Adalet Veyis Turgut
2017400210

CMPE 362 HOMEWORK-3 REPORT

Question 1)

Genereal Notes:

- Due to very bright images after the convolution, I divided results with some number
- Weighted sum function takes two n by n matrices and convolute them.

```
% a function to convolute two n by n matrices
function k = weightedSum(x,y,n)
      k = 0;
      for i = 1:n
            for j = 1:n
                  k = k+ x(i,j)*y(i,j);
            end
      end
end
```

- I added two rows and columns that consist of 0's to my image matrix.

```
Padded_Initial_Red = zeros(516);%initializing padded matrices
Padded_Initial_Blue = zeros(516);
Padded_Initial_Green = zeros(516);
Padded_Initial_Red(3:514,3:514) = Initial_Red; %added 2 rows and columns of
zeros
Padded_Initial_Blue(3:514,3:514) = Initial_Blue;
Padded_Initial_Green(3:514,3:514) = Initial_Green;
```

- I found kernel matrices from various sites of web:

Emboss matrix: https://docs.gimp.org/2.6/en/plug-in-convmatrix.html

Edge matrix: https://lodev.org/cgtutor/filtering.html

Blur and Sharpener matrices:
https://www.codingame.com/playgrounds/2524/basic-image-manipulation/filtering

- I hope that “zeros” will not be counted as built-in function because I couldn’t find another way to pad matrices.

Adalet Veyis Turgut
2017400210

Part A)

```
%horizantally and vertically flipped kernel matrix is equal to itself,
%therefore there is no need to flip
Blur_Gaussian_Kernel = [1 / 256, 4  / 256,  6 / 256,  4 / 256, 1 / 256;
    4 / 256, 16 / 256, 24 / 256, 16 / 256, 4 / 256;
    6 / 256, 24 / 256, 36 / 256, 24 / 256, 6 / 256;
    4 / 256, 16 / 256, 24 / 256, 16 / 256, 4 / 256;
    1 / 256, 4  / 256,  6 / 256,  4 / 256, 1 / 256];
Blurred_Green = zeros(512); %initializing blurred image matrices
Blurred_Red = zeros(512);
Blurred_Blue = zeros(512);
for i = 3:514 %slide kernel matrix over image matrices
    for j = 3:514
        Blurred_Green(i-2,j-2) = weightedSum(Blur_Gaussian_Kernel,
Padded_Initial_Green(i-2:i+2,j-2:j+2),5)/250;
        Blurred_Blue(i-2,j-2) = weightedSum(Blur_Gaussian_Kernel,
Padded_Initial_Blue(i-2:i+2,j-2:j+2),5)/250;
        Blurred_Red(i-2,j-2) = weightedSum(Blur_Gaussian_Kernel,
Padded_Initial_Red(i-2:i+2,j-2:j+2),5)/250;
    end
end
Blurred(:,:,1) = Blurred_Red;
Blurred(:,:,2) = Blurred_Green;
Blurred(:,:,3) = Blurred_Blue;
imwrite(Blurred,'Blurred.png','png');
```

$$\frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Gaussian Blur Kernel Matrix

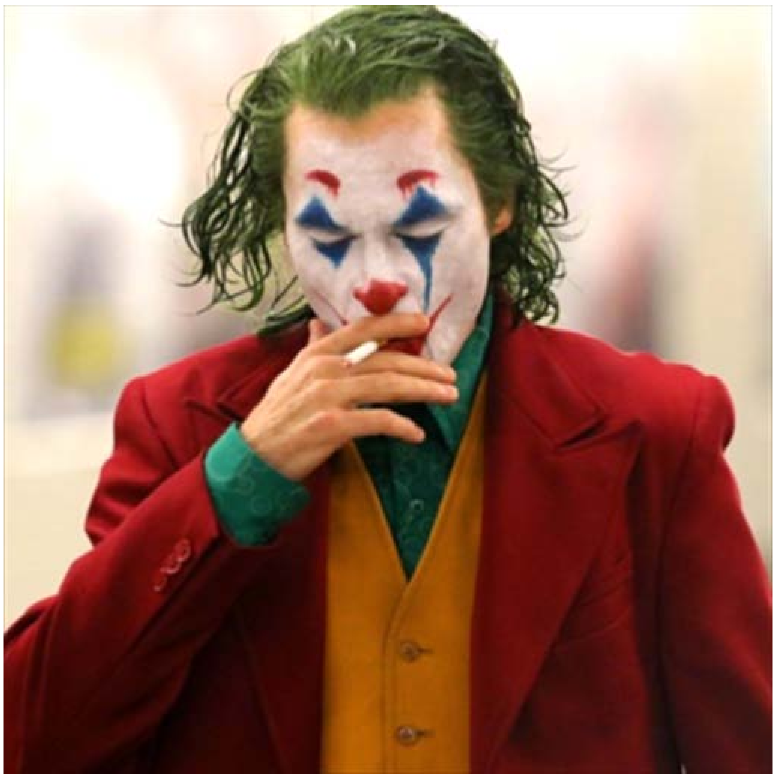
Blurred Image

Part B)

```
%horizantally and vertically flipped kernel matrix is equal to itself,
%therefore there is no need to flip
Sharpener_Kernel = [0, -0.5, 0; -0.5, 3, -0.5; 0, -0.5, 0];

Padded_Blurred_Red = zeros(516);
Padded_Blurred_Blue = zeros(516);
Padded_Blurred_Green = zeros(516);
Padded_Blurred_Red(3:514,3:514) = Blurred_Red; %added 2 rows and columns of
zeros
Padded_Blurred_Blue(3:514,3:514) = Blurred_Blue;
Padded_Blurred_Green(3:514,3:514) = Blurred_Green;

Sharpened_Green = zeros(512);
Sharpened_Red = zeros(512);
Sharpened_Blue = zeros(512);
for i = 3:514
    for j = 3:514
        Sharpened_Red(i-2,j-2) = weightedSum(Sharpener_Kernel,
Padded_Blurred_Red(i-1:i+1,j-1:j+1),3);
        Sharpened_Green(i-2,j-2) = weightedSum(Sharpener_Kernel,
Padded_Blurred_Green(i-1:i+1,j-1:j+1),3);
        Sharpened_Blue(i-2,j-2) = weightedSum(Sharpener_Kernel,
Padded_Blurred_Blue(i-1:i+1,j-1:j+1),3);
    end
end

Sharpened(:,:,1)=Sharpened_Red;
Sharpened(:,:,2)=Sharpened_Green;
Sharpened(:,:,3)=Sharpened_Blue;
imwrite(Sharpened,'Sharpened.png','png');
```

$$\begin{bmatrix} 0 & -0.5 & 0 \\ -0.5 & 3 & -0.5 \\ 0 & -0.5 & 0 \end{bmatrix}$$

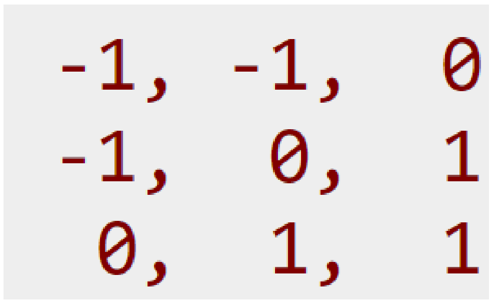
Sharpener Kernel Matrix



Sharpened Image

Part C)

```
Edge_Kernel = [-1,-1,0; -1,0,1; 0,1,1];
Edged_Green = zeros(512);
Edged_Red = zeros(512);
Edged_Blue = zeros(512);
for i = 3:514
    for j = 3:514
        Edged_Green(i-2,j-2)  = weightedSum(Edge_Kernel,
Padded_Initial_Green(i-1:i+1,j-1:j+1),3)/100;
        Edged_Blue(i-2,j-2)  = weightedSum(Edge_Kernel,
Padded_Initial_Blue(i-1:i+1,j-1:j+1),3)/100;
        Edged_Red(i-2,j-2)  = weightedSum(Edge_Kernel,
Padded_Initial_Red(i-1:i+1,j-1:j+1),3)/100;
    end
end
Edged(512,512,3) = zeros;
Edged(:,:,1) = Edged_Red;
Edged(:,:,2) = Edged_Green;
Edged(:,:,3) = Edged_Blue;
imwrite(Edged,'Edged.png','png');
```



```
-1, -1,  0
-1,  0,  1
 0,  1,  1
```

Kernel Matrix to Find Edges

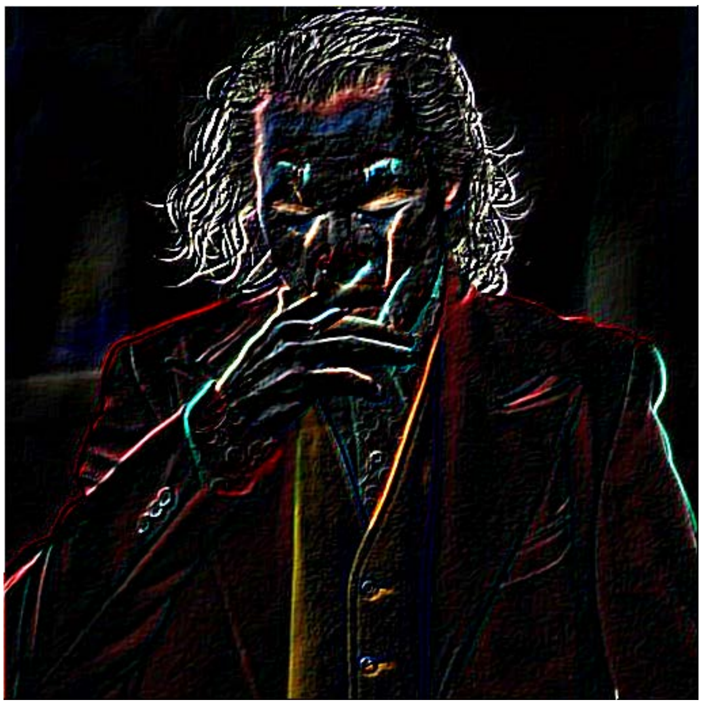
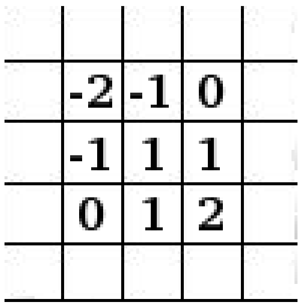


Image With Edges

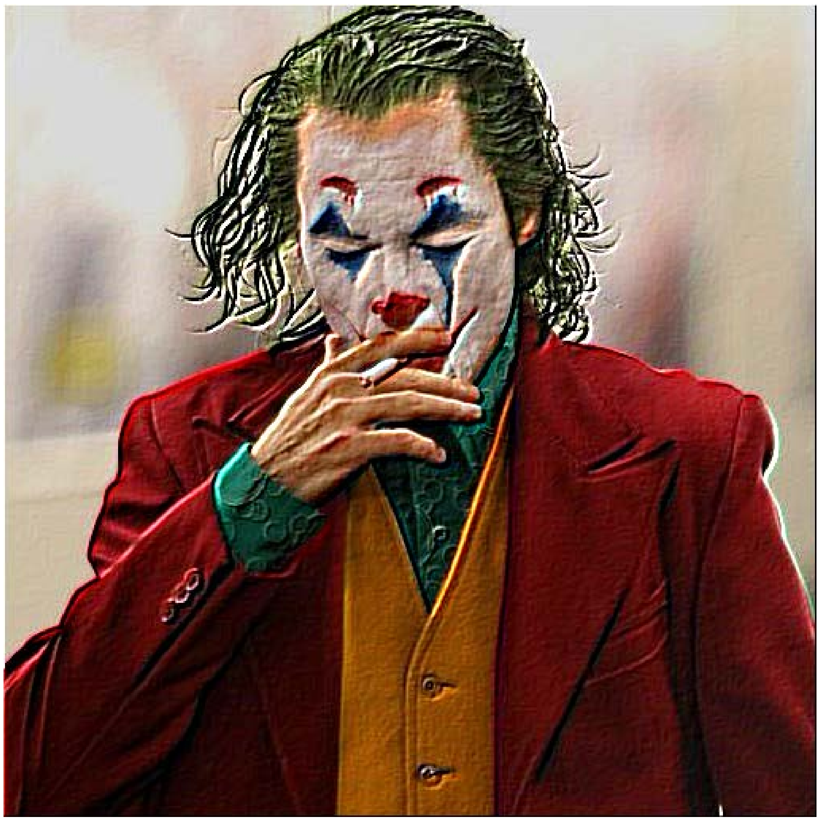
Part D)

```
Emboss_Kernel = [ -2, -1,  0; -1,  1,  1; 0,  1,  2];
Embossed_Green = zeros(512);
Embossed_Red = zeros(512);
Embossed_Blue = zeros(512);
for i = 3:514
    for j = 3:514
        Embossed_Green(i-2,j-2)  = weightedSum(Emboss_Kernel,
Padded_Initial_Green(i-1:i+1,j-1:j+1),3)/300;
        Embossed_Blue(i-2,j-2)  = weightedSum(Emboss_Kernel,
Padded_Initial_Blue(i-1:i+1,j-1:j+1),3)/300;
        Embossed_Red(i-2,j-2)  = weightedSum(Emboss_Kernel,
Padded_Initial_Red(i-1:i+1,j-1:j+1),3)/300;
    end
end
Embossed(512,512,3) = zeros;
Embossed(:,:,1) = Embossed_Red;
Embossed(:,:,2) = Embossed_Green;
Embossed(:,:,3) = Embossed_Blue;
imwrite(Embossed,'Embossed.png','png');
```



-2	-1	0
-1	1	1
0	1	2

Emboss Kernel Matrix



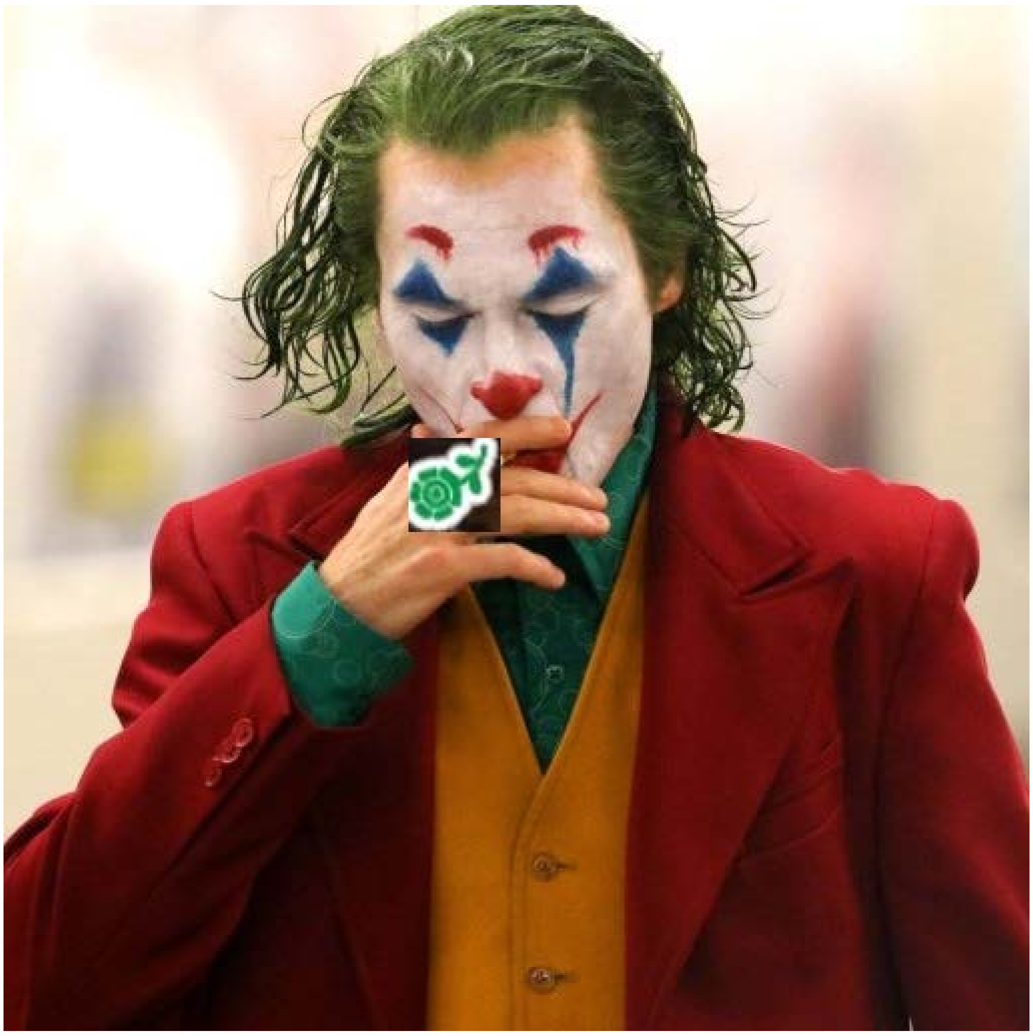
Embossed Image

Adalet Veyis Turgut
2017400210

Question 2)

```
cigarImage = rgb2gray(imread('cigar.png'));  % read images
OriginalJokerImage = imread('jokerimage.png');
% convert image to gray because detectSURFFeatures uses gray images (2
% dimensional matrices)
GrayJokerImage = rgb2gray(OriginalJokerImage);

cigarPoints = detectSURFFeatures(cigarImage); % Detect feature points in
both images
jokerPoints = detectSURFFeatures(GrayJokerImage);

%Extract feature descriptors at the interest points in both images.
[cigarFeatures, cigarPoints] = extractFeatures(cigarImage, cigarPoints);
[jokerFeatures, jokerPoints] = extractFeatures(GrayJokerImage,
jokerPoints);

% take common features
cigarPairs = matchFeatures(cigarFeatures, jokerFeatures);

matchedCigarPoints = cigarPoints(cigarPairs(:, 1), :);
matchedJokerPoints = jokerPoints(cigarPairs(:, 2), :);

%tform has the indexes of matching cigar object in gray joker image
[tform, inlierCigarPoints, inlierJokerPoints] = ...
    estimateGeometricTransform(matchedCigarPoints, matchedJokerPoints,
'affine');

%creating an imaginary rectangle as the size of detected cigar
cigarPolygon = [1, 1;...                           % top-left
        size(cigarImage, 2), 1;...                 % top-right
        size(cigarImage, 2), size(cigarImage, 1);... % bottom-right
        1, size(cigarImage, 1);...                 % bottom-left
        1, 1];                   % top-left again to close the polygon

% defining correct place of rectangle with reindexing tform
newCigarPolygon = transformPointsForward(tform, cigarPolygon);
flower = imread('flower.png');

%taking the middle of the detected object
verticalmiddle = floor((newCigarPolygon(2,1) + newCigarPolygon(4,1))/2);
horizantalmiddle = floor((newCigarPolygon(1,2) + newCigarPolygon(3,2))/2);

%putting flower to the middle of polygon so that it covers cigar
OriginalJokerImage( floor( verticalmiddle - size(flower,1)/2) :
floor(verticalmiddle + size(flower,1)/2) -1 ,...
    floor(horizantalmiddle - size(flower,2)/2) : floor(horizantalmiddle +
size(flower,2)/2) -1,...
    1:3) = flower;

imwrite(OriginalJokerImage,'censored.png','png');
```

Adalet Veyis Turgut
2017400210



Censored Image