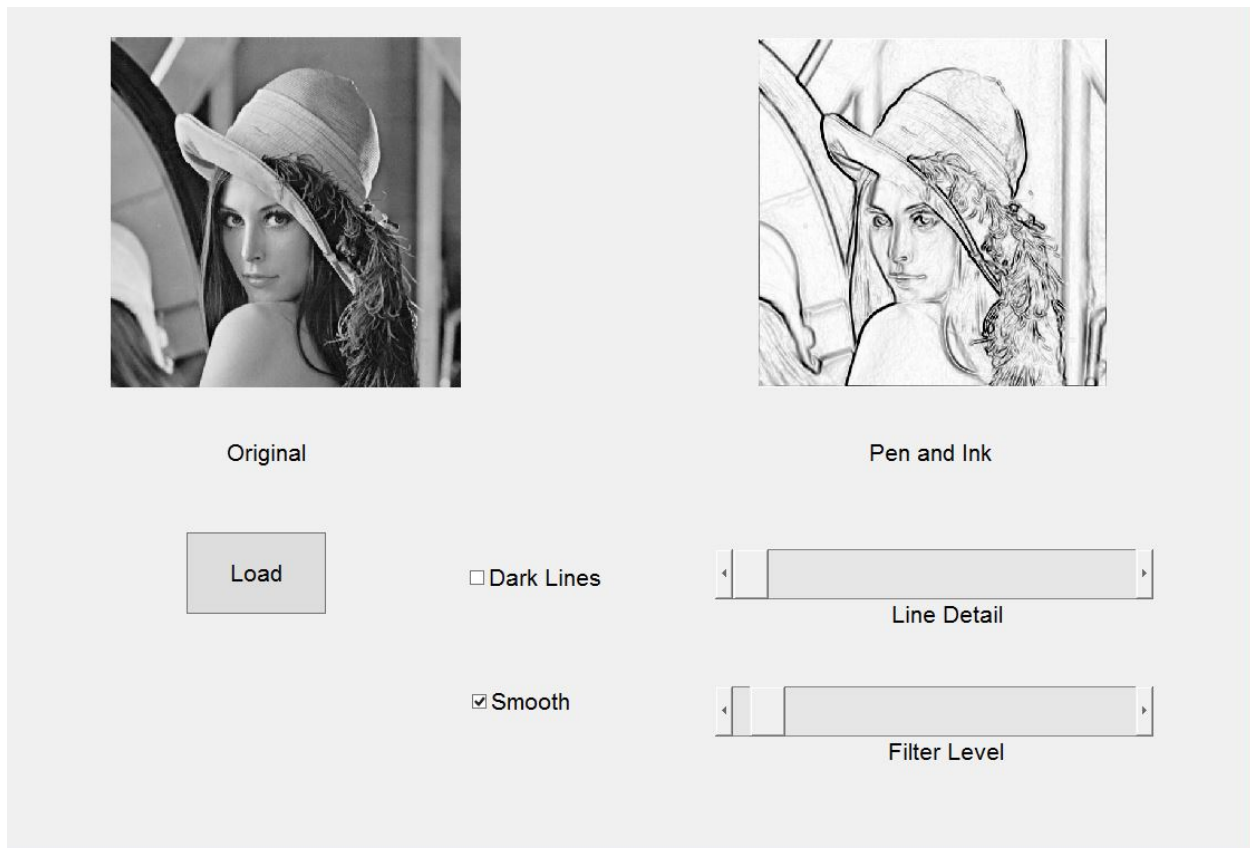


Overview

In this program, I created my version of the “pen and ink” effect. The GUI allows you to load an image of your choosing and render a “sketched” version of that image. The GUI comes with parameters that let you adjust how detailed you want the sketch to be as well as how much noise you want to be included. The image below shows a summary of the GUI.

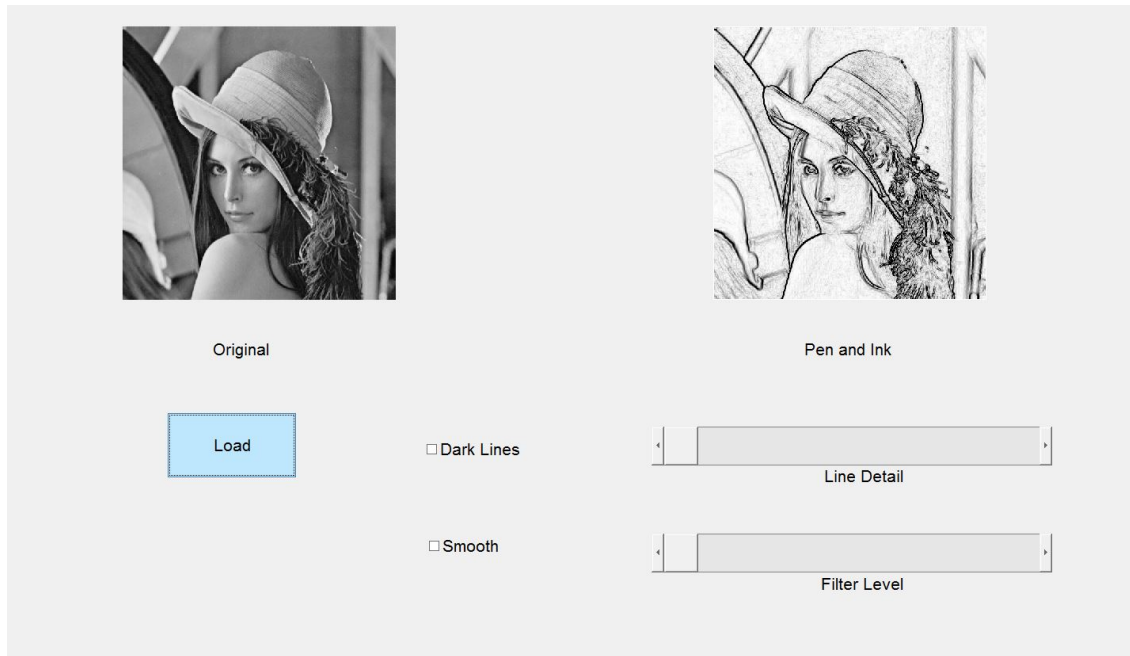


Instructions

1. Open finalproject.m with Matlab and run the code. The GUI should open with two blank windows.
2. Click the “load” button. Select an image of your choosing and click “open”. The original image should appear in the left window, and a sketched version of the image should appear in the right window.
3. If you would like your image to be comprised of only black lines without shading, select the “dark lines” checkbox. You can adjust how much detail appears by adjusting the “line detail” slider. This slider will not affect the image if the “dark lines” checkbox is not selected.
4. If you would like to eliminate some of the noise in the image, select the “smooth” checkbox. Without “dark lines” selected, increasing the “filter level” slider will make the brush strokes lighter. With “dark lines” selected, increasing the slider will decrease the detail of the image and emphasize the more important components. This slider will not affect the image if the “smooth” checkbox is not selected.

Implementation Details

In order to implement my pen and ink effect, I used three main techniques. The first technique involved using horizontal and vertical Sobel operators on the image in order to determine the gradient magnitude between pixels. The inverted version of these values is displayed as an image, showing the parts of the image with higher gradient magnitude as darker and lower gradient magnitude as lighter. This effect can be seen in the image below, and the following code was written to create this effect.



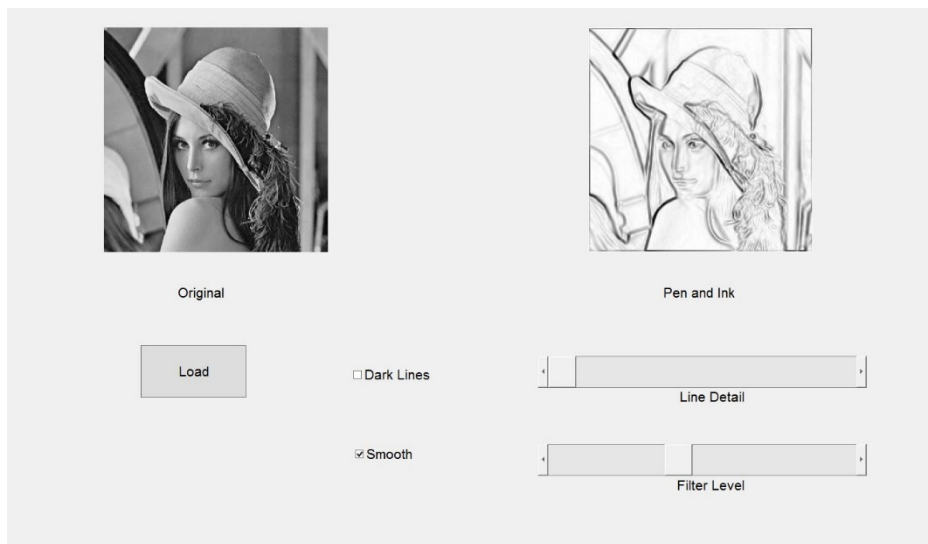
```
function displayResult
%access the image data matrix and desired outpuse axes
global X;
global hAxes2;

%Determine horizontal and vertical gradiant madnitude of image
[height, width, dim] = size(Y);
hSobel = [-1 0 1; -2 0 2; -1 0 1];
vSobel = [1 2 1; 0 0 0; -1 -2 -1];
hMag = zeros(height, width);
vMag = zeros(height, width);
for i = 1:1:width-2
    for j = 1:1:height-2
        window = Y(i:i+2,j:j+2,1);
        hConv = conv2(window, hSobel);
        hMag(i+1,j+1)=hConv(3,3);
        vConv = conv2(window, vSobel);
        vMag(i+1,j+1)=vConv(3,3);
    end
end

%Find total gradiant magnitude and invert pixel values
totalMag = sqrt((hMag.*hMag) + (vMag.*vMag));
totalMag = 1 - (totalMag ./ 255);
```



```
% show the result
set(gcf, 'CurrentAxes', hAxes2);
imshow(totalMag);
```



The second technique involves applying a gaussian filter of varying size to the image before determining the gradient magnitude. Because the filter effectively smooths the image, it decreases the image gradient magnitudes or, as seen later, decreases the number of lines representing the modified image. The figure below displays the effect of applying the gaussian filter, and the following code was written to apply this effect.



```
%get the filter status
smoothStatus = findobj(gcf, 'Tag', 'smooth_checkbox');
smooth = get(smoothStatus, 'Value');
% get the filter size
gaussLevel = findobj(gcf, 'Tag', 'filter_slider');
gauss = get(gaussLevel, 'Value');
%apply gaussian filter if "smooth" box is checked
Y = X;
if smooth == true
    H = fspecial('gaussian', round(gauss), 0.5 * round(gauss));
    Y = imfilter(X, H);
end
```

The final technique involves limiting the pixel values of the image to be either 0 or 1, eliminating shading and creating a sharper version of the pen and ink effect. This is done by setting a specific value based on slider placement, and values above that will be white (1) and values below will be black (0). This is done after normalizing the image, which is why it is on a 0-1 scale instead of a 0-255 scale. The figures below show the effect of this and how applying the gaussian filter affects the result. The following code was written to create this effect.

	
Original	Pen and Ink
<div>Load</div>	<div><input checked="" type="checkbox"/> Dark Lines</div>
	<div><div></div><div></div></div> <div>Line Detail</div>
	<div><input type="checkbox"/> Smooth</div>
	<div><div></div><div></div></div> <div>Filter Level</div>

	
Original	Pen and Ink
<div>Load</div>	<div><input checked="" type="checkbox"/> Dark Lines</div>
	<div><div></div><div></div></div> <div>Line Detail</div>
	<div><input checked="" type="checkbox"/> Smooth</div>
	<div><div></div><div></div></div> <div>Filter Level</div>

```
%get the dark line status
darkStatus = findobj(gcf, 'Tag', 'dark_lines');
dark = get(darkStatus, 'Value');
% get the detail level
detailLevel = findobj(gcf, 'Tag', 'detail_slider');
detail = get(detailLevel, 'Value');
%If 'dark lines' is checked, set pixel values to white or black
%depending on threshold value 'detail'.
```

```
if dark == true
    for i = 1:1:width
        for j = 1:1:height
            if totalMag(i,j) <= detail
                totalMag(i,j) = 0;
            else
                totalMag(i,j) = 1;
            end
        end
    end
end
end
```

The figures below are the same four effects side by side for easier comparison.



Pen and Ink effect



Pen and Ink effect with gaussian filter



Pen and Ink effect with dark lines



Pen and ink effect with dark lines and gaussian