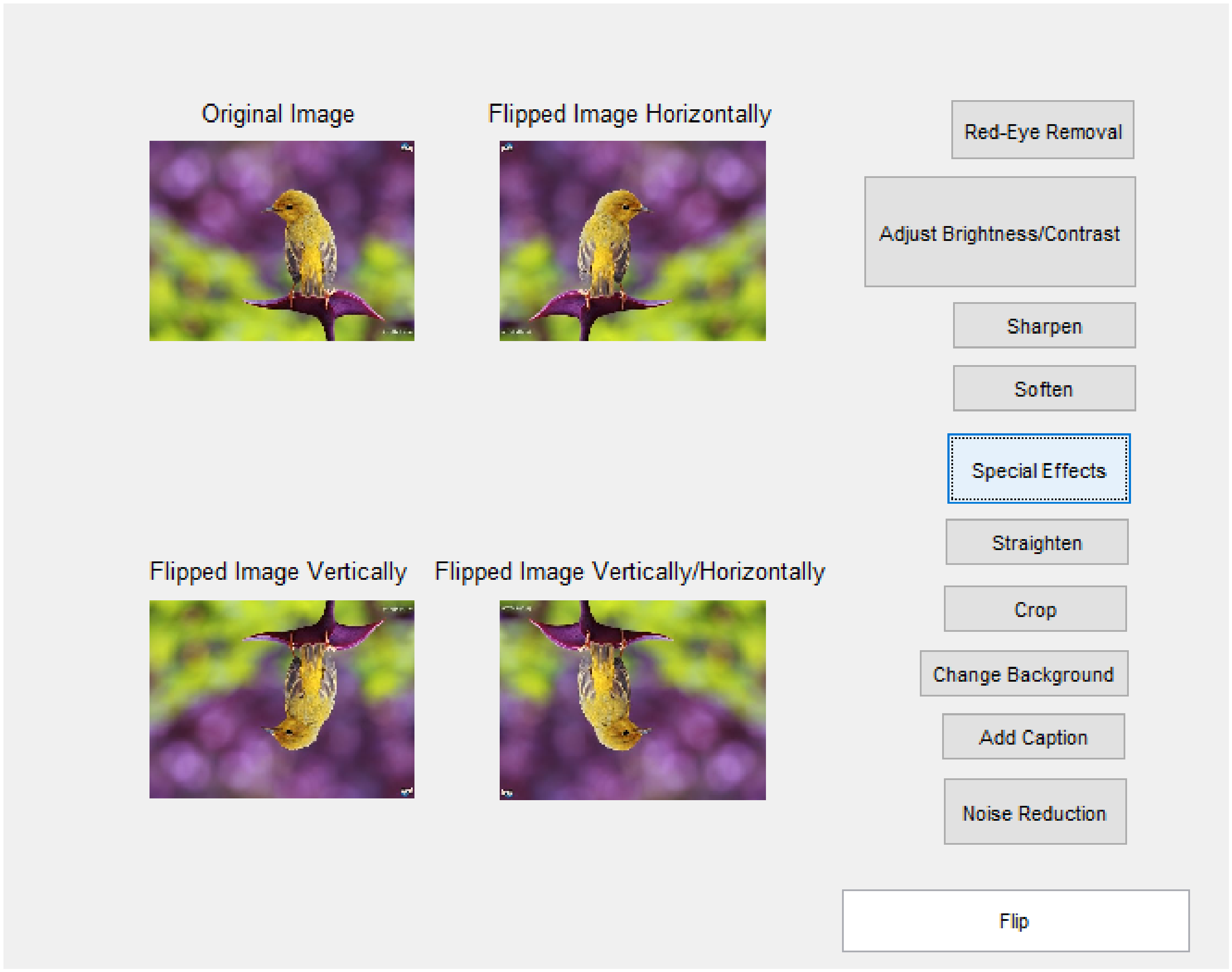
1. Flip
2. Rotate|AOR
3. Negative
4. Collage
5. Glassy Effect

Flip Images

We specify the **Flip** option in the textbox present in lower-right corner.

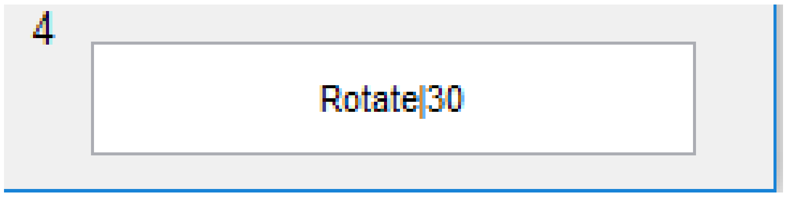


We get the following output :

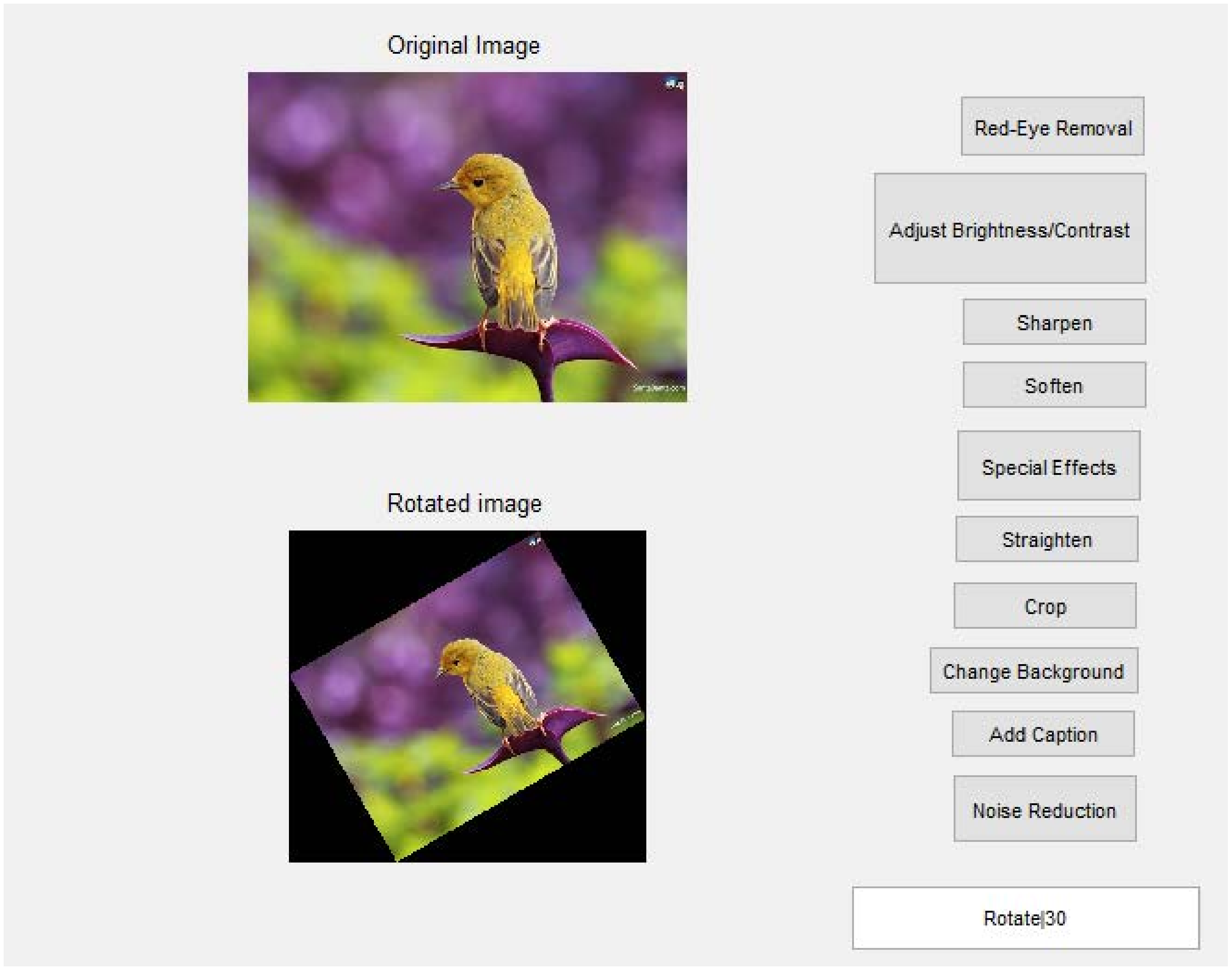


Rotate Images

We specify the **Rotate** option along with the angle of rotation in the textbox present in lower-right corner.

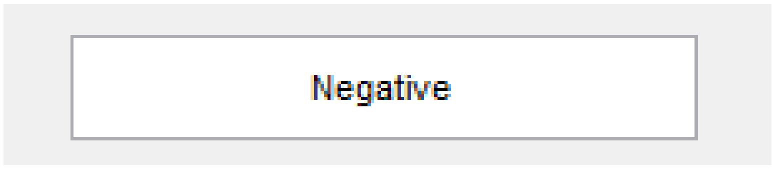


We get the following output :

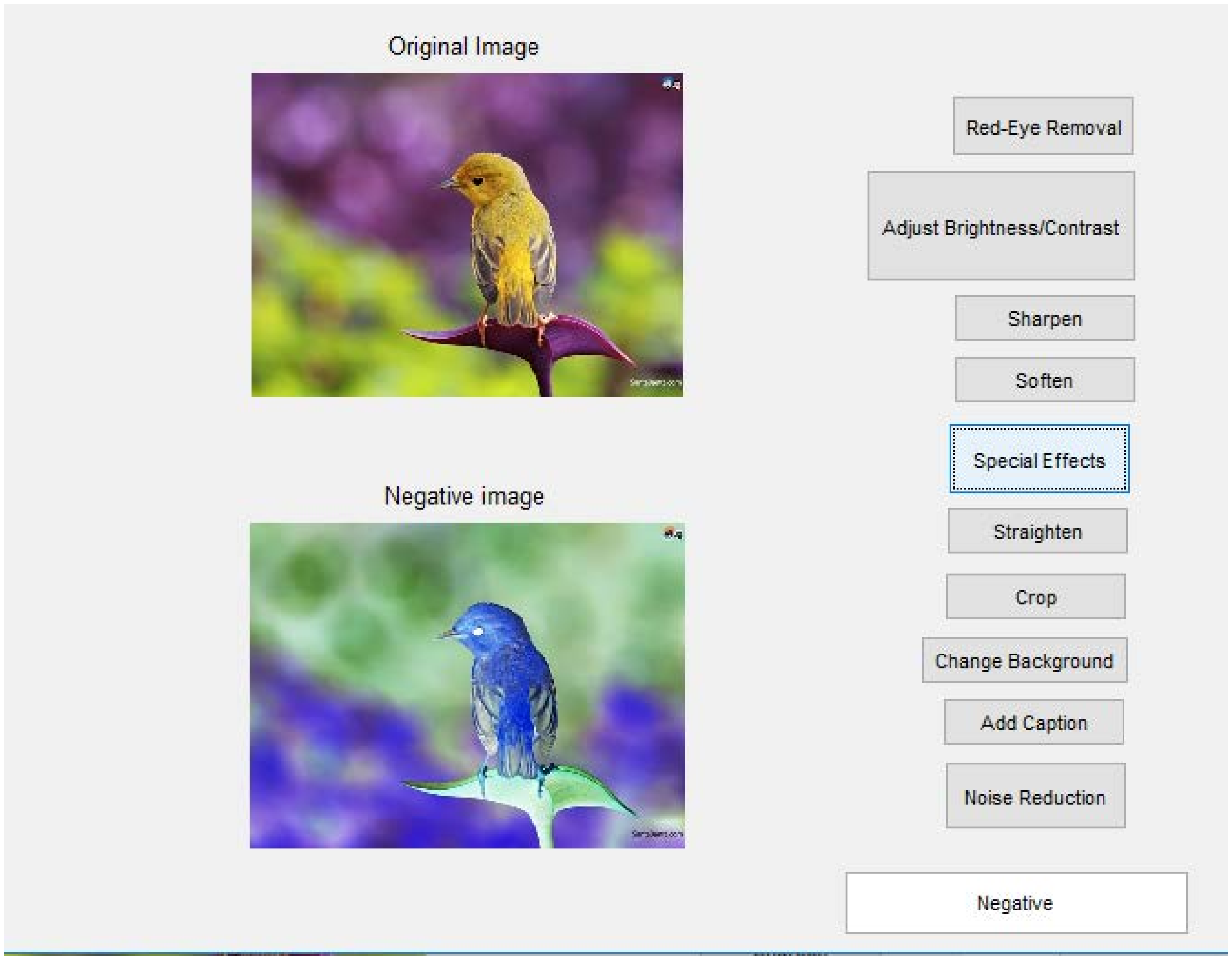


Create Negative of Images

We specify the **Negative** option in the textbox present in lower-right corner.

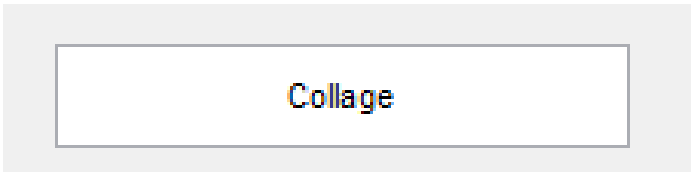


We get the following output :

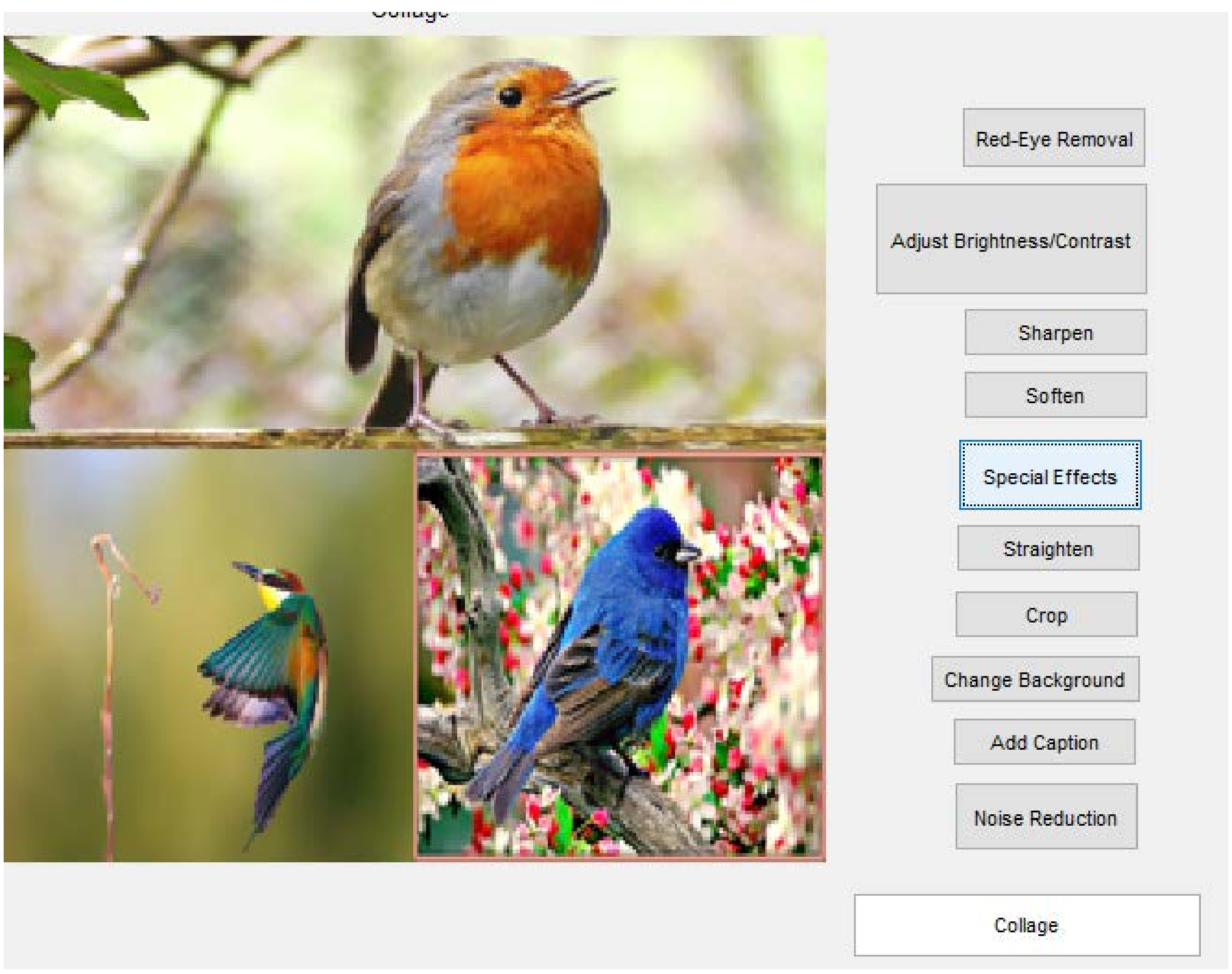


Create Collage of Images

We specify the **Collage** option in the textbox present in lower-right corner.

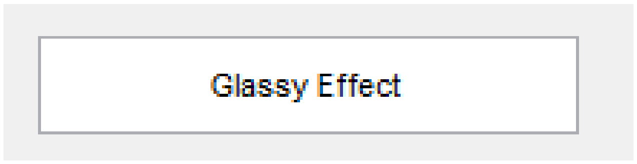


We get the following output :

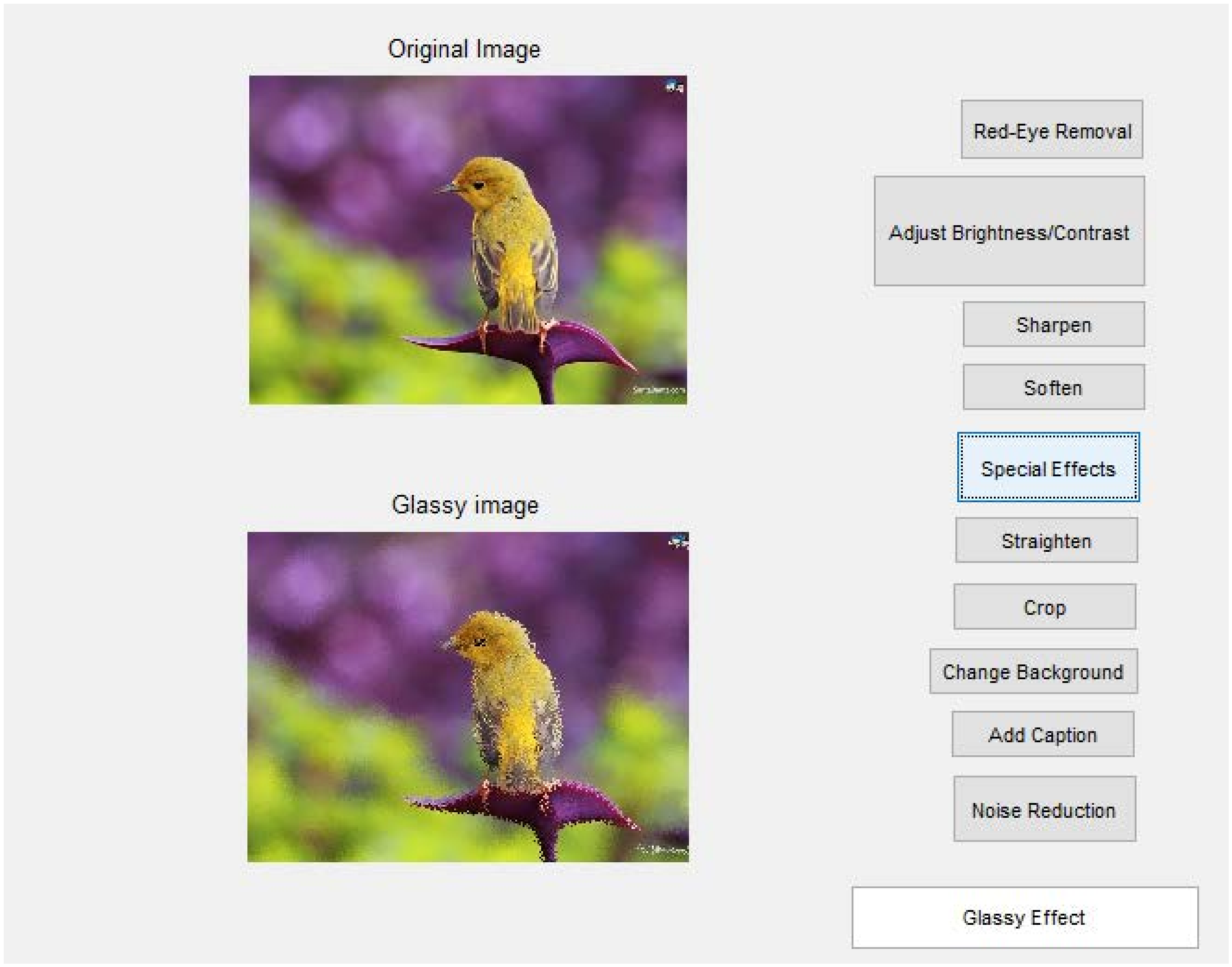


Create Glassy Effect on Images

We specify the **Glassy Effect** option in the textbox present in lower-right corner.



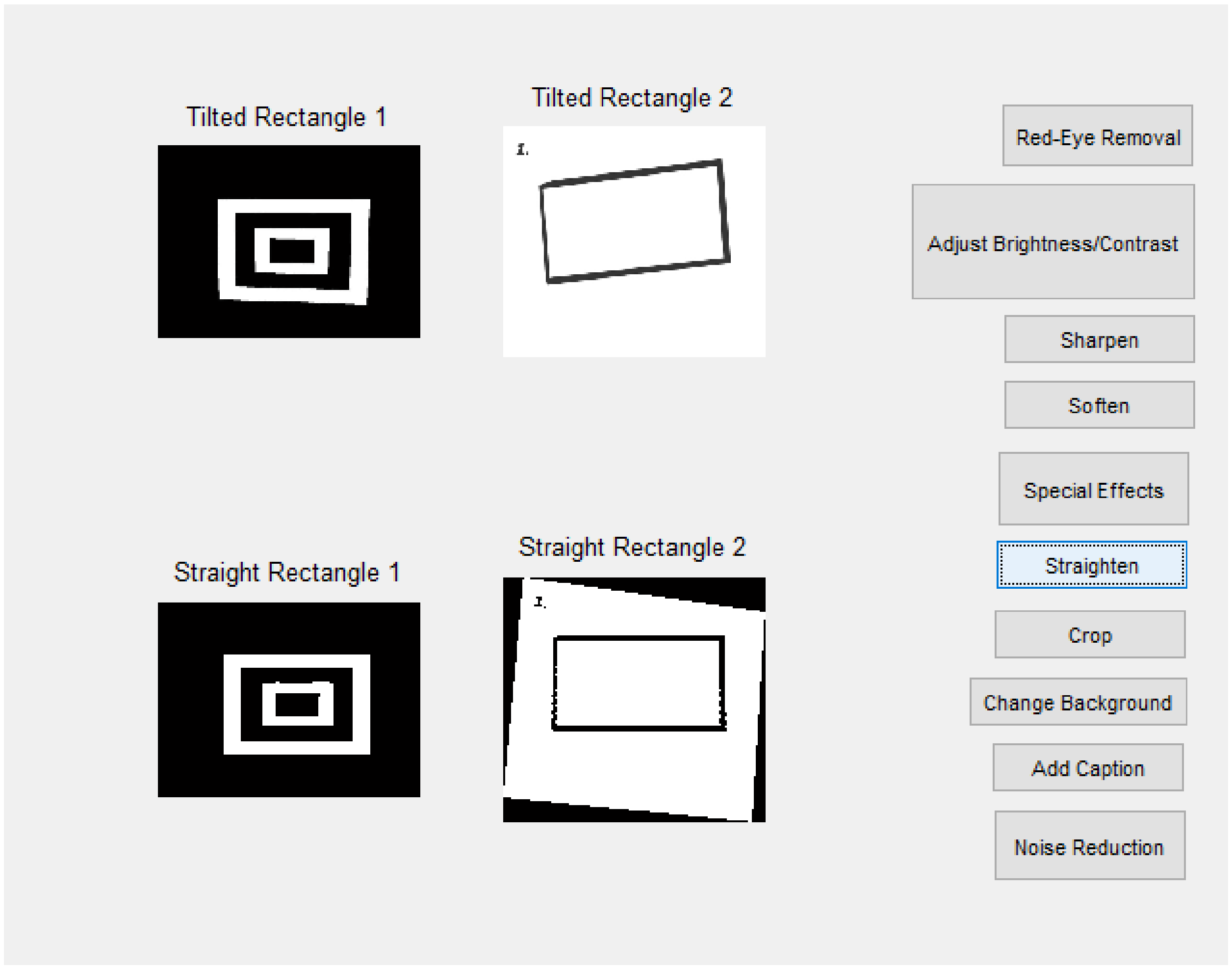
We get the following output :



5. Straighten images

This feature enables the user to straighten a tilted rectangle. By executing the provided code we get the following output :

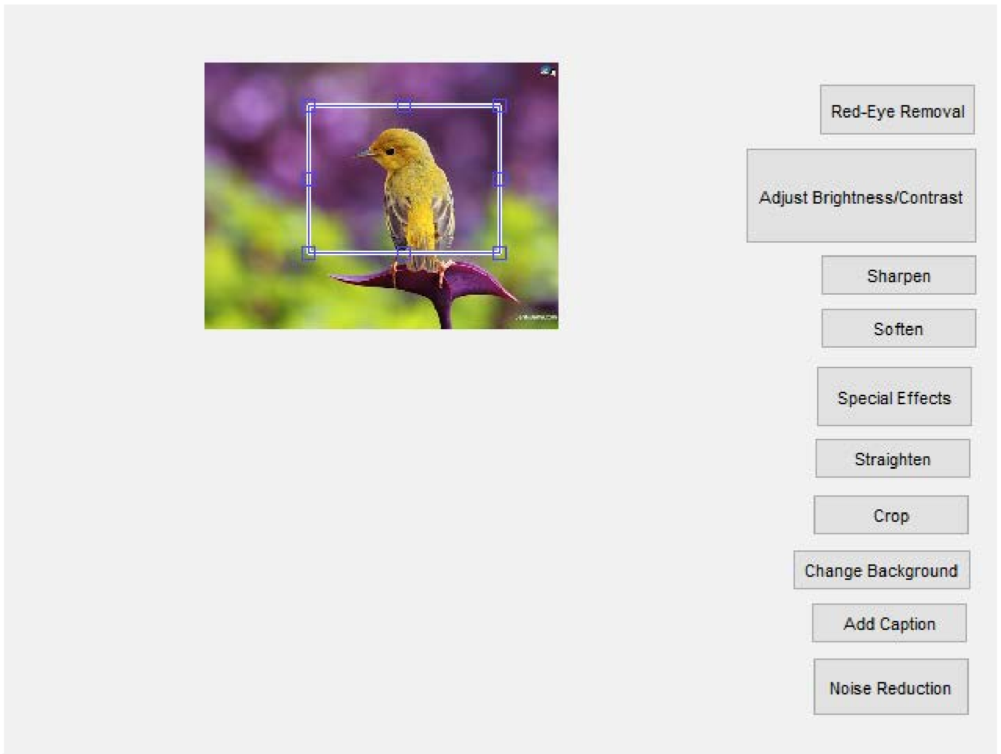
I fed two tilted rectangles as 2 different images for the process of straightening them.

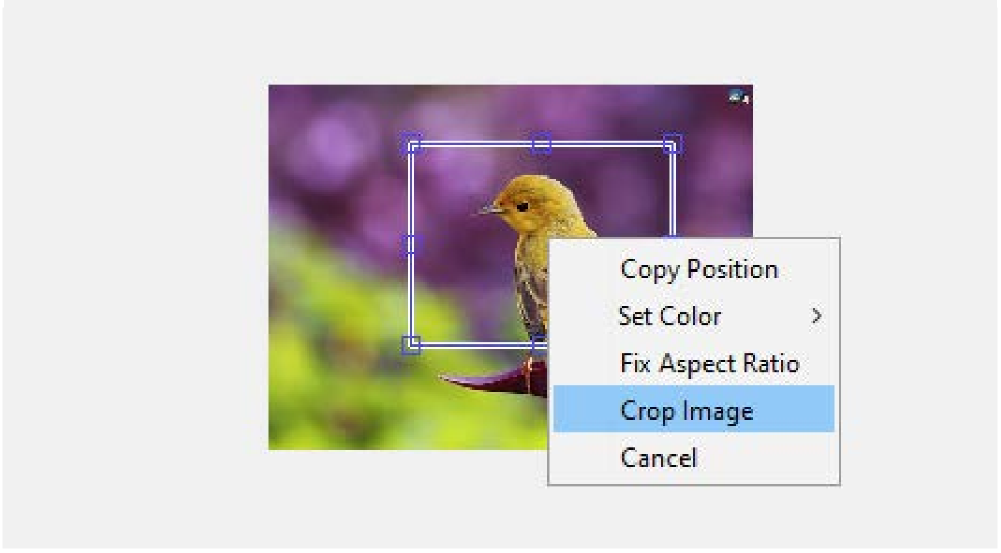


6. Crop images

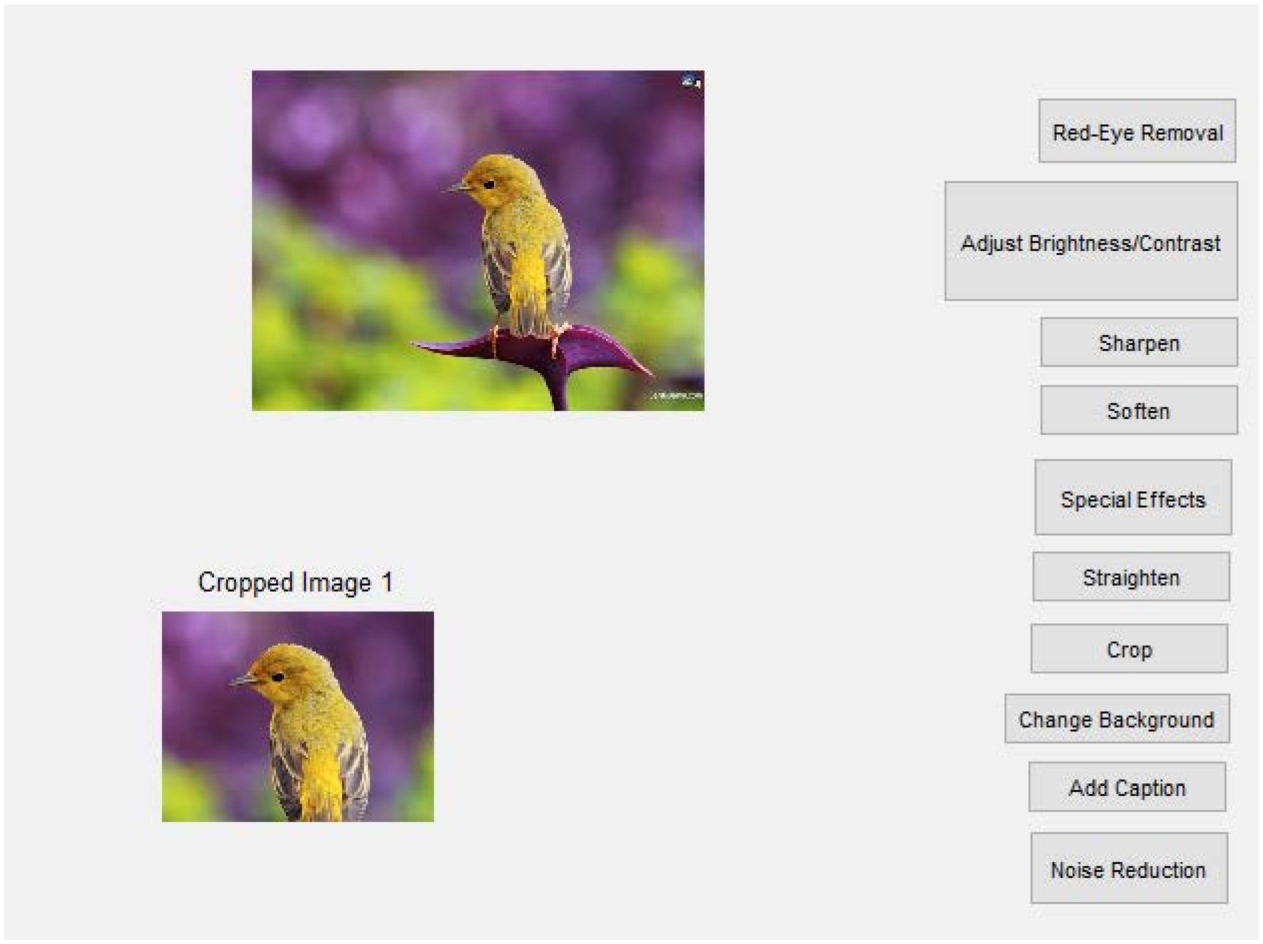
This feature enables the user to crop an image in two different ways : Either by using **imcrop()** tool that enables the user to crop through GUI   OR by providing the **imcrop()** parameters. By executing the provided code we get the following output :

Cropping Method 1 : Using GUI Cropping Tool



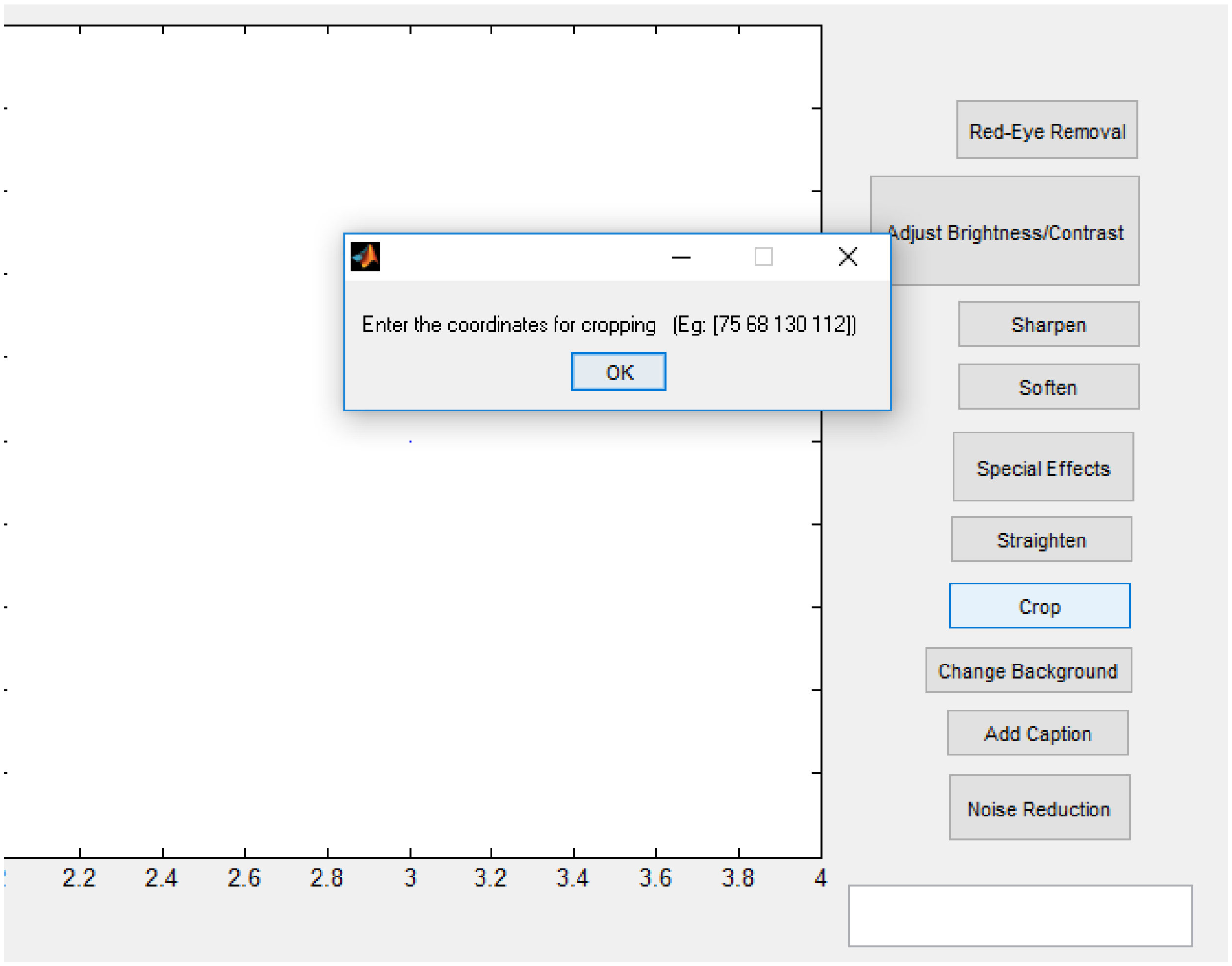


Final figure after using imcrop() tool :

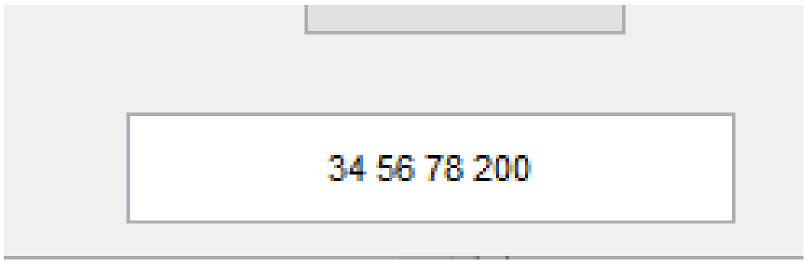


Cropping Method 2 : By Providing the coordinates for cropping

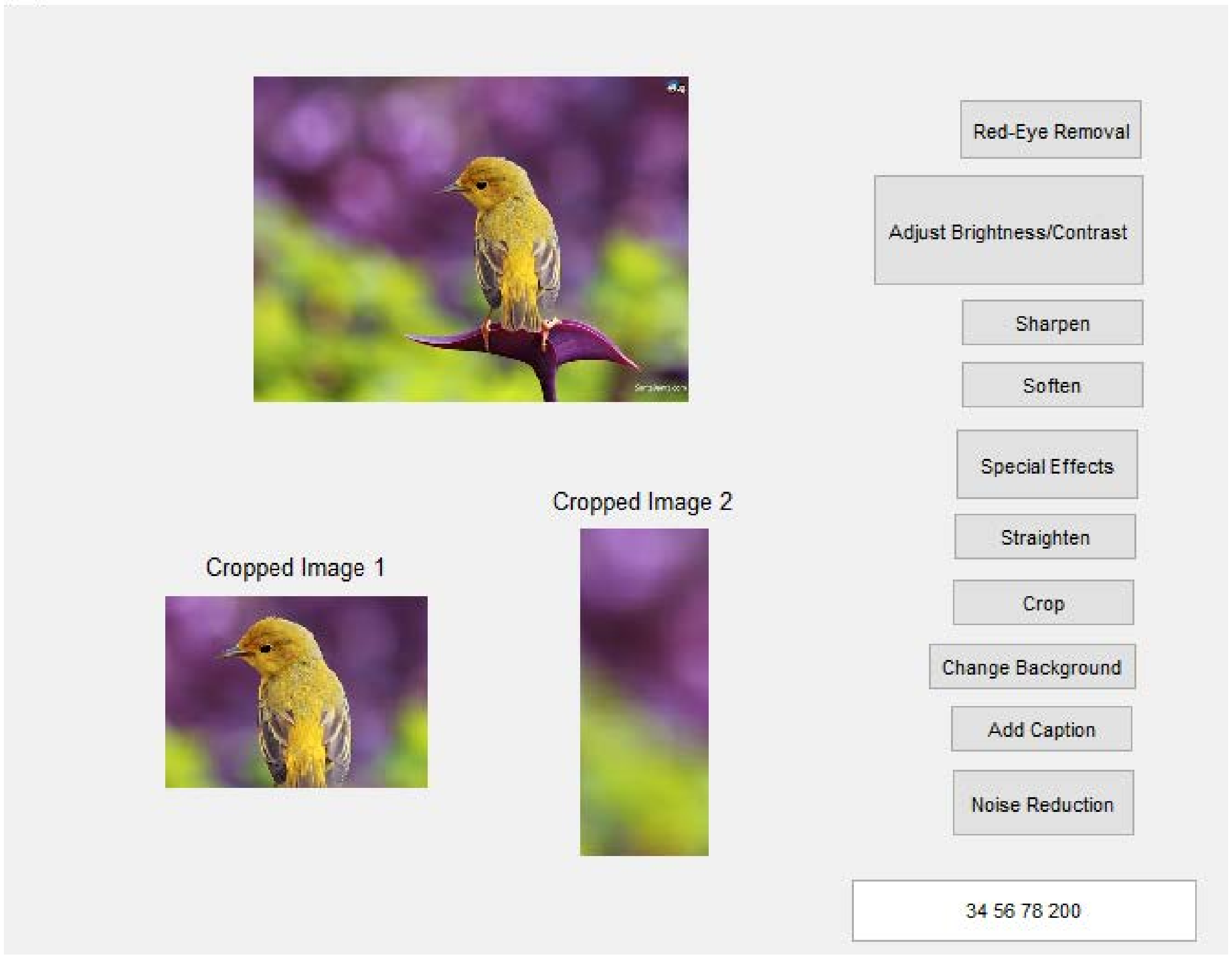
When we click on **Crop**, we get the following pop up :


Enter the coordinates for cropping   (Eg: [75 68 130 112])

We specify the cropping coordinates in the textbox present in lower-right corner.


34 56 78 200

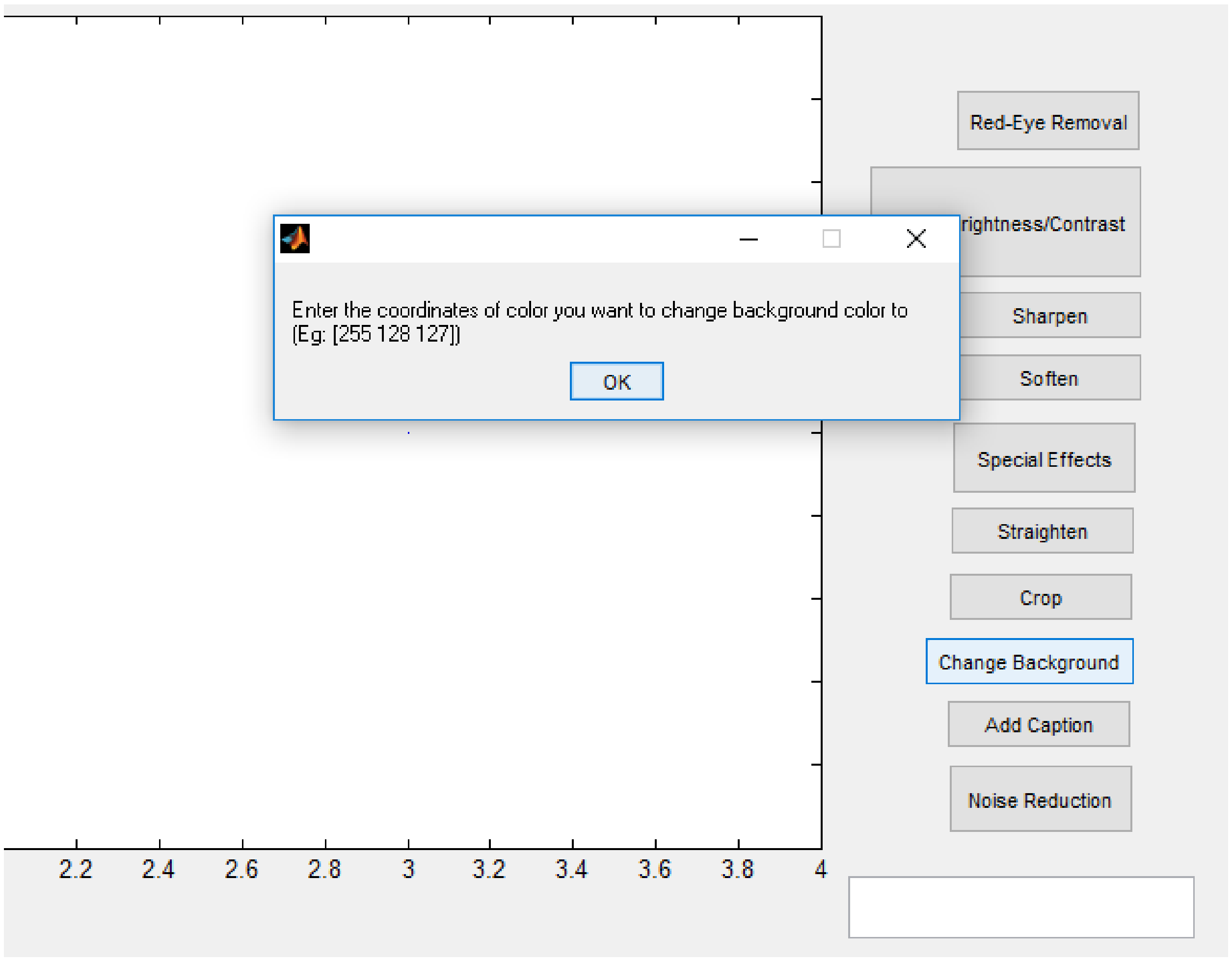
Final figure after using imcrop() function and providing the coordinates for cropping :



7. Change the background of Images

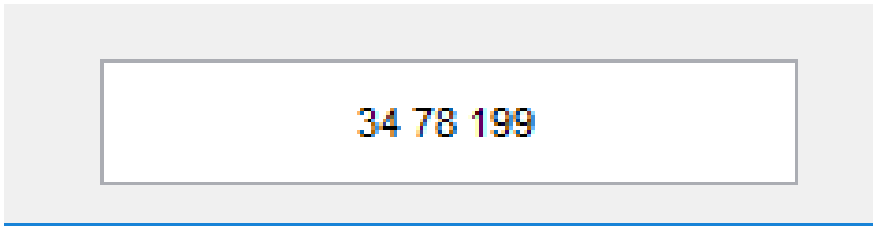
This feature enables the user to change the background color from Black or White to any color specified by the user. By executing the provided code we get the following output :

When we click on **Change Background**, we get the following pop up :

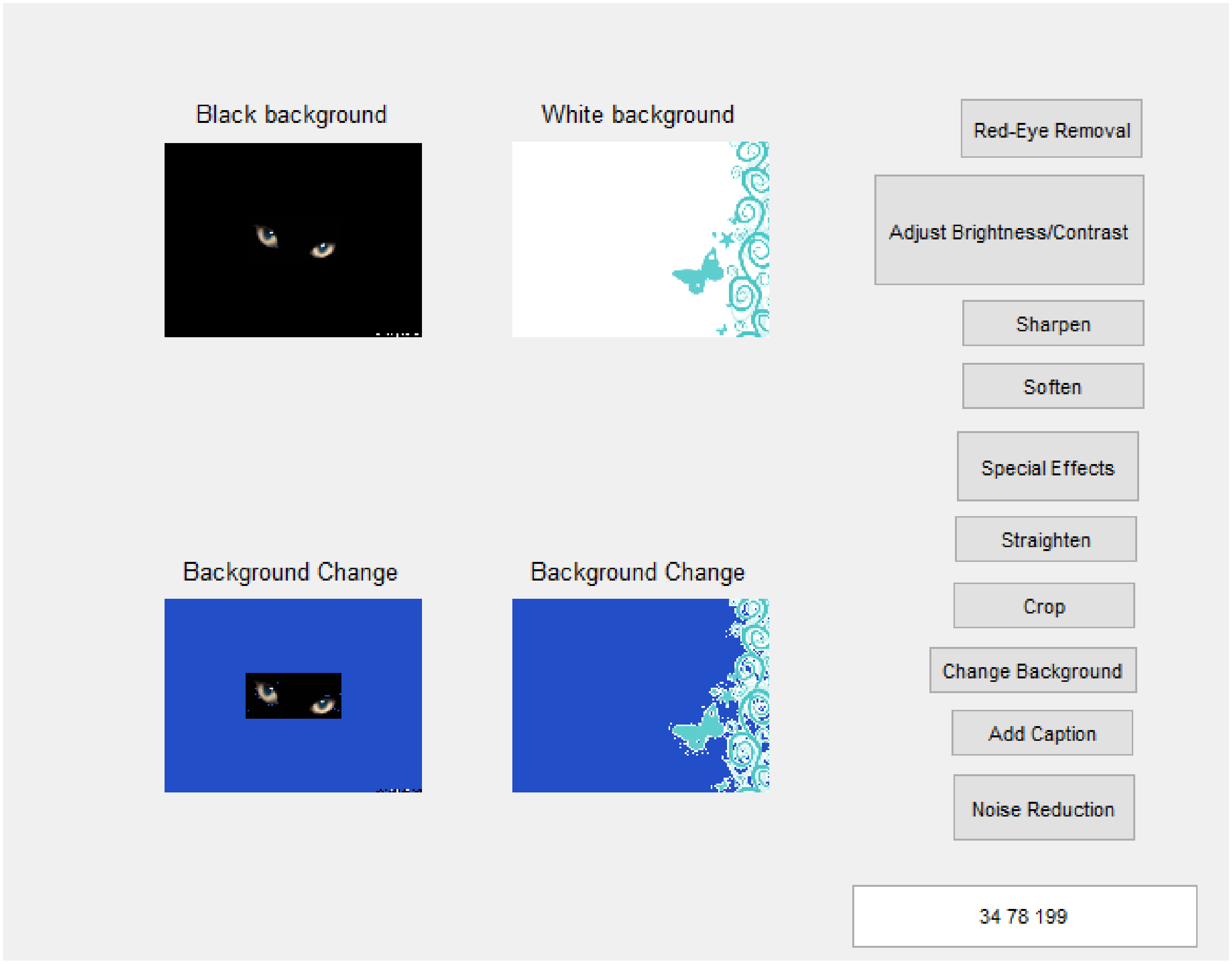


Enter the coordinates of color you want to change background color to
[Eg: [255 128 127]]

We specify the color coordinates in the textbox present in lower-right corner.



The above color coordinates correspond to a variant of Blue color :

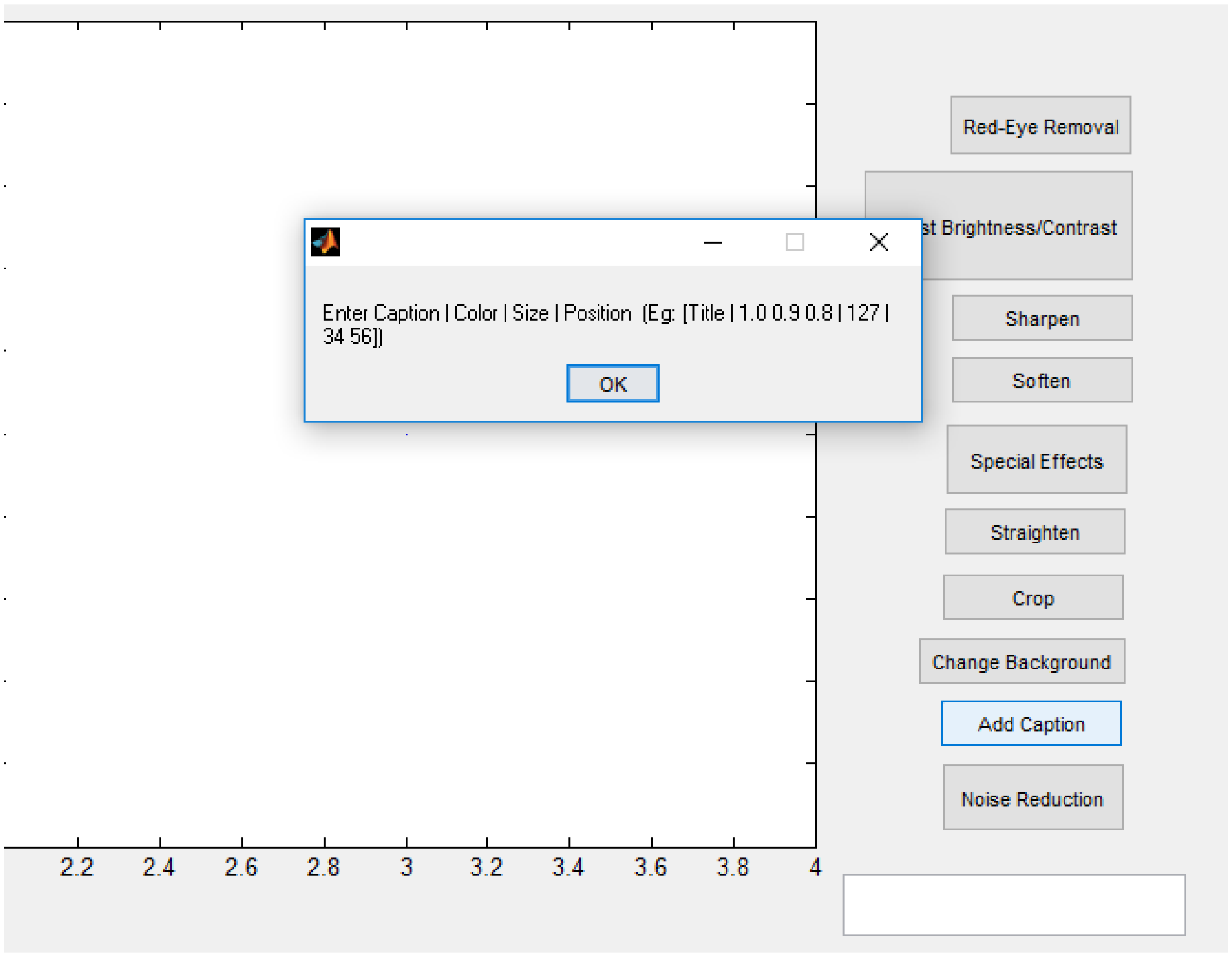


34 78 199

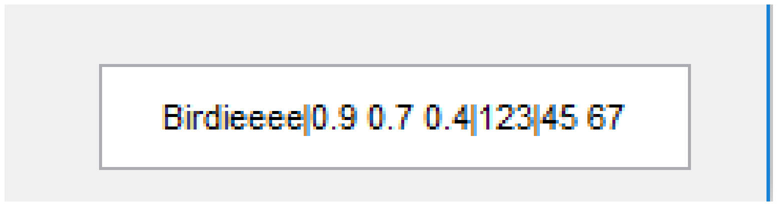
8. Add a Caption to images

This feature enables the user to add a caption to the image in various font colors and sizes and place it anywhere on the image. By executing the provided code we get the following output :

When we click on **Add Caption**, we get the following pop up :

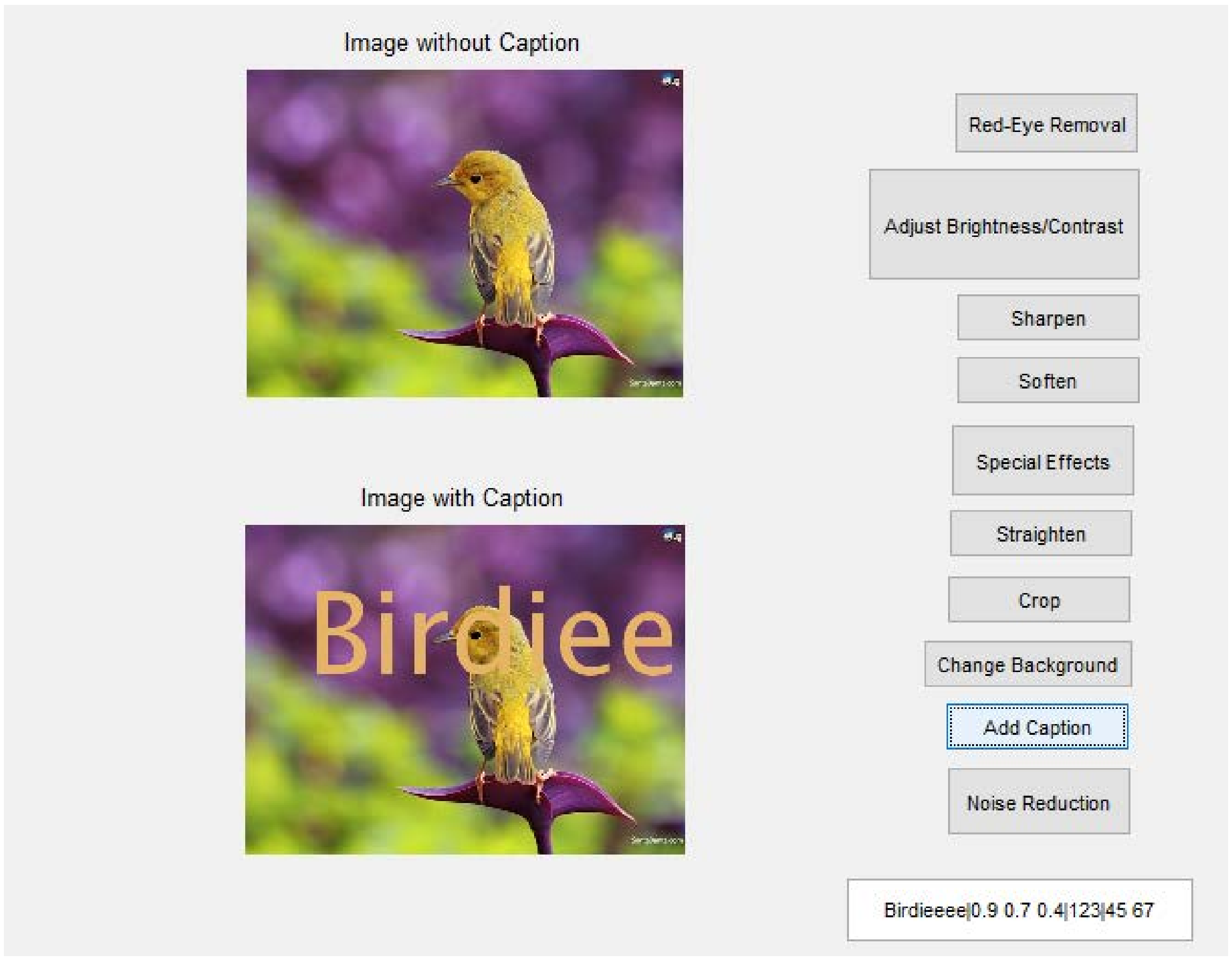

Enter Caption | Color | Size | Position (Eg: [Title | 1.0 0.9 0.8 | 127 | 34 56])

We specify the various parameters for the caption separated by '**|**' in the textbox present in lower-right corner.


Birdieeee|0.9 0.7 0.4|123|45 67

First input corresponds to the caption we want to declare in the image.
Second input is the color of the caption.
Third input is the size of the font of the caption.
Fourth input is the position [Row,Col] where we want to place the caption in the image.

Following figure is generated upon providing all the inputs.

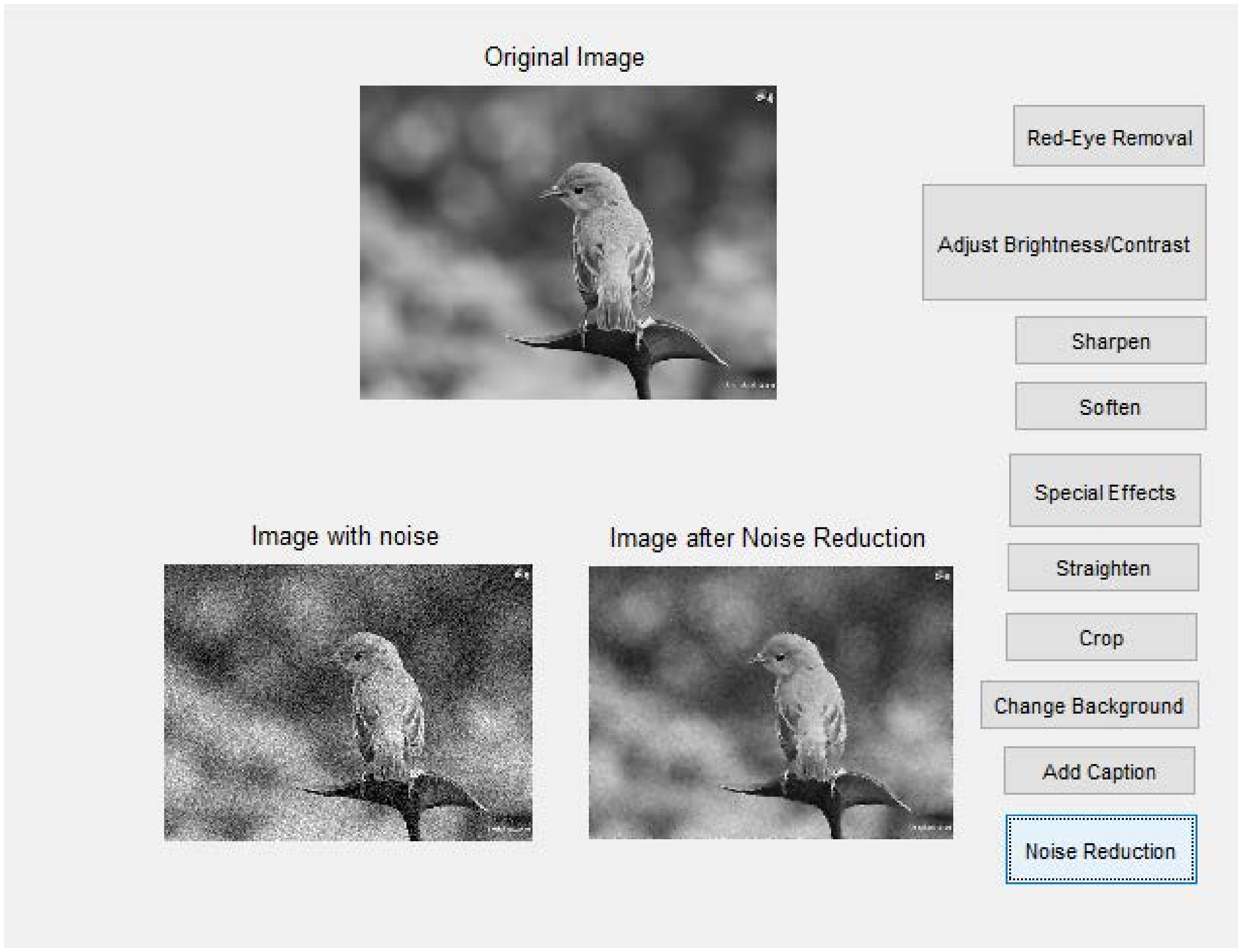


Birdieeee|0.9 0.7 0.4|123|45 67

9. Noise Reduction

This feature enables the user to remove the unnecessary graining present in the images. By executing the provided code we get the following output :

Noise is first introduced to Original image using **imnoise()** function of MATLAB and then noise is reduced using **weiner2()** function.



CONCLUSION

Following GUI for Photo Editing Software has been created and various Push Buttons has been implemented for features of the software.

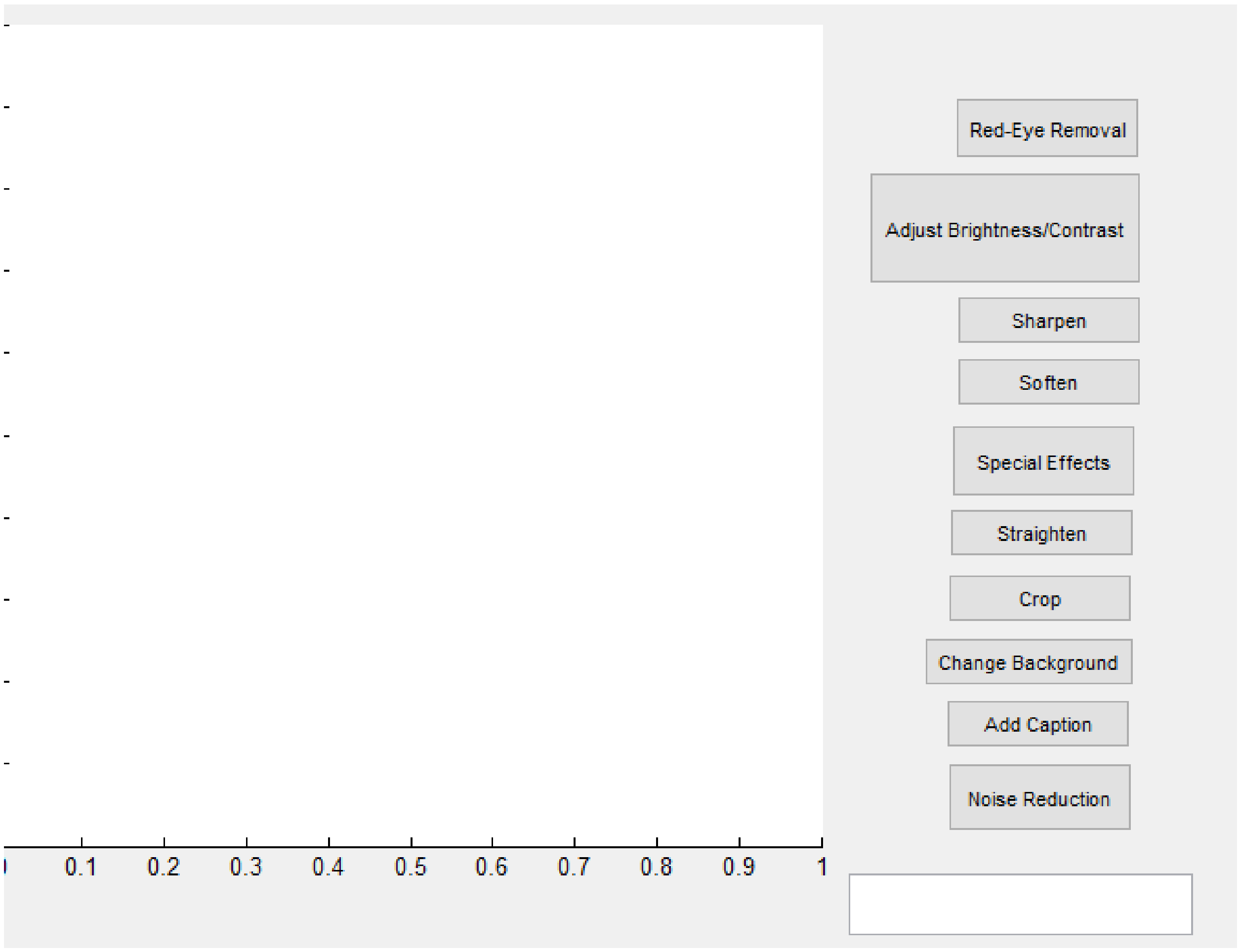


Photo Editing Software has been created and implemented using MATLAB R2011a and has been used to implement the following features :

1. Red Eye Removal

2. Adjust brightness and contrast

3. Sharpening and softening effects on image

4. Special effects

5. Straighten an image

6. Crop an image

7. Change background of an image

8. Add a caption to an image

9. Noise reduction

BIBLIOGRAPHY

1. http://in.mathworks.com/help for :
   - http://in.mathworks.com/help/matlab/ref/input.html
   - http://in.mathworks.com/help/matlab/creating_guis/about-the-simple-guide-gui-example.html
   - http://in.mathworks.com/discovery/matlab-gui.html
   - http://in.mathworks.com/help/matlab/ref/image.html
   - http://in.mathworks.com/help/matlab/ref/ginput.html
   - http://in.mathworks.com/help/images/ref/imadjust.html
   - http://in.mathworks.com/help/images/ref/wiener2.html
   - http://in.mathworks.com/help/images/ref/im2bw.html
   - http://in.mathworks.com/help/images/ref/imclearborder.html
   - http://in.mathworks.com/help/matlab/ref/find.html
   - http://in.mathworks.com/help/images/ref/cp2tform.html
   - http://in.mathworks.com/help/images/ref/imcrop.html
   - http://in.mathworks.com/help/matlab/ref/im2double.html
   - http://in.mathworks.com/help/vision/ref/vision.textinserter-class.html
   - http://in.mathworks.com/help/images/ref/imnoise.html
   - http://in.mathworks.com/help/images/noise-removal.html?requestedDomain=www.mathworks.com
   - http://in.mathworks.com/matlabcentral/answers/66123-matlab-find-the-contour-and-straighten-a-nearly-rectangular-image
   - http://in.mathworks.com/help/images/ref/imrotate.html
   - http://in.mathworks.com/help/matlab/ref/flipdim.html

2. Other sources :
   - http://stackoverflow.com/questions/15251045/matlab-find-the-contour-and-straighten-a-nearly-rectangular-image
   - http://mikko.wtf/perspective-rectification/