Digital Signal Processing
-Project1-

Miltiadis Saratzidis
AEM:8237

March 29, 2018

Chapter 1

Convolution - Image with Image

In this part of the project was created 2 different functions with the same final result. In both functions the result is an image representing the convolution of the two input image imX and inY with name $imOut$ and dimensions $(M_1 + M_2 - 1) \times (N_1 + N_2 - 1)$.

1.1 Convolution in the Spatial domain

The first function is called *myConvSpat* and calculates the convolution between the two images in the spatial domain. This is the function that calculates the convolution

```
function imOut = myConvSpat(imX , imY)

    %% Pading in the First Image And Initialization of ImOut
    [M1,N1]=size(imX);
    [M2,N2]=size(imY);

    imYPD=padarray(imY,[M1-1,N1-1],0,'both');
    imOut=zeros(M1+M2-1,N1+N2-1);

    %% Flip of the Image because of the Convolution
    imX=fliplr(imX);
```

```
    imX=flipud(imX);

    %% for loop in all the output image
    for i=1:M1+M2-1
        for j=1:N1+N2-1
            Temp=imYPD(i:(M1-1+i),j:(N1-1+j));
            imOut(i,j)=sum(sum(imX.*Temp));
        end
    end

end
```

More specifically the first step is to pad the imY image in order to have an easier convolution algorithm and then flip the image imX in the horizontal and vertical axis.
Then a for loop is created that runs for all $imOut$ values and multiply every cell of the two arrays. The sum of this values are the result that is been stored in the output array.

1.2 Convolution in the Frequency domain

This function basically has the same exact result image but in a significant changed processing time. One of the most important steps of this procedure is to pad both array in order to achieve the result dimensions, with that way we avoid the circular convolution.

```
function imOut = myConvFreq(imX , imY)

    %% Pading in the First Image And Initialization of ImOut
    [M1,N1]=size(imX);
    [M2,N2]=size(imY);

    %% Pading for stoping the circular convolution
    imX=padarray(imX,[M2-1 N2-1],'post');
    imY=padarray(imY,[M1-1 N1-1],'post');
    %% 2D Fast fourier tranform
```

```
    imXFFT=fft2(imX);
    imYFFT=fft2(imY);

    %% Regular multiplication of the two tables

    OutFFT = imXFFT.*imYFFT;

    %% Inverse tranform
    imOut=real(ifft2(OutFFT));

end
```

Chapter 2

Convolution Image with Mask

In this part of the project the previews functions *myConvSpat* or *myConvFreq* is been used with the purpose of convolving an image with a mask. This is basically the same exact algorithm with the small difference that the second image is a mask, so it is a small image with predetermine values in order to achieve a specific goal.
With that been said the same exact function will be used inside *myFilter* the only think that will be created is a block of *if statements* in order to implement the different statements for different dimensions of the input image and the mask.

```
[M1,N1,Nimg]=size(imX)
[M2,N2,Nmask]=size(aMask)
if Nmask==1
    NChannel=input('1 = One Channel Output, 2 = N Channel Output ');
    if NChannel==1 % One Channel Output
        imX=rgb2gray(imX);imX=im2double(imX);
        imOut = myConvSpat(imX , aMask);
    elseif NChannel==2 % N Channel Output
        imOut=zeros(M1+M2-1,N1+N2-1,Nimg);
        for i=1:Nimg
            Temp=myConvSpat(imX(:,:,i),aMask);
            imOut(:,:,i)=Temp;
        end
```

```
    end
else % Pola kanalia tis maskas
    if Nimg==Nmask
        imOut=zeros(M1+M2-1,N1+N2-1,Nimg);
        size(imOut)
        for i=1:Nimg
            Temp=myConvSpat(imX(:,:,i),aMask(:,:,i));
            imOut(:,:,i)=Temp;
        end
    else
        'error'
    end
end
```

Depending on the different number of channels in the input image and mask the above program will provide different results of puring. The results of both the RGB and GrayScaled images blured are given in the Fig. 3.1.

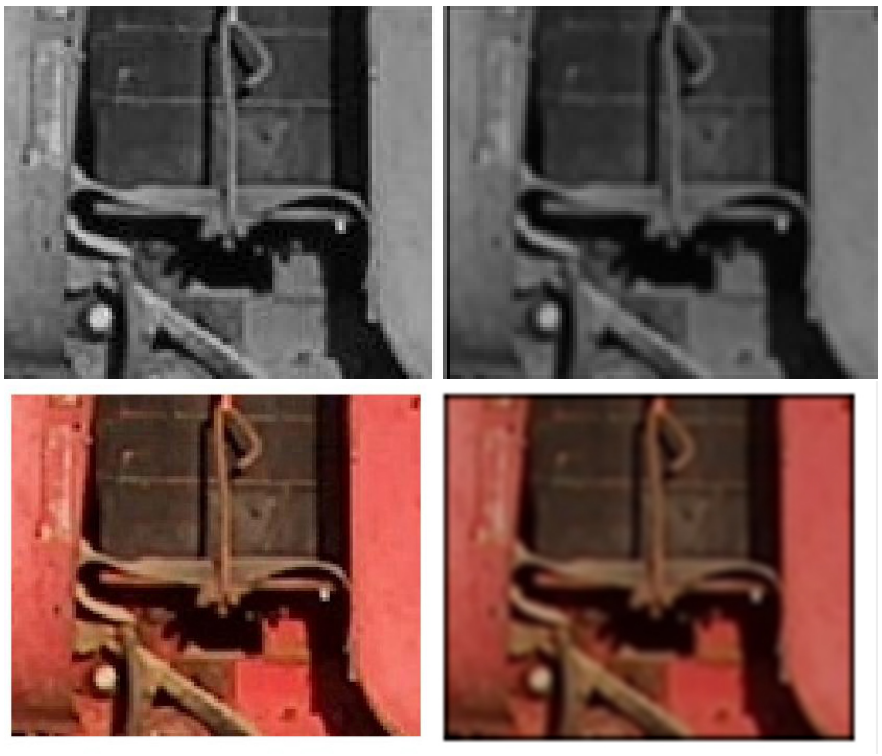


Figure 2.1: GrayScale to Blured GrayScale Image and RGB to Blur RGB Image

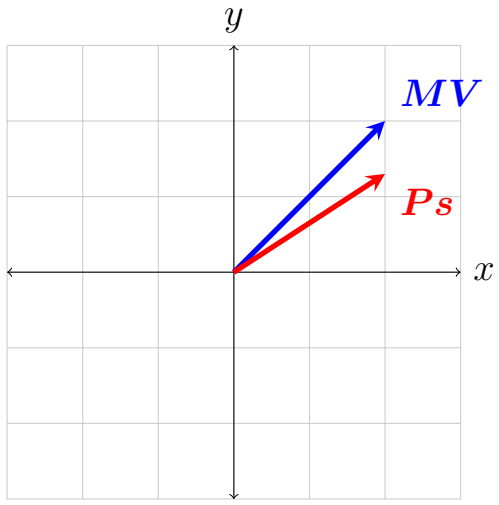
Chapter 3

Motion Blur

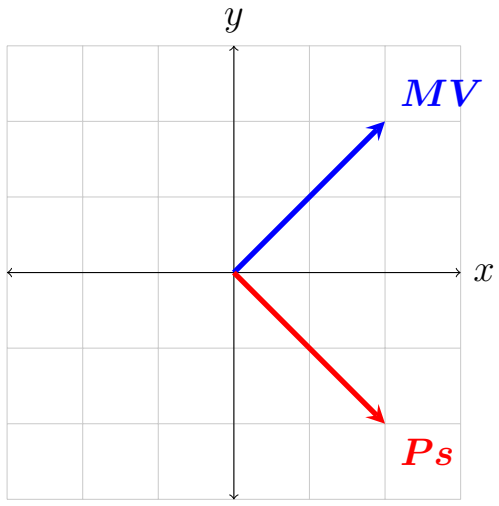
The objective of this chapter is to find the correct mask that will be used with the function *myFilter*. This mask will have the ability to recreate a motion blur in the input image. This motion blur will have a direction $MV = [mvX, mvY](\frac{pixel}{second})$ and time of exposure $Expos(second)$. The vector will be multiplied with the time of exposure with the purpose of calculating the actual distance of motion $[mvX, mvY] = [mvX, mvY] * Expos$.
The size of the mask is calculated with the distance of the motion vector, more specifically the size of the mask is the half of the distance.

$$Distance = \begin{bmatrix} mvX \\ mvY \end{bmatrix} \begin{bmatrix} mvX & mvY \end{bmatrix}$$

The main idea of this algorithm is that, if a positions Ps in the mask is near the line that is been created by the MV and at the same is near the center of the mask will have great values.



In contrast the positions Ps in the mask that are vertical to the MV will be zero or close to zero. With this way it will be achieved a masked that combines high values in positions and close to zero to the other positions which recreates a movement of the camera in MV direction.



This can be achieved by calculating the inter product between the MV and Ps, for every possible position $Ps = [i, j]$ of the mask.

$$InPrMvPs = \begin{bmatrix} mvX \\ mvY \end{bmatrix} \begin{bmatrix} i & j \end{bmatrix}$$

This this value needs to has relatively large values near the center of the mask and low values at the borders of the mask. For that reason it is been used

this equation $FT = 1 - \frac{inPrMvPs}{|MV|^2}$.
Finally is been observed experimentally that the blur results are more significant if a threshold is been in use. More specifically for values less than 0.5 for example the value in the mask is been made zero. Finally a very important step is to calculate the sum of every position in the array and divide this position with the sum. With that way we ensure that the summation of the weights are equal to 1.

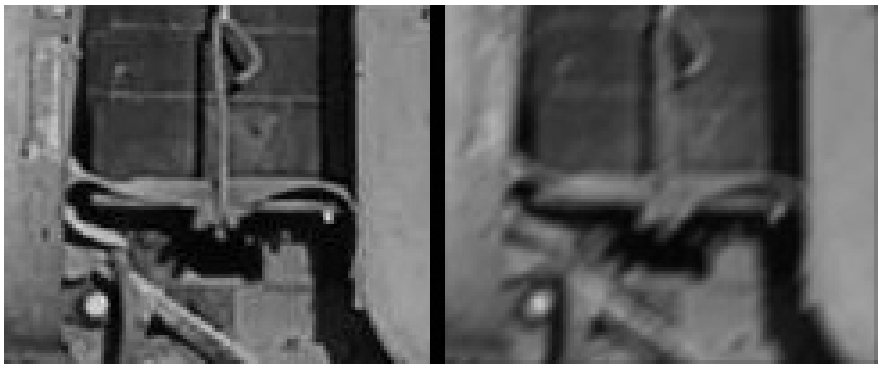


Figure 3.1: GrayScale Image and the Corresponding Blurred Image