# Cmpe362 - Introduction to Signals for Computer Engineers

Spring 2020

**Homework 3**

Afra Akbaş
2015400024

# Contents

**1 Question 1** **2**

1.1 Part A . . . . . 2

1.1.1 Explanation . . . . . 2

1.1.2 Matlab Code . . . . . 2

1.1.3 Image . . . . . 3

1.2 Part B . . . . . 3

1.2.1 Explanation . . . . . 3

1.2.2 Matlab Code . . . . . 4

1.2.3 Image . . . . . 5

1.3 Part C . . . . . 5

1.3.1 Explanation . . . . . 5

1.3.2 Matlab Code . . . . . 5

1.3.3 Image . . . . . 6

1.4 Part D . . . . . 6

1.4.1 Explanation . . . . . 6

1.4.2 Matlab Code . . . . . 7

1.4.3 Image . . . . . 7

**2 Question 2** **8**

2.1 Explanation . . . . . 8

2.2 Matlab Code . . . . . 8

2.3 Images . . . . . 10

# 1 Question 1

In the first question, we did convolution filtering on images. Our goal is to implement convolution and apply it to the given image.
I read the image as a 3-dimensional matrix since it is an RGB image, not a gray-scale. I considered it like it has three 2-dimensional matrices for three colors. (red-green-blue) For all parts in the question, my filters are 2-dimensional 3x3 smaller matrixes, called the "kernel matrix". To apply filters, I flipped the kernel matrixes in both horizontal and vertical directions and slid it over the input matrix. Then, calculate the weighted sum for overlapping regions and write that value into a new matrix. Repeated that for all three colors and formed a new 3-dimensional array for the new image.
To handle edges and corners, I did zero paddings. I expanded image matrices and added both columns and rows that contain zeros. Therefore, the number of rows and columns increased by two.

## 1.1 Part A

### 1.1.1 Explanation

I designed a kernel that adds blur to the image. My blur-filter is [1, 1, 1; 1, 1, 1; 1, 1, 1]. It calculates the mean of kernel and writes it on the center point of the kernel.

### 1.1.2 Matlab Code

```
clear all;
%read image with imread
image = imread('jokerimage.png');
%convert image to double
imageD=double(image);
%size of the image
[m,n,l] = size(imageD);
%define a new array for padding
imagePadding = zeros(m+2, n+2, l);
imagePadding(2:m+1, 2:n+1, :) = imageD;
%declare the kernel
kernel = ones(3,3);
%define output image
output = zeros(m,n,l);
for k = 1:l
    for i = 2:m+1
        for j = 2:n+1
            row1 = i-1;
            row2 = i+1;
            col1 = j-1;
            col2 = j+1;
            subImage = imagePadding(row1:row2, col1:col2, k).*kernel;
            output(i-1, j-1, k) = mean(subImage(:));
        end
    end
end
%convert back to int
output=uint8(output);
imshow(output);
%write image with imwrite
imwrite(output, 'blur.png');
```

### 1.1.3 Image

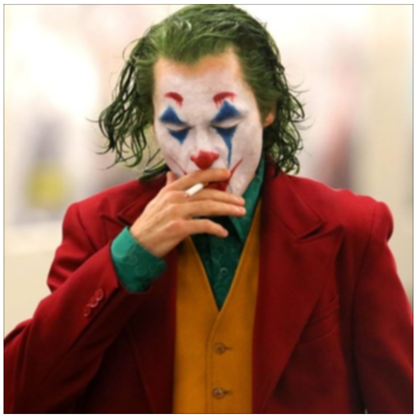

Figure 1: Image with blur

## 1.2 Part B

### 1.2.1 Explanation

I designed a kernel that sharpens the image to get rid of the blur. My filter is [0, -1, 0; -1, 5, -1; 0, -1, 0]. It calculates the sum of kernel and writes it on the center point of the kernel.

### 1.2.2 Matlab Code

```matlab
clear all;
%read image with imread
image = imread('jokerimage.png');
%convert image to double
imageD=double(image);
%size of the image
[m,n,l] = size(imageD);
%define a new array for padding
imagePadding = zeros(m+2, n+2, l);
imagePadding(2:m+1, 2:n+1, :) = imageD;
%declare the kernel
kernel = [0, -1, 0; -1, 5, -1; 0, -1, 0];
%define output image
output = zeros(m,n,l);
for k = 1:l
    for i = 2:m+1
        for j = 2:n+1
            row1 = i-1;
            row2 = i+1;
            col1 = j-1;
            col2 = j+1;

            subImage = imagePadding(row1:row2, col1:col2, k).*kernel;
            output(i-1, j-1, k) = sum(subImage(:));
        end
    end
end
%convert back to int
output=uint8(output);
imshow(output);
%write image with imwrite
imwrite(output, 'sharpen.png');
```

### 1.2.3 Image

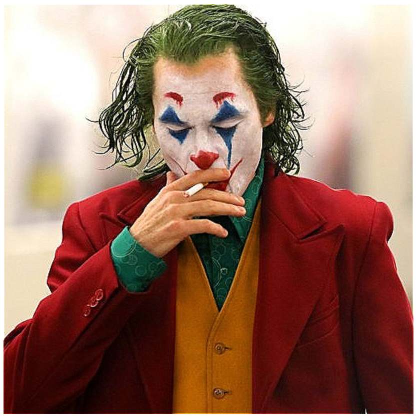

Figure 2: Image with sharpening

## 1.3 Part C

### 1.3.1 Explanation

I designed a kernel that highlights edges in the image. My filter is [-1, 0, 1; -1, 0, 1; -1, 0, 1]. It calculates the sum of kernel and writes it on the center point of the kernel. In this part, I found two kernels to highlight edges. I think both of them work fine but, I added the picture that looks like more.

### 1.3.2 Matlab Code

```
clear all;
%read image with imread
image = imread('jokerimage.png');
%convert image to double
imageD=double(image);
%size of the image
[m,n,l] = size(imageD);
%define a new array for padding
imagePadding = zeros(m+2, n+2, l);
imagePadding(2:m+1, 2:n+1, :) = imageD;
%declare the kernel
%kernel = [1, 0, -1; 1, 0, -1; 1, 0, -1];
kernel = [-1, 0, 1; -1, 0, 1; -1, 0, 1];
%define output image
output = zeros(m,n,l);
```

```
for k = 1:l
    for i = 2:m+1
        for j = 2:n+1
            row1 = i-1;
            row2 = i+1;
            col1 = j-1;
            col2 = j+1;

            subImage = imagePadding(row1:row2, col1:col2,
                k).*kernel;
            output(i-1, j-1, k) = sum(subImage(:));
        end
    end
end
%convert back to int
output=uint8(output);
imshow(output);
%write image with imwrite
imwrite(output, 'edges.png');
```

### 1.3.3 Image

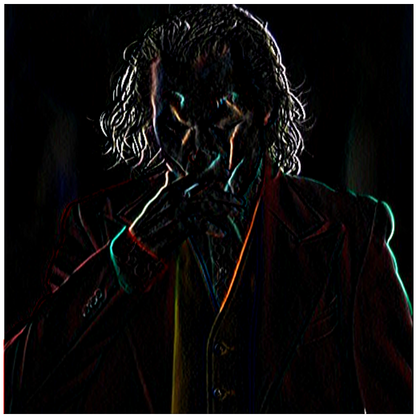

Figure 3: Image with edges

## 1.4 Part D

### 1.4.1 Explanation

I designed a kernel that makes the image embossed. My filter is [1, 1, 1; 1, 1, -1; -1, -1, -1]. It calculates the sum of kernel and writes it on the center point of the kernel.

### 1.4.2 Matlab Code

```
clear all;
%read image with imread
image = imread('jokerimage.png');
%convert image to double
imageD=double(image);
%size of the image
[m,n,l] = size(imageD);
%define a new array for padding
imagePadding = zeros(m+2, n+2, l);
imagePadding(2:m+1, 2:n+1, :) = imageD;
%declare the kernel
kernel = [1, 1, 1; 1, 1, -1; -1, -1, -1];
%define output image
output = zeros(m,n,l);
for k = 1:l
    for i = 2:m+1
        for j = 2:n+1
            row1 = i-1;
            row2 = i+1;
            col1 = j-1;
            col2 = j+1;

            subImage = imagePadding(row1:row2, col1:col2,
                k).*kernel;
            output(i-1, j-1, k) = sum(subImage(:));
        end
    end
end

%convert back to int
output=uint8(output);
imshow(output);
%write image with imwrite
imwrite(output, 'embosed.png');
```

### 1.4.3 Image

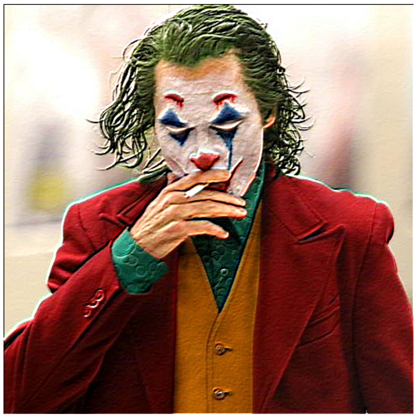

Figure 4: Image embosed

# 2 Question 2

## 2.1 Explanation

In the second question, I detect the cigarette in the image and replaced it with a flower. To do that I cropped the image first where ever I want to
While I was doing the question, I got some help from this link:
https://www.mathworks.com/help/vision/examples/object-detection-in-a-cluttered-scene-using-point-feature-matching.html

## 2.2 Matlab Code

```
clear all;
%read the cropped image
cropedImage = imread('cropedImage.png');
figure;
imshow(cropedImage);
title('Croped Image');

%read the joker image
jokerImage = imread('jokerimage.png');
figure;
imshow(jokerImage);
title('Image of Joker');

%convert images to gray scale
cropedGRAY = rgb2gray(cropedImage);
jokerGRAY = rgb2gray(jokerImage);
```

```matlab

%detect features of the cigarette and the joker
cigarette = detectSURFFeatures(cropedGRAY);
joker = detectSURFFeatures(jokerGRAY);

%plot the strongest points
figure;
imshow(cropedGRAY);
title('100 Strongest Feature Points from Box Image');
hold on;
plot(selectStrongest(cigarette, 100));

figure;
imshow(jokerGRAY);
title('300 Strongest Feature Points from Scene Image');
hold on;
plot(selectStrongest(joker, 300));

%extract feature descriptors
[boxFeatures, cigarette] = extractFeatures(cropedGRAY, cigarette);
[sceneFeatures, joker] = extractFeatures(jokerGRAY, joker);

%match the features using descriptors and plot them
boxPairs = matchFeatures(boxFeatures, sceneFeatures);

matchedBoxPoints = cigarette(boxPairs(:, 1), :);
matchedScenePoints = joker(boxPairs(:, 2), :);
figure;
showMatchedFeatures(cropedGRAY, jokerGRAY, matchedBoxPoints, matchedScenePoints, 'montage');
title('Putatively Matched Points (Including Outliers)');

%locate points and get the coordinates
[tform, inlierBoxPoints, inlierScenePoints] = estimateGeometricTransform(matchedBoxPoints, matchedScenePoints, 'affine');
figure;
showMatchedFeatures(cropedGRAY, jokerGRAY, inlierBoxPoints, inlierScenePoints, 'montage');
title('Matched Points (Inliers Only)');

boxPolygon = [1, 1;...                           % top-left
```

```
        size(cropedImage, 2), 1;...                   % top-right
        size(cropedImage, 2), size(cropedImage, 1);... % bottom-right
        1, size(cropedImage, 1);...                   % bottom-left
        1, 1];                   % top-left again to close the polygon

newBoxPolygon = transformPointsForward(tform, boxPolygon);

figure;
imshow(jokerGRAY);
hold on;
line(newBoxPolygon(:, 1), newBoxPolygon(:, 2), 'Color', 'y');
title('Detected Box');

%read the flower image
figure;
flowerImage = imread('rose.png');

imshow(flowerImage);
title('Flower Image');

%declare boundaries
x_upperBound = uint16(max(newBoxPolygon(:, 2)));
x_lowerBound = uint16(min(newBoxPolygon(:, 2)));
y_upperBound = uint16(max(newBoxPolygon(:, 1)));
y_lowerBound = uint16(min(newBoxPolygon(:, 1)));

%replace selected coordinates with the flower image
figure;
jokerImage(x_lowerBound:x_upperBound, y_lowerBound:y_upperBound, :) = flowerImage;
imshow(jokerImage);
title('New Image of Joker');
imwrite(jokerImage, 'newJoker.png');
```

## 2.3 Images

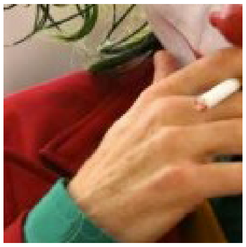

Figure 5: Image that I cropped

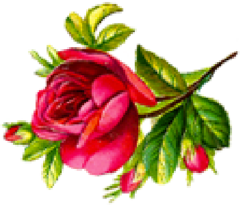

Figure 6: Flower that I used

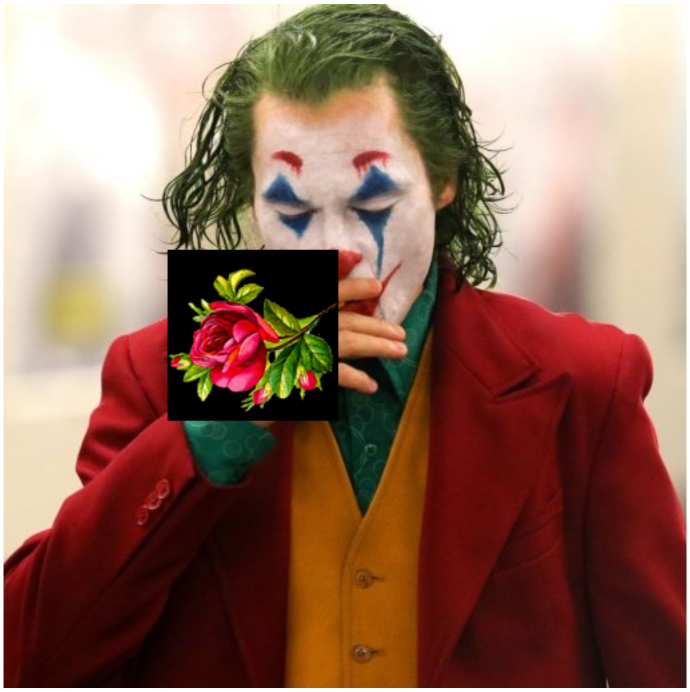

Figure 7: My New Joker Image