Tool to visualise the sketching process

Problem Statement

Aspiring artists usually struggle in visualising the process of sketching and consequently adding depth to the artwork using shading and hatching.

Proposed solution

We hope to help them visualise these steps by providing a step by step breakdown of the sketching process.
In our application, the user is first able to either upload a reference image of their choice or to capture it with their webcam. We then use fuzzy logic based edge detection techniques to provide them with a preliminary image with basic edges so as to help first get the basic outlines right. We apply image enhancing techniques and also increase sensitivity of the edge detection algorithm to provide the user with an image with detailed edges.

After this step, we threshold the intensity levels in the input image to provide the user with a series of images with different shades visible to them, hence providing them with a guide of how to go about adding definition and depth to the sketch through means of hatching and shading,

Concepts used

Image Enhancement

Image enhancement is the procedure of improving the quality and information content of original data before processing. The two techniques used in this project are the following.

Uniform Brightness Adaptation

This process brings in a uniform brightness all over the image. This could be a result of the non uniform background of the image which could hinder the performance of the edge detection algorithms.

Enhancing of Low Light image

Images captured in outdoor scenes can be highly degraded due to poor lighting conditions. These images can have low dynamic ranges with high noise levels that affect the overall performance of edge detection algorithms. To make it robust in low-light conditions, use low-light image enhancement to improve the visibility of an image.

Edge Detection

The primary step in visualising the image is edge detection. Edge detection highlights high frequency components in the image.
We have implemented Edge Detection using Fuzzy logic .
Fuzzy logic is a method that depends on true values rather than Boolean logic of the given system. Membership function represents the degree of truth in fuzzy logic.
The sequence of steps involved in this technique is
1. Importing of RGB image and converting it into grayscale image
2.Conversion of the image to double precision data
3.Obtaining the image gradient
4.Defining of the Fuzzy Inference System(FIS) for Edge Detection
5.Specifying the FIS rules
6.Evaluation of FIS rules and plotting of the results.

The Fuzzy rule or the Fuzzy inference system is a collection of rules for a fuzzy system that operates on the given object.
The gaussian membership function used for the inputs is defined by:

$$f(x;\sigma,c) = e^{\frac{-(x-c)^2}{2\sigma^2}}$$

where σ refers to the standard deviation and c refers to the mean.

The triangular membership function used for the output is defined by:

$$f(x;a,b,c) = \begin{cases} 0, & x \le a \\ \frac{x-a}{b-a}, & a \le x \le b \\ \frac{c-x}{c-b}, & b \le x \le c \\ 0, & c \le x \end{cases}$$

where a,b,c represent the start,peak and end of the triangles of the membership functions.

The fuzzy logic approach for image processing allows us to use membership functions to define the degree to which a pixel belongs to an edge or a uniform region.

Histogram equalisation

It's the process of increasing the contrast of an image using its histogram.This method usually increases the global contrast of many images, especially when the usable data of the image is represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. This allows for areas of lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values.

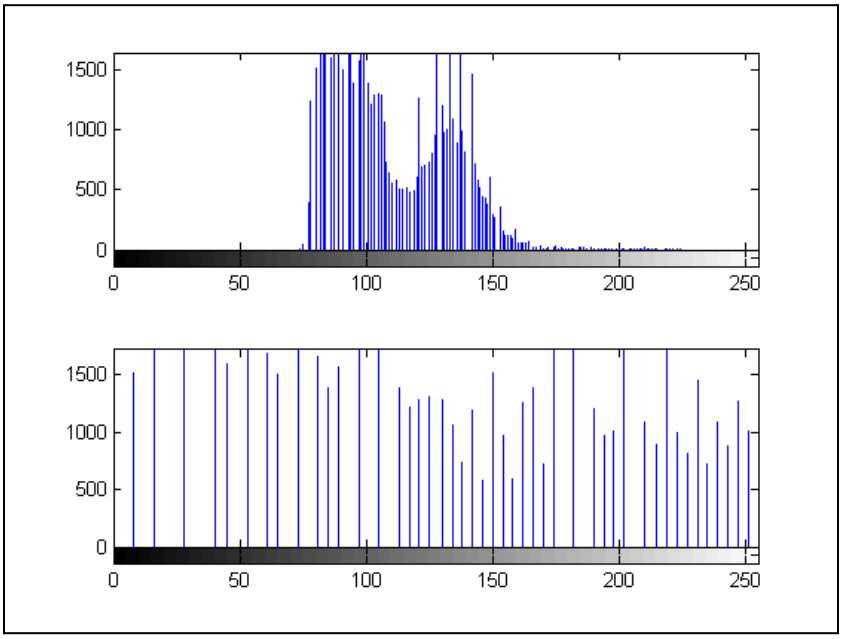

Figure 1 : Histogram Equalisation

Image segmentation using thresholding

Image segmentation is the process of partitioning a digital image into multiple segments /sets of pixels. The goal of segmentation is to simplify and/or change the representation of an image into something that is more meaningful and easier to analyze.
Simple thresholding methods replace all pixel values that live between a certain range with some constant value.

Methodology

Edge detection using fuzzy logic

Under the Edge Detection section,we first convert the image into a grayscale image and then find Image gradients .Image Gradients are used to locate breaks in a uniform image.
The algorithm used for the implementation is mamfis(Mamdani fuzzy inference system)
Zero mean Gaussian membership function is used to input the image into the Fuzzy Inference system and Triangular membership function is used to get the Output Image from the FIS.
In the Gaussian membership function,by changing the values of standard deviations of the inputs,we can control the edge detection performance.
By increasing the standard deviations in the x and y directions, the algorithm becomes less sensitive to edges in the images and decreases the intensity of detected edges.
The intensity of the edge-detected image is specified as output of FIS.
The membership functions of input and output are obtained as shown(might vary slightly depending on the values changed in the membership functions)

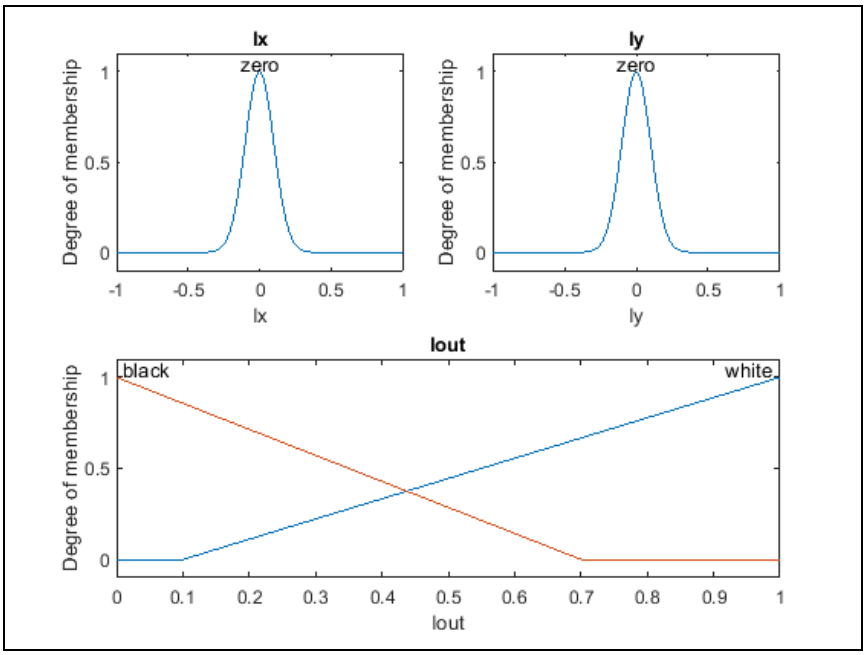

Figure 2 : Plot of membership functions of inputs and outputs of edgeFIS function

Fuzzy Inference Rules are added to make a pixel 'white' if it belongs to a uniform image and 'black' otherwise. A pixel is in a uniform region when the image gradient is zero in both (x and y)directions . If either direction has a non-zero gradient value,the pixel is on edge.
Finally,the output of the edge detector is evaluated for each row of pixels in the image using corresponding rows of Image gradients as inputs.

Histogram Equalisation

Before enhancing the brightness of the image, its histogram is equalised in order to increase its contrast. We h

Algorithm

1. We first calculate the number of occurrences of each pixel value.
2. We then calculate the probability of each frequency, which is given by the number of occurrences of each pixel value divided by the total number of pixels.
3. We find the cumulative histogram of each pixel value by summing up all pixel values before it.
4. The cumulative distribution probability of each pixel is calculated as :

   Cdf of a pixel = Cumulative histogram of that pixel / Total number of pixels (255)

5. The final value of the pixel is given by the product of its Cumulative distribution function and the bins.

Image Enhancement

After we get our image in the screen, edge detection algorithm is applied to the enhanced image.By enhancing the image before further application we can get the minute details in the image in place so that in the later stage while outlining the image it helps in a better overview every detail in the edge of the image.So in order to enhance the image for this purpose we have mainly used two enhancing techniques.

Uniform Brightness adaptation helps to enhance an image as a pre-processing step before analysis. Here we correct the nonuniform background illumination and convert the image into a binary image to make it easy to identify and highlight foreground objects. We can then analyse the image, such as finding the outline of each object present in the image.

Algorithm

1. The coloured image is converted to a grayscale image.
2. Histogram equalization is applied on the image.
3. As a first step, we remove all the foreground objects using morphological opening(imopen).
4. The opening operation(strel) removes small objects that cannot completely contain the structuring element. Here we define a rectangle with the size same as the image to obtain the background.
5. The background is subtracted from the original image.
6. We use the imadjust function to increase the contrast of the image.

Enhancing low light images helps to improve the image in low light conditions. We have defined a function which replicates the **imlocalbrighten** function where darker valued pixels are brightened which is resulted due to the low light condition.

Algorithm

1. The image is histogram equalized and the resulting image is copied from the input image.
2. The pixels(0-255) are divided into sub groups and a series of variables are assigned with various brightness values.
3. Depending on the range of the pixel we assign the corresponding brightness values i.e. a lower value for the brighter pixel range and a higher value of brightness for a darker pixel range.
4. This value is added to the pixel value hence enhancing the low light areas.

We can compare the results of the edge detection before and after enhancement.

Image thresholding

We defined a series of thresholds in order to segment the given image into different image segments based on their intensity levels hence highlighting the different areas to be shaded.

Algorithm

1. We first define the threshold levels based on which the image will be segmented.
2. We then iterate through the pixel values and check the following
    a. Whether the pixel is a boundary by comparing the index with that of the edge detected image, if yes, we don't alter the value.
    b. If the pixel is not a boundary, we perform various levels of thresholding based on the function called so as to display the appropriate number of shades.
    c. If the pixel value does not fit into any of the predefined ranges, we set it to whiite.

Results

The resulting guide interface of the tool with the required options to upload the image from the device/upload via webcam can be noticed. There are buttons for edge detection to view the results before and after the image enhancement process. The shade button gives the different shade levels on the outline of the image.

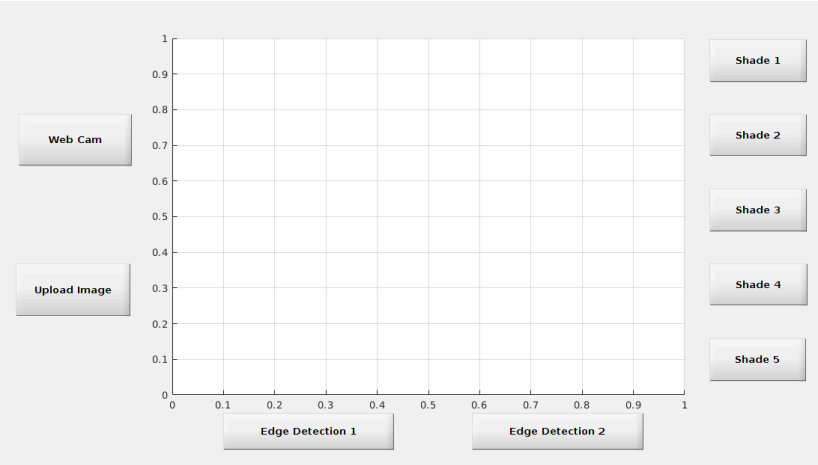

Figure 3 : The graphical user interface

When the upload image button is chosen, the required image is browsed and selected.

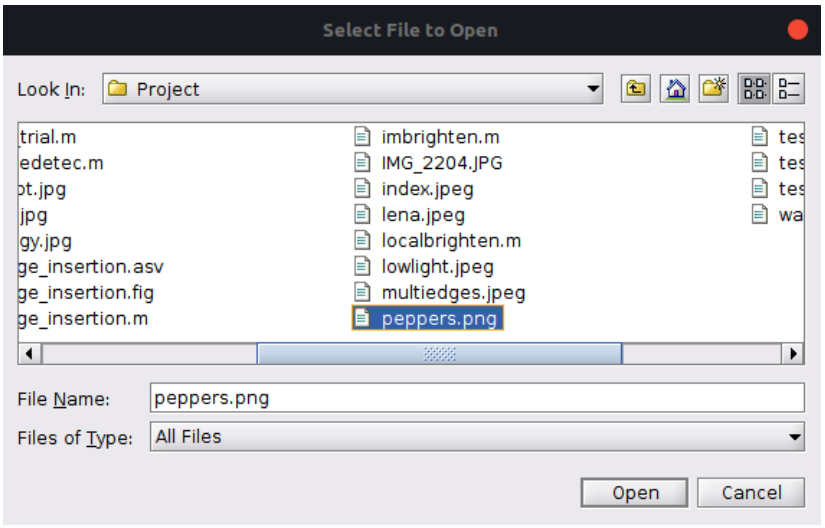

Figure 4 : Image selection

The following figures give us an insight of how the process is carried out.

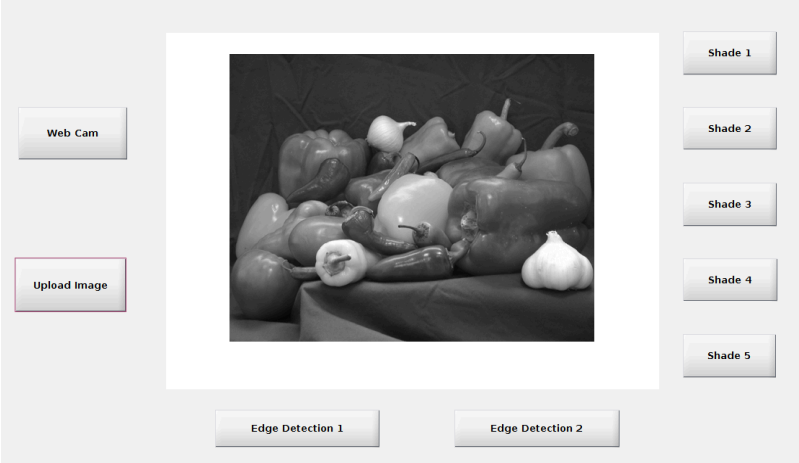

Figure 5 : Original Image to be sketched

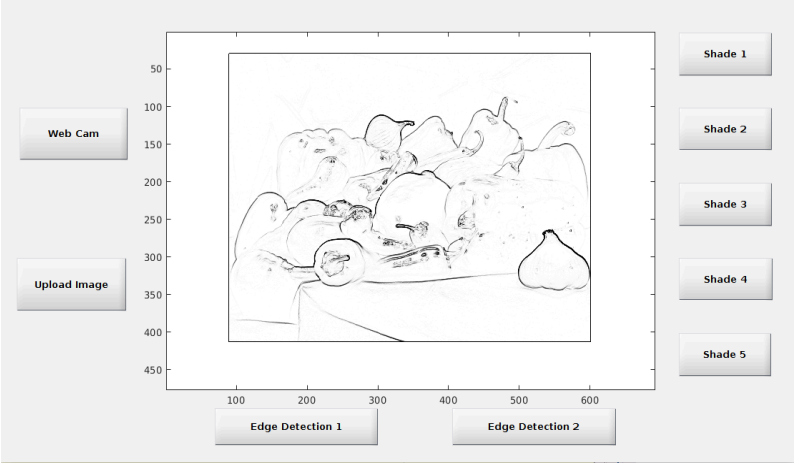

Figure 6 : Edge Detection before Enhancement

Figure 7 :Edge Detection after Enhancement

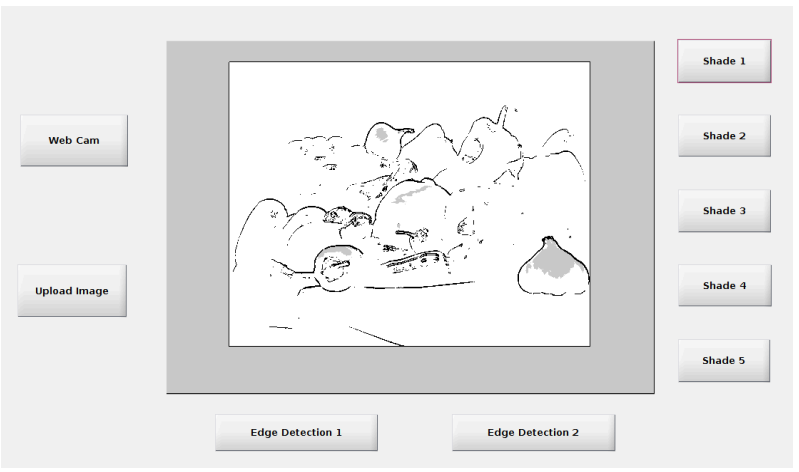

Figure 8: Shade 1 applied on the image

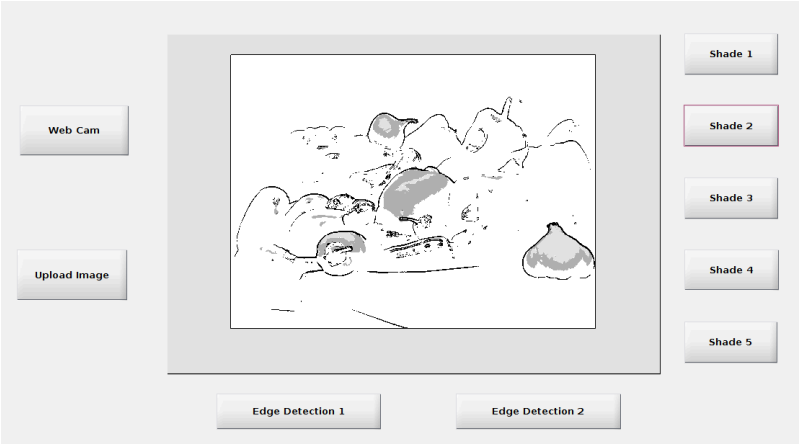

Figure 9 : Shade 2 applied on the image

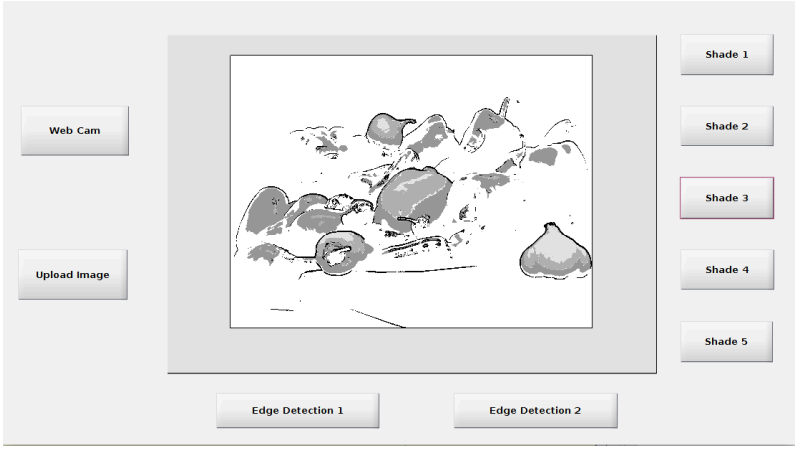

Figure 10 :Shade 3 applied on the image

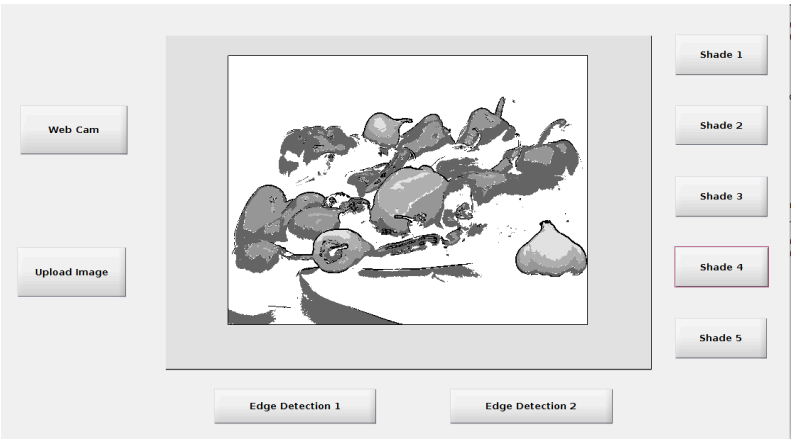

Figure 11 : Shade 4 applied on the image

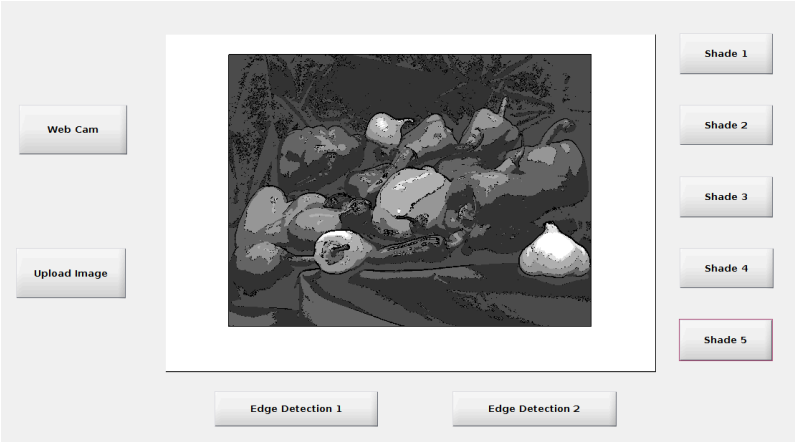

Figure 12 : Shade 5 applied on the image

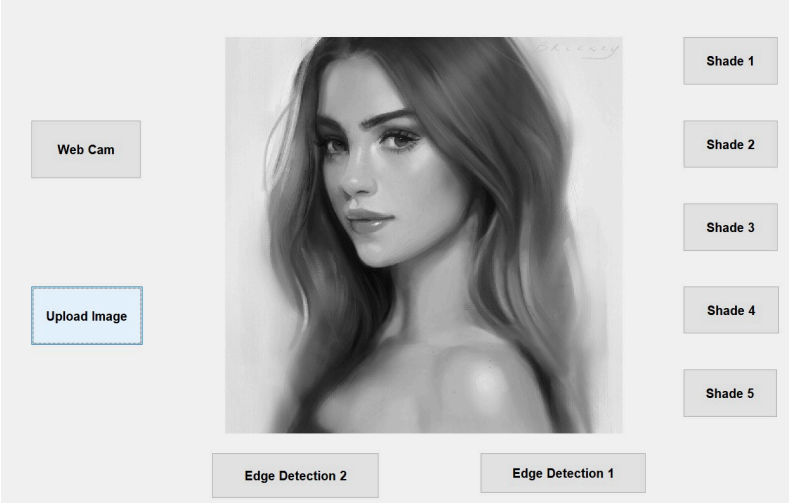

Figure 13 : Original Image to be sketched

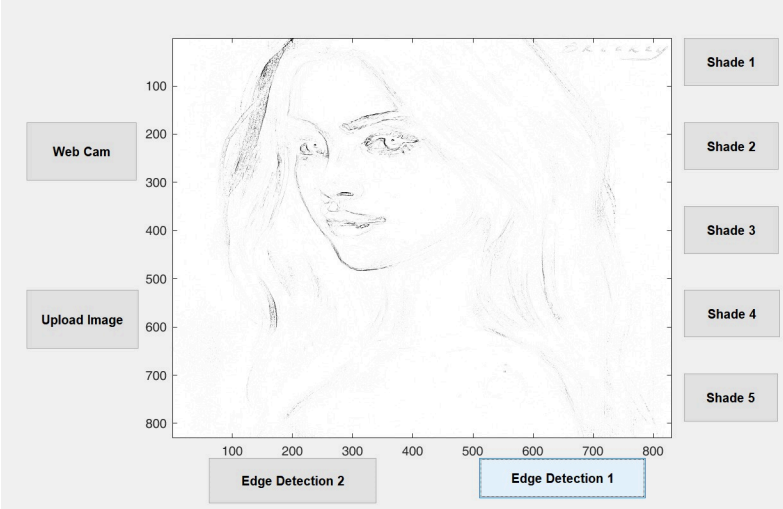

Figure 14 : Edge Detection before Enhancement

Figure 15 :Edge Detection after Enhancement

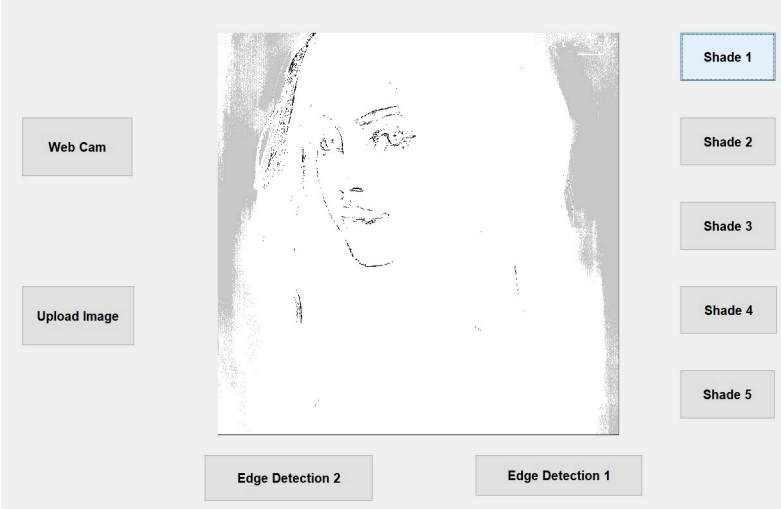

Figure 16 : Shade 1 applied on the image

Figure 17 : Shade 2 applied on the image

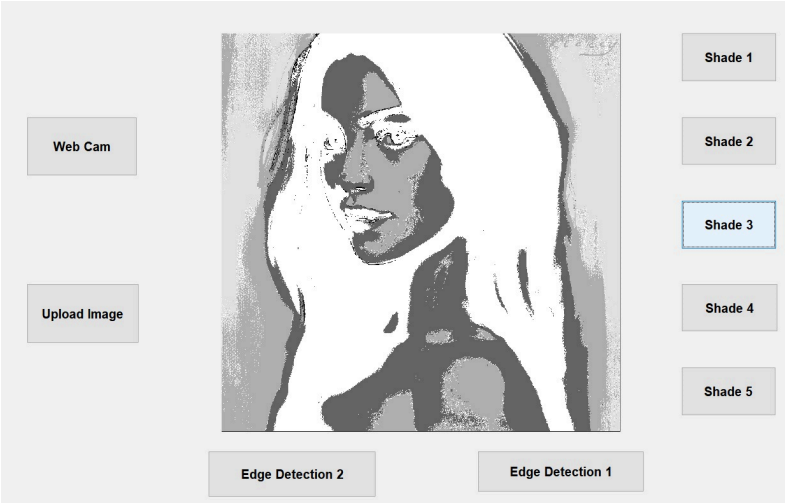

Figure 18 : Shade 3 applied on the image

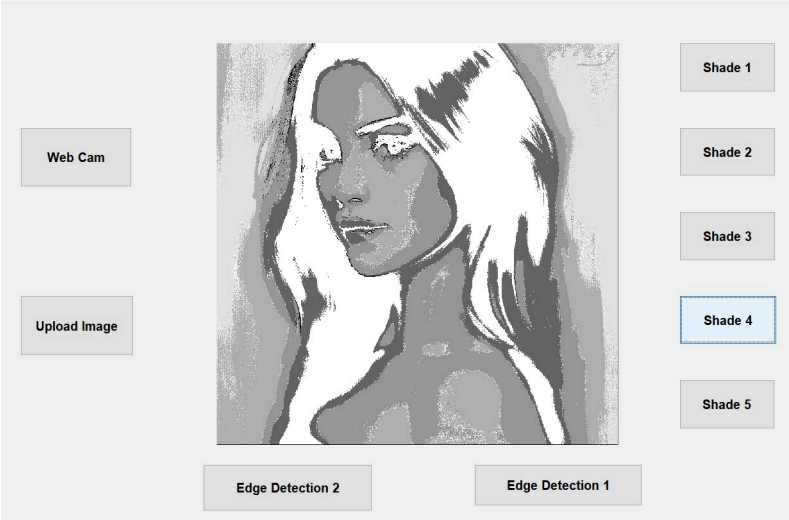

Figure 19 : Shade 4 applied on the image

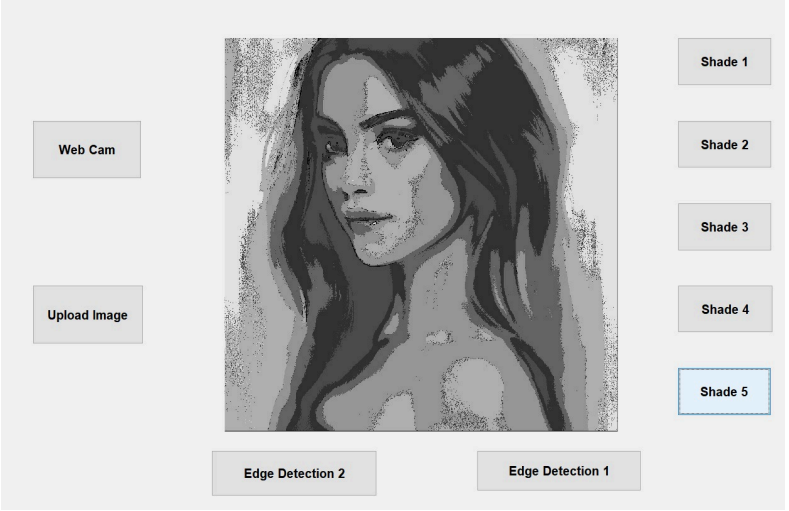

Figure 20 : Shade 5 applied on the image

```
%% Thresholding

function img = showshades1(I,edges)

image = I(:,:,1);
rmin = 50;%decide the min. level of intensity level slicing range
r1 = 90;
r2 = 130;
r3 = 180
rmax = 225;       %decide the max. level of intensity level slicing range
[r,c]= size(image); % get the dimensions of image
s = zeros(r,c);   % pre allocate a variable to store the result image

for i = 1:r
      for j = 1:c
      if (edges(i,j)>0.5)
            if (image(i,j) > rmax)
                  s(i,j) = 200;
            else
                  s(i,j) = 255;
            end
      end
      end
end
img = uint8(s);   %display result image
end

% Thresholding 2

function img = showshades2(I,edges)

image = I(:,:,1);
rmin = 50;%decide the min. level of intensity level slicing range
r1 = 90;
r2 = 130;
r3 = 180
rmax = 225;       %decide the max. level of intensity level slicing range
[r,c]= size(image); % get the dimensions of image
s = zeros(r,c);   % pre allocate a variable to store the result image
for i = 1:r
      for j = 1:c
      if (edges(i,j)>0.5)
```

```
            if(r3 < image(i,j)&& image(i,j) < rmax)
                  s(i,j) = 175;
            elseif(image(i,j) > rmax)
                  s(i,j) = 225;
            else
                  s(i,j) = 255;
            end
      end
      end
end
img = uint8(s);   %display result image
end

% Thresholding 3

function img = showshades3(I,edges)

image = I(:,:,1);
rmin = 50;%decide the min. level of intensity level slicing range
r1 = 90;
r2 = 130;
r3 = 180
rmax = 225;       %decide the max. level of intensity level slicing range
[r,c]= size(image); % get the dimensions of image
s = zeros(r,c);   % pre allocate a variable to store the result image
for i = 1:r
      for j = 1:c
      if (edges(i,j)>0.5)
            if(r2 < image(i,j)&& image(i,j) < r3)
                  s(i,j) = 150;
            elseif(r3 < image(i,j)&& image(i,j) < rmax)
                  s(i,j) = 175;
            elseif(image(i,j) > rmax)
                  s(i,j) = 225;
            else
                  s(i,j) = 255;
            end
      end
      end
end
img = uint8(s);   %display result image
end
```

```
% Thresholding 4

function img = showshades4(I,edges)

image = I(:,:,1);
rmin = 50;%decide the min. level of intensity level slicing range
r1 = 90;
r2 = 130;
r3 = 180
rmax = 225;       %decide the max. level of intensity level slicing range
[r,c]= size(image); % get the dimensions of image
s = zeros(r,c);   % pre allocate a variable to store the result image
for i = 1:r
      for j = 1:c
      if (edges(i,j)>0.5)
            if (r1 < image(i,j)&& image(i,j) < r2)
                  s(i,j) = 100;
            elseif(r2 < image(i,j)&& image(i,j) < r3)
                  s(i,j) = 150;
            elseif(r3 < image(i,j)&& image(i,j) < rmax)
                  s(i,j) = 175;
            elseif(image(i,j) > rmax)
                  s(i,j) = 225;
            else
                  s(i,j) = 255;
            end
      end
      end
end
img = uint8(s);  %display  image

end

% Thresholding 5

function img = showshades5(I,edges)

image = I(:,:,1);
rmin = 50;%decide the min. level of intensity level slicing range
r1 = 90;
r2 = 130;
r3 = 180
rmax = 225;       %decide the max. level of intensity level slicing range
```

```
[r,c]= size(image); % get the dimensions of image
s = zeros(r,c);   % pre allocate a variable to store the result image
for i = 1:r
      for j = 1:c
      if (edges(i,j)>0.5)
            if(image(i,j) > 240)
                  s(i,j) = 255;
            elseif (image(i,j) > rmin && image(i,j)<r1)
                  s(i,j) = 50;
            elseif (image(i,j) < rmin)
                  s(i,j) = 75;
            elseif (r1 < image(i,j)&& image(i,j) < r2)
                  s(i,j) = 100;
            elseif(r2 < image(i,j)&& image(i,j) < r3)
                  s(i,j) = 150;
            elseif(r3 < image(i,j)&& image(i,j) < rmax)
                  s(i,j) = 175;
            elseif(image(i,j) > rmax)
                  s(i,j) = 225;
            end
      end
      end
end
img = uint8(s);  %display  image

end
%% Edge detection using fuzzy logic

function edges = edgedetect(img,sx,sy)
I = im2double(img);

%calc image gradient
Gx = [-1 1];
Gy = Gx';
Ix = conv2(I,Gx,'same');
Iy = conv2(I,Gy,'same');

edgeFIS = mamfis('Name','edgeDetection');

edgeFIS = addInput(edgeFIS,[-1 1],'Name','Ix');
edgeFIS = addInput(edgeFIS,[-1 1],'Name','Iy');

sx = 0.1;
```

```
sy = 0.1;
edgeFIS = addMF(edgeFIS,'Ix','gaussmf',[sx 0],'Name','zero');
edgeFIS = addMF(edgeFIS,'Iy','gaussmf',[sy 0],'Name','zero');

edgeFIS = addOutput(edgeFIS,[0 1],'Name','Iout');

wa = 0.1;
wb = 1;
wc = 1;
ba = 0;
bb = 0;
bc = 0.7;
edgeFIS = addMF(edgeFIS,'Iout','trimf',[wa wb wc],'Name','white');
edgeFIS = addMF(edgeFIS,'Iout','trimf',[ba bb bc],'Name','black');

r1 = "If Ix is zero and Iy is zero then Iout is white";
r2 = "If Ix is not zero or Iy is not zero then Iout is black";
edgeFIS = addRule(edgeFIS,[r1 r2]);
edgeFIS.Rules

Ieval = zeros(size(I));
for ii = 1:size(I,1)
      Ieval(ii,:) = evalfis(edgeFIS,[(Ix(ii,:));(Iy(ii,:))]');
end

edges = Ieval;

end

%% Correcting non unifrom illumination

function I3 = correctIllumination(b)

c=histogrameq(b);
% Illumination Correction
MN=size(c);
background = imopen(c,strel('rectangle',MN));
I2 = imsubtract(c,background);
I3= imadjust(I2);
% Convert to binary image
level = graythresh(b);
d=im2bw(I3,level);
bw = bwareaopen(d, 50);
```

```
end

%% Histogam equalisation

function Himg = histogrameq(img)

nopixel = size(img,1) * size(img,2);

Himg = uint8(zeros(size(img,1),size(img,2)));
freq = zeros(256,1);
probc = zeros(256,1);
probf = zeros(256,1);
cumud = zeros(256,1);
output = zeros(256,1);
sum=0;
no_bins=255;

 for i=1:size(img,1)

      for j=1:size(img,2)

      value=img(i,j);

      freq(value+1)=freq(value+1)+1;

      probf(value+1)=freq(value+1)/nopixel;

      end

 end

 for i=1:size(probf)

   sum=sum+freq(i);

   cumud(i)=sum;

   probc(i)=cumud(i)/nopixel;

   output(i)=round(probc(i)*no_bins);

end
```

```
for i=1:size(img,1)

      for j=1:size(img,2)

            Himg(i,j)=output(img(i,j)+1);

      end

end

end


%% Brightening

function outimg = makebright(inimg)

brightness = 50;
brightness1 = 60;
brightness2 = 0;
brightness3 = 25;
brightness4 = 5;
minVal = 0;
maxVal = 75;
minVal1 = 76;
maxVal1 = 99;
minVal2 = 100;
maxVal2 = 150;
minVal3 = 151;
maxVal3 = 200;
minVal4 = 201;
maxVal4 = 255;
c = imlocalbrighten(inimg);
inimg = histeq(inimg);
outimg = inimg;

for i = 1:size(inimg,1)
      for j = 1:size(inimg,2)
      if (inimg(i,j) >= minVal) && (inimg(i,j) <= maxVal)
      outimg(i,j) = inimg(i,j) +  brightness;
      end
      if (inimg(i,j) >= minVal1) && (inimg(i,j) <= maxVal1)
```

```
      outimg(i,j) = inimg(i,j) +  brightness1;
      end
      if (inimg(i,j) >= minVal2) && (inimg(i,j) <= maxVal2)
      outimg(i,j) = inimg(i,j) +  brightness3;
      end
      if (inimg(i,j) >= minVal3) && (inimg(i,j) <= maxVal3)
      outimg(i,j) = inimg(i,j) +  brightness4;
      end
      if (inimg(i,j) == maxVal4)
      outimg(i,j) = 255;
      end
      end
end

end
```

Bibliography

1. https://in.mathworks.com/help/fuzzy/fuzzy-logic-image-processing.html
2. http://www.dma.fi.upm.es/recursos/aplicaciones/logica_borrosa/web/fuzzy_inferencia/funpert_en.htm#:~:text=Definition%3A%20a%20membership%20function%20for,to%20the%20fuzzy%20set%20A.
3. https://www.google.com/search?channel=fs&client=ubuntu&q=histogram+equalization+matlab
4. https://www.mathworks.com/discovery/image-thresholding.html