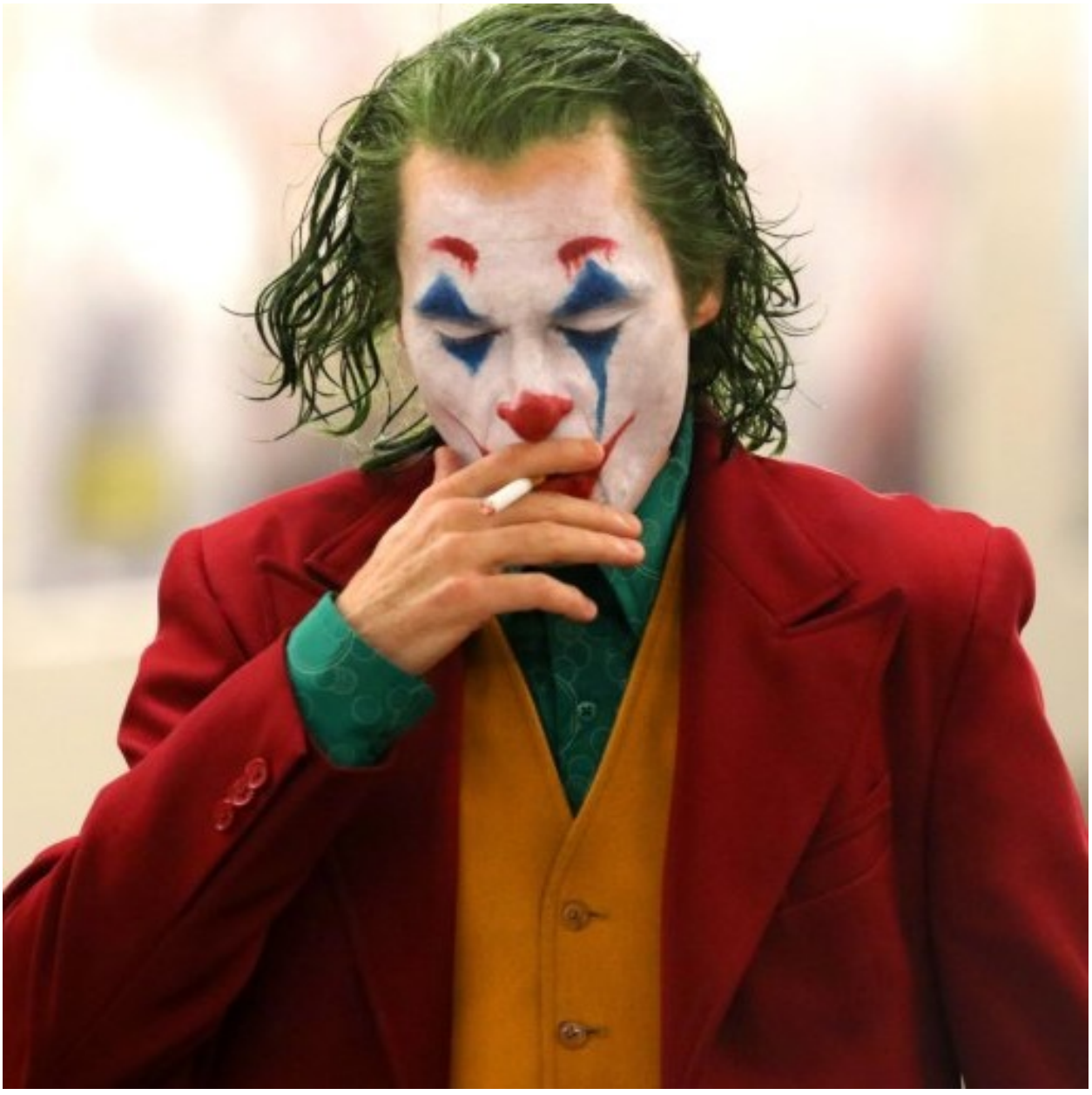
CmpE 362 Homework 3

Taha Eyup Korkmaz – 2014400258

The Original Image



Question 1 - Convolution filter for images

1.1 Part A Blurring

Here I needed to do an image blurring using the a convolution filter with the Gaussian Kernel. I applied the filter on every color matrix to get the fully colored and blurred image. You can see the code

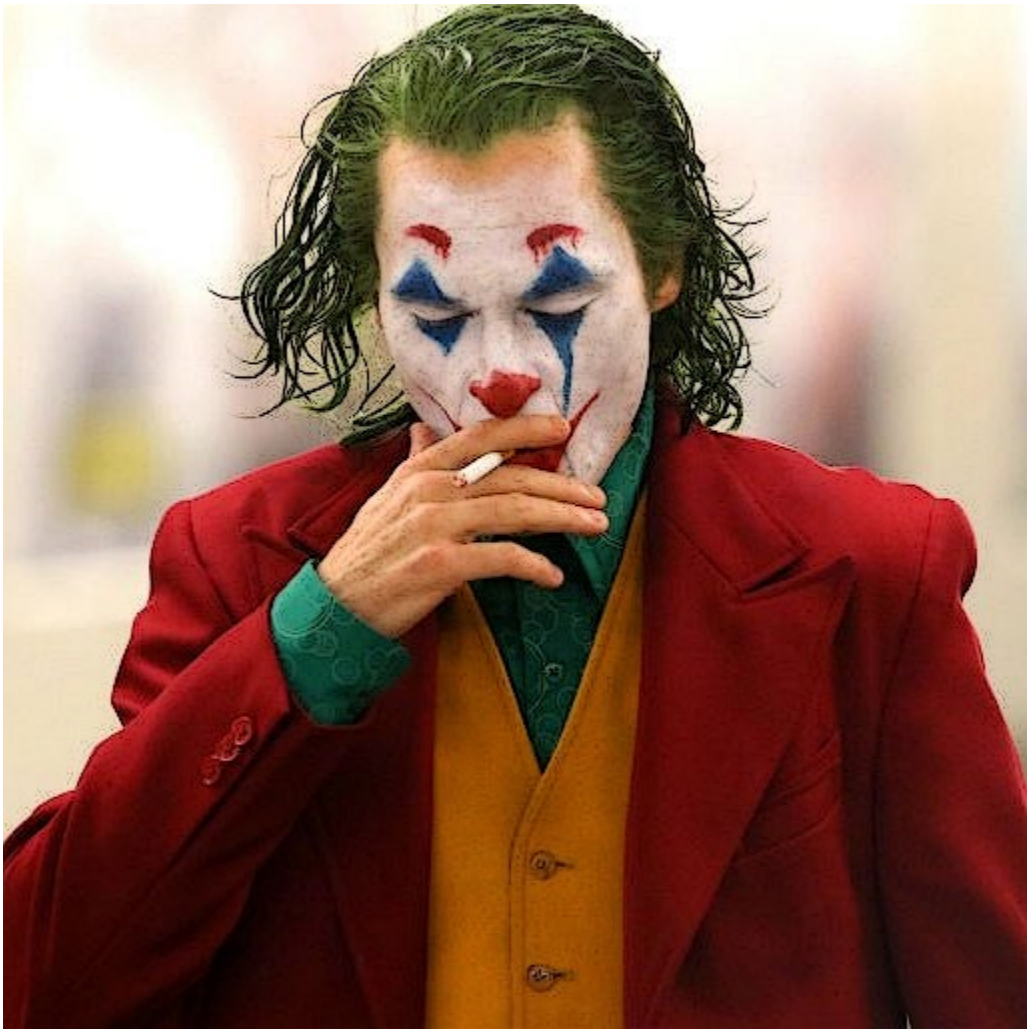
and the output below.

```
%Q1=======================================

%Reading the image
Img = imread('jokerimage.png');
A = imnoise(Img,'Gaussian',0.04,0.003);
%Image with noise
figure,imshow(A);title('The Joker Image');
I = double(A);

%Design the Gaussian Kernel
%Standard Deviation
sigma = 1.76;
%Window size
sz = 9;
[x,y]=meshgrid(-sz:sz,-sz:sz);
```

```
M = size(x,1)-1;
N = size(y,1)-1;
Exp_comp = -(x.^2+y.^2)/(2*sigma*sigma);
Kernel= exp(Exp_comp)/(2*pi*sigma*sigma);

%Initialize
Output=zeros(size(I));
%Pad the vector with zeros
I = padarray(I,[sz sz]);

%Convolution
for i = 1:size(I,1)-M
    for j =1:size(I,2)-N
        Temp = I(i:i+M,j:j+M,1).*Kernel;
        Output(i,j,1)=sum(Temp(:));
    end
end
for i = 1:size(I,1)-M
    for j =1:size(I,2)-N
        Temp = I(i:i+M,j:j+M,2).*Kernel;
        Output(i,j,2)=sum(Temp(:));
    end
end

for i = 1:size(I,1)-M
    for j =1:size(I,2)-N
        Temp = I(i:i+M,j:j+M,3).*Kernel;
        Output(i,j,3)=sum(Temp(:));
    end
end


%Image without Noise after Gaussian blur
Output = uint8(Output);
figure,imshow(Output);title('Blurred Image');
imwrite(Output,'Blurred Image.png','png');
```

1.2 Part B Sharpening

Here I needed to get a convolution filter to reshape the blurred image back to normal. Again, I used filter masks to make sure its back on normal on every color matrix which you can see below.



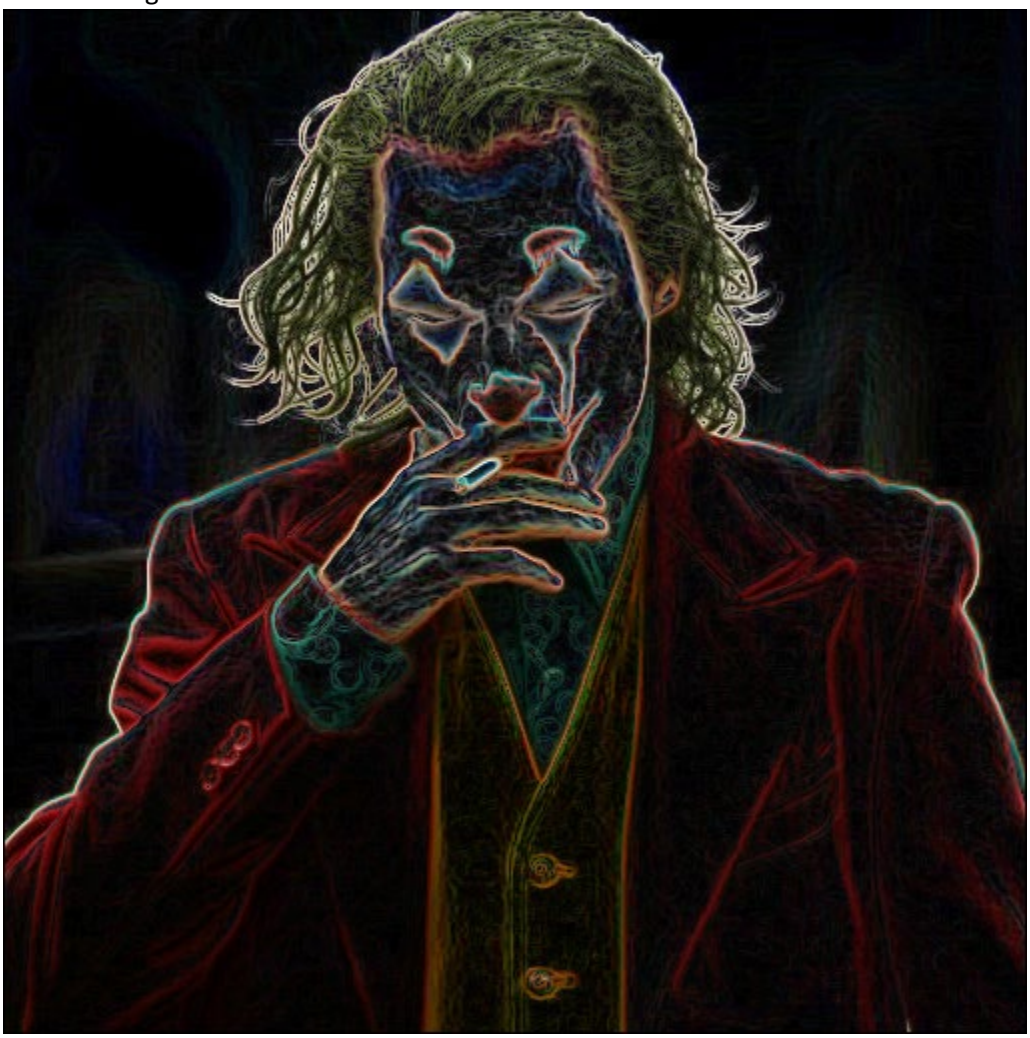
```
%Q2================================================


A=Img;
%Preallocate the matrices with zeros
I1=A;
I=zeros(size(A));
I2=zeros(size(A));

%Filter Masks
F1=[0 1 0;1 -4 1; 0 1 0];
```

```
F2=[1 1 1;1 -8 1; 1 1 1];

%Padarray with zeros
A=padarray(A,[1,1]);
A=double(A);

%Implementation of the equation in Fig.D
for i=1:size(A,1)-2
    for j=1:size(A,2)-2

        I(i,j,1)=sum(sum(F1.*A(i:i+2,j:j+2,1)));

    end
end
for i=1:size(A,1)-2
    for j=1:size(A,2)-2

        I(i,j,2)=sum(sum(F1.*A(i:i+2,j:j+2,2)));

    end
end
for i=1:size(A,1)-2
    for j=1:size(A,2)-2

        I(i,j,3)=sum(sum(F1.*A(i:i+2,j:j+2,3)));

    end
end

I=uint8(I);
%Sharpenend Image
%Refer Equation in Fig.F
B=I1-I;
figure,imshow(B);title('Sharpened Image');
imwrite(B,'Sharpened Image.png','png');
```

1.3 Part C Edge Detecting

In this section I needed to design a kernel that highlights edges in our first image which I used the gradient of the image, then find its vectors and finally use the borders to get the edge detected. Code

and the image are seen below.

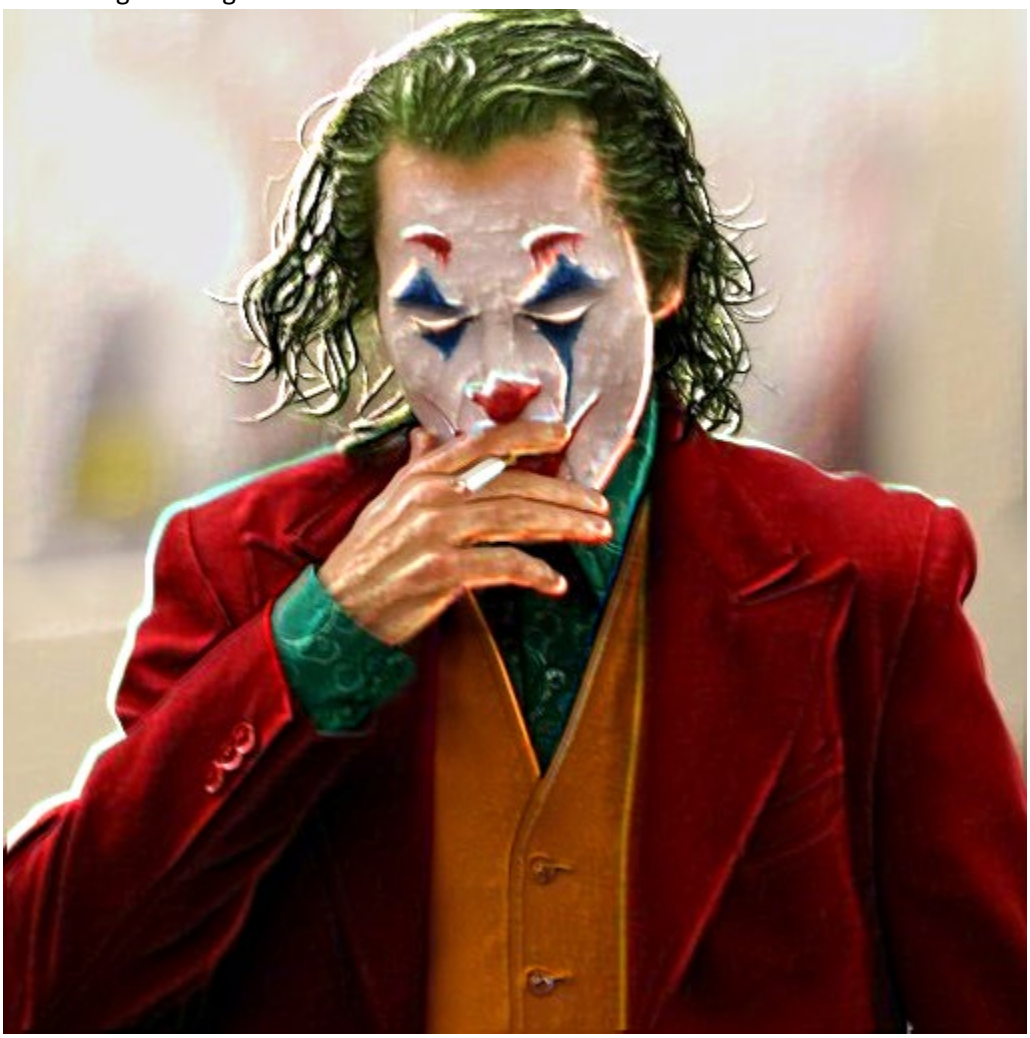


```
%Q3=============================================

%Input Image
A=Img;

%Preallocate the matrices with zeros
I=zeros(size(A));


%Filter Masks
F1=[-1 0 1;-2 0 2; -1 0 1];
F2=[-1 -2 -1;0 0 0; 1 2 1];

A=double(A);
```

```
for i=1:size(A,1)-2
    for j=1:size(A,2)-2
        %Gradient operations
        Gx=sum(sum(F1.*A(i:i+2,j:j+2,1)));
        Gy=sum(sum(F2.*A(i:i+2,j:j+2,1)));

        %Magnitude of vector
         I(i+1,j+1,1)=sqrt(Gx.^2+Gy.^2);

    end
end


for i=1:size(A,1)-2
    for j=1:size(A,2)-2
        %Gradient operations
        Gx=sum(sum(F1.*A(i:i+2,j:j+2,2)));
        Gy=sum(sum(F2.*A(i:i+2,j:j+2,2)));

        %Magnitude of vector
         I(i+1,j+1,2)=sqrt(Gx.^2+Gy.^2);

    end
end


for i=1:size(A,1)-2
    for j=1:size(A,2)-2
        %Gradient operations
        Gx=sum(sum(F1.*A(i:i+2,j:j+2,3)));
        Gy=sum(sum(F2.*A(i:i+2,j:j+2,3)));

        %Magnitude of vector
         I(i+1,j+1,3)=sqrt(Gx.^2+Gy.^2);

    end
end


I=uint8(I);
%figure,imshow(I);title('Filtered Image');

B=Img-(Img-I);
figure,imshow(B);title('Edge Detected Image');
imwrite(B,'Edge Detected Image.png','png');
```

1.4 Part D Embossing

Here I designed a kernel that makes our image embossed using the kernel in the Wikipedia which stated in the description we should and could find it on the internet. Using that I got the embossed version of the text again using convolution which can be seen here.



```
%Q4=================================================
%Reading the image
im=imread('jokerimage.png');
[h w r]=size(im);
%Creating the new image
new_im=zeros(h,w,r);
%Kernel from Wikipedia Embossing Image
kernal=[1 0 0 0 0;0 1 0 0 0;0 0 0 0 0;0 0 0 -1 0;0 0 0 0 -1];
%kernal=[-1 -1 0;-1 0 1;0 1 1];
kernal=int16(kernal);
k_size=5;
hw=floor(k_size/2);
```

```
%Convolution
for i=hw+1:1:h-hw
   for  j=hw+1:1:w-hw
       A=im(i-hw:i+hw,j-hw:j+hw,:);
       A=int16(A);
       A1=A(:,:,1).*kernal;
       A2=A(:,:,2).*kernal;
       A3=A(:,:,3).*kernal;
       new_im(i,j,1)=sum(sum(A1));
       new_im(i,j,2)=sum(sum(A2));
       new_im(i,j,3)=sum(sum(A3));
   end
end
new_im=uint8(new_im);
new_im=(im-24)+new_im;
figure;imshow(new_im);title('Embossed Image');
imwrite(new_im,'Embossed Image.png','png');
```

Final Thoughts

I did not realize these functions are that easy to apply on images which is great because we can realize every image sharing app uses these concepts to change the images. After this project I can safely say we can even create our own kernels for newfound filters. This is one of the eye opening thing that I've learned many in this course.