# CMPE 362

# INTRO.TO SIGNAL PROC. FOR COMPUTER.ENG.

## 2020 Spring - CMPE 362 Project 3 Report

İsmet Sarı
2016400324

June 22, 2020

# Contents

# 1 First part Convolution Operations

## 1.1 Blurring Joker image

```
clc;
close all;
Img=imread('joker.png');
size_img=length(Img);
Img=double(Img);
%Kernel=[1 4 7 4 1;7 16 26 16 4;7 26 41 26 7;4 16 26 16 4;1 4 7 4 1]/273;
Kernel=[1 2 1; 2 4 2; 1 2 1]/ 16;
%Kernel=ones(5)/25;
size_K=length(Kernel);
N=size_K-1;
one=zeros(size_img+2*N,size_img+2*N);
two=zeros(size_img+2*N,size_img+2*N);
three=zeros(size_img+2*N,size_img+2*N);
one(N+1:size_img+N,N+1:size_img+N)=Img(:,:,1);
two(N+1:size_img+N,N+1:size_img+N)=Img(:,:,2);
three(N+1:size_img+N,N+1:size_img+N)=Img(:,:,3);
out1=zeros(size_img,size_img);
out2=zeros(size_img,size_img);
out3=zeros(size_img,size_img);
for i=1:512-N
    for j=1:512-N
        Temp1=dot(one(i:i+N,j:j+N),Kernel);
        out1(i,j)=sum(Temp1(:));
        Temp2=dot(two(i:i+N,j:j+N),Kernel);
        out2(i,j)=sum(Temp2(:));
        Temp3=dot(three(i:i+N,j:j+N),Kernel);
        out3(i,j)=sum(Temp3(:));
    end
end
o=zeros(size_img,size_img,3);
o(:,:,1)=out1(:,:);
o(:,:,2)=out2(:,:);
o(:,:,3)=out3(:,:);
o=uint8(o);
imwrite(o,"blurred_joker_image.png");
```

Figure 1: Blurring Image

First I read the image with imread function then get the size of the image matrix and convert it to double to make some operation on the image. We have three 2D matrix so, I collect them by Img(:,:,x) and design the Kernel filter I have read a article on Wikipedia and found a paper on internet (you can find the links on references section) and I tried 3 different filters. I found the size of Kernel matrix to use in zero padding section. We may not use zero-padding but this time the resulting matrix will be smaller than the original matrix size by kernel matrix size -1. We create three matrices fulled with all zeros then apply

convolution operation as explained in given project PDF. After all I create 3D matrix and insert the created out1, out2, out3 2D matrices then convert from double to 1-byte (8-bit) unsigned integers and the write that as image by using imwrite function.

$$FirstKernel = 1/273 \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 7 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

I have used Gaussian filter in the PDF that I found on internet. The kernel filter make the image blurred but the resulting image is not something we want so I decided to use something different.

$$SecondKernel = 1/16 \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

This time I have different kind of Gaussian filter that I have found on Wikipedia and again this time I am not satisfied of the result.

$$ThirdKernel = 1/25 \begin{bmatrix} 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 1 & 1 & 1 & 1 \end{bmatrix}$$

Lastly I used this averaging filter and the result absolutely fulfill my expectations.

## 1.2 Sharpening The Image To Get Rid Of The Blur

```
clc;
close all;
Img=imread('blurredimage.png');
size_img=length(Img);
Img=double(Img);
Kernel = [0 -1 0; -1 5 -1; 0 -1 0];
%Kernel=-ones(3);
%Kernel(2,2)=9;
size_K=length(Kernel);
N=size_K-1;
one=zeros(size_img+2*N,size_img+2*N);
two=zeros(size_img+2*N,size_img+2*N);
three=zeros(size_img+2*N,size_img+2*N);
one(N+1:size_img+N,N+1:size_img+N)=Img(:,:,1);
two(N+1:size_img+N,N+1:size_img+N)=Img(:,:,2);
three(N+1:size_img+N,N+1:size_img+N)=Img(:,:,3);
out1=zeros(size_img,size_img);
out2=zeros(size_img,size_img);
out3=zeros(size_img,size_img);
for i=1:512-N
    for j=1:512-N
        Temp1=dot(one(i:i+N,j:j+N),Kernel);
        out1(i,j)=sum(Temp1(:));
        Temp2=dot(two(i:i+N,j:j+N),Kernel);
        out2(i,j)=sum(Temp2(:));
        Temp3=dot(three(i:i+N,j:j+N),Kernel);
        out3(i,j)=sum(Temp3(:));
    end
end
o=zeros(size_img,size_img,3);
o(:,:,1)=out1(:,:);
o(:,:,2)=out2(:,:);
o(:,:,3)=out3(:,:);
o=uint8(o);
imwrite(o,"refined_sharpened_joker_image.png");
```

Figure 2: Sharpening image

In this section I have used a kernel filter that sharpens our image. The blurred image is blurred by using second Gaussian filter in first part. The idea is again similar to first part.

$$Kernel = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 9 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

This kernel is good enough to get rid of the blur however I used another kernel

$$Kernel = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Consequently I used this kernel and the result satisfy me.

### 1.3 Highlighting Edges

```
clc;
close all;
Img=imread('joker.png');
size_img=length(Img);
Img=double(Img);
Kernel = [2 -0 0;0 -1 0;0 0 -1];
%Kernel = [0 1 0;0 0 0;0 -1 0]; %buda iyi fena değil
size_K=length(Kernel);
N=size_K-1;
one=zeros(size_img+2*N,size_img+2*N);
two=zeros(size_img+2*N,size_img+2*N);
three=zeros(size_img+2*N,size_img+2*N);
one(N+1:size_img+N,N+1:size_img+N)=Img(:,:,1);
two(N+1:size_img+N,N+1:size_img+N)=Img(:,:,2);
three(N+1:size_img+N,N+1:size_img+N)=Img(:,:,3);
out1=zeros(size_img,size_img);
out2=zeros(size_img,size_img);
out3=zeros(size_img,size_img);
for i=1:512-N
    for j=1:512-N
        Temp1=dot(one(i:i+N,j:j+N),Kernel);
        out1(i,j)=sum(Temp1(:));
        Temp2=dot(two(i:i+N,j:j+N),Kernel);
        out2(i,j)=sum(Temp2(:));
        Temp3=dot(three(i:i+N,j:j+N),Kernel);
        out3(i,j)=sum(Temp3(:));
    end
end
o=zeros(size_img,size_img,3);
o(:,:,1)=out1(:,:);
o(:,:,2)=out2(:,:);
o(:,:,3)=out3(:,:);
o=uint8(o);
imwrite(o,"joker_image_with_highlighted_edges.png");
```

Figure 3: Highlighting the edges of the image

In this subsection I highlighted Edges of the joker image by using a particular kernel filter and apply convolution operation. The idea is the same as the previous parts.

$$Kernel = \begin{bmatrix} 2 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

I design this kind of kernel and the closest image to the in the given pdf is above. I found this kernel by accidenet. I was designing the embossing part and I changed some part or the matrix then I found this image above. After that I realized that this image is very close to image we want to have so, I have decided to use this kernel. The other kernel filters that I have seen do not give me good results.

## 1.4 Embossing Image

```
clc;
close all;
Img=imread('joker.png');
size_img=length(Img);
Img=double(Img);
Kernel = [2 -0 0;1 -1 0;0 0 -1];
size_K=length(Kernel);
N=size_K-1;
one=zeros(size_img+2*N,size_img+2*N);
two=zeros(size_img+2*N,size_img+2*N);
three=zeros(size_img+2*N,size_img+2*N);
one(N+1:size_img+N,N+1:size_img+N)=Img(:,:,1);
two(N+1:size_img+N,N+1:size_img+N)=Img(:,:,2);
three(N+1:size_img+N,N+1:size_img+N)=Img(:,:,3);
out1=zeros(size_img,size_img);
out2=zeros(size_img,size_img);
out3=zeros(size_img,size_img);
for i=1:512-N
    for j=1:512-N
        Temp1=dot(one(i:i+N,j:j+N),Kernel);
        out1(i,j)=sum(Temp1(:));
        Temp2=dot(two(i:i+N,j:j+N),Kernel);
        out2(i,j)=sum(Temp2(:));
        Temp3=dot(three(i:i+N,j:j+N),Kernel);
        out3(i,j)=sum(Temp3(:));
    end
end
o=zeros(size_img,size_img,3);
o(:,:,1)=out1(:,:);
o(:,:,2)=out2(:,:);
o(:,:,3)=out3(:,:);
o=uint8(o);
imwrite(o,"embossed_joker_image.png");
```

Figure 4: Embossing image

In this section I design a kernel filter and apply convolution operation. The idea is the same as previous ones.

$$Kernel = \begin{bmatrix} 2 & 0 & 0 \\ 1 & -1 & 0 \\ 0 & 0 & -1 \end{bmatrix}$$

I have used this kind of embossing kernel filter that I have read from a paper ( please look at reference part ) and consequently result perfectly fulfill my expectations.

## 2 Bonus Part

In bonus section I design a code that crop the cigarette the joker holds and replace that with a flower.

```
clc;
close all;
original_cigarette= imread('cigarette.png');
original_joker=imread('joker.png');
gray_cigarette = rgb2gray(original_cigarette);
gray_joker = rgb2gray(original_joker);
flower = imread('flower.png');
cigarettePoints = detectSURFFeatures(gray_cigarette);
jokerPoints = detectSURFFeatures(gray_joker);
[cigaretteFeatures, cigarettePoints] = extractFeatures(gray_cigarette, cigarettePoints);
[jokerFeatures, jokerPoints] = extractFeatures(gray_joker, jokerPoints);
mainPairs = matchFeatures(cigaretteFeatures, jokerFeatures);
matchedcigarettePoints = cigarettePoints(mainPairs(:, 1), :);
matchedjokerPoints = jokerPoints(mainPairs(:, 2), :);
[tform, inliercigarettePoints, inlierjokerPoints] = ...
    estimateGeometricTransform(matchedcigarettePoints, matchedjokerPoints, 'similarity');
cigarettePolygon = [1, 1;...                            % top-left
size(gray_cigarette, 2), 1;...                % top-right
size(gray_cigarette, 2), size(gray_cigarette, 1);... % bottom-right
1, size(gray_cigarette, 1);...                % bottom-left
1, 1];                   % top-left again to close the polygon
flowerPolygon = transformPointsForward(tform, cigarettePolygon);
figure;
imshow(joker);
hold on;
image(flowerPolygon(:, 1), flowerPolygon(:, 2), papatya);
```

Figure 5: Cropping cigarette and replacing with a flower

The idea behind the design is the following. First we read the cigarette image that I cropped before and the original joker image and convert them into grayed form since all the function that we are going to use at the rest of the code accept only 2D images. Then I read the flower image and detect the cigarette and main joker images points which we are going to use these point to differentiate the features. After that we create an array with size 2 and insert the features into that array. By using matchFeatures we take the place where the two image overlaps. Then take the point we found before for each image and we check if we can find a similarity between the two images. I gave similarity to estimateGeometricTransform function since the cigarette is very small and the overlapping point will be very small. Then get the coordinates of cigarette that we place with a flower and create a figure plot the original joker and hold on for a while then plot the flower image on this joker image in the coordinate of the cigarette.

# 3 References

# References


[1] The kernel type used in convolution filter. *https://en.wikipedia.org/wiki/Kernel*$(image_processing)$.$(Website)$

[2] The other kernel types used in the operation of convolution. *http://cg.ivd.kit.edu/downloads/assignment3*$_G PUC.pdf$.$(Website)$

[3] Object Detection in a Cluttered Scene Using Point Feature Matching operations used in bonus part *https://www.mathworks.com/help/vision/examples/object-detection-in-a-cluttered-scene-using-point-feature-matching.html*. (Website)

[4] Plowing image on an existing image with a given coordinates. *https://www.mathworks.com/matlabcentral/answers/300826-how-do-i-insert-an-image-in-my-2-d-and-3-d-plots-in-matlab-8-2-r2013b*. (Website)