# Image Processing
# lab 2
## Group 11

Jan Boonstra (s2774100)     Rick de Jonge (s2775832)

December 10, 2018

## Exercise 1 – Spatial filterings

**a**. To implement a spacial filter, we add a shifted and weighted version of the original image to the output image, this for each cell of the mask. We shift the image by removing rows on one side of the image and adding them on the other side, this way any pixels that would have been outside the image when applying the mask use the values on the opposite boundary of the image, wrapping the values around. The code for this function can be found in Listing 1.

**b**. We implemented gradient magnitude computation in the function IPgradient by first padding the image. Then we initialize a Prewitt or Sobel mask, which we apply to the image twice (using the IPfilter function we made earlier), of which one with a rotated copy of the mask. This way we get the gradient of the image along the x and y axes. Using pythagoras' theorem we combine these two results to get the gradient magnitude. We then return the de-padded version of the resulting calculation. The resulting code can be found in Listing 2. We applied this function to the image *blurrymoon.tif* using the code in Listing 3, the result can be found in Figure 1. In this image, we can see the edges of the original image in white, and smooth areas on the original image are now black.

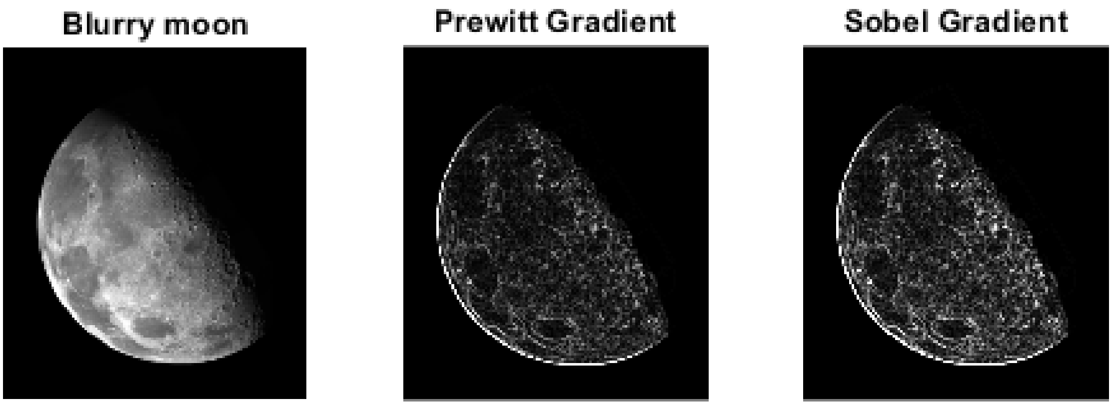


Figure 1: IPgradient.m applied to blurrymoon.tif

**c**. The result would have the opposite effect. When the sharpening mask is applied first, the noise in the image will be enhanced before they are removed. The second pass of the averaging filter will then smooth out this noise over the area around it, making it more visible.

## Exercise 2 – Lowpass filtering in the frequency domain

- In order to make a Gaussian lowpass transfer function, we used the formula $e^{\frac{-D(n,m)}{2D_0}}$, where the function D(n, m) will be the manhattan distance from the center of the mask for each cell on location (n, m). The code for this function can be found in Listing 4.
- Using matlabs built-in Fast Fourier Transform (fft), we were able to convert input images to the frequency domain where it is possible to compute their convolution by applying point-wise multiplication according to the convolution theorem. After applying this multiplication we used the inverse fft to convert the result back to the spacial domain. The code can be found in Listing 5.
- We created the images in Figure 2 using a mask created by IPgaussian using the parameters $D_0$, $N$ and $M$ above the images. These masks, together with the *characters* image were the input for IPftfilter, as can be seen in Listing 6. The new images

## Contribution

The matlab code for exercise 1 was written by Jan, while Rick supported (pair programming). Exercise 2 was partly written by both people. The report was mostly written by Rick, with the help of Jannick, who also reviewed.

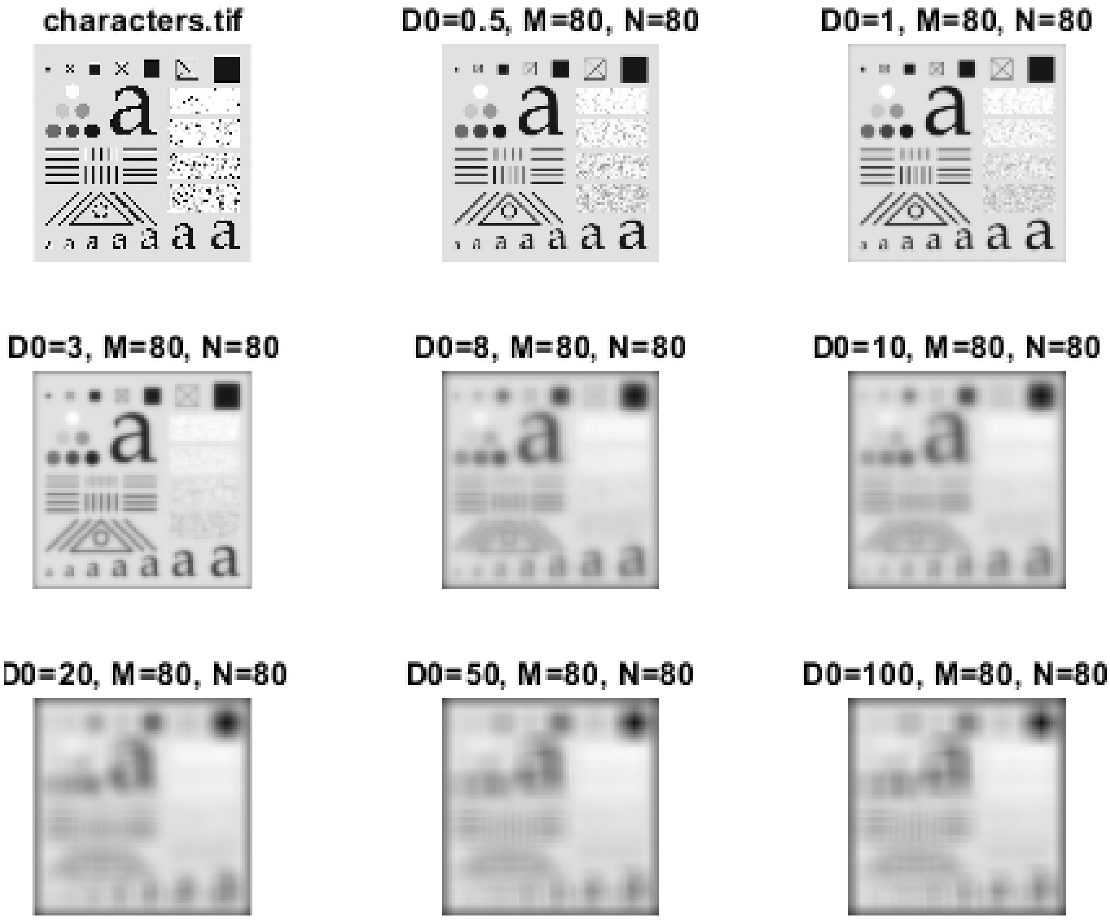


Figure 2: IPftfilter.m applied to characters.tif

# Appendix A The MATLAB code

```
%Performs spacial filtering on image with the specified mask
function out = IPfilter(mask, image)
    [N, M] = size(mask);
    out = zeros(size(image));

    for n = -floor(N/2): floor(N/2)
        for m = -floor(M/2): floor(M/2)
            %Use wrapping to shift image
            copy = image;
            if n > 0
                copy = [copy(n+1:end, :) ; copy(1:n,:)];
            end
            if m > 0
                copy = [copy(:, m+1:end) copy(:, 1:m)];
            end
            if n < 0
                copy = [copy(end+n+1:end, :) ; copy(1:end+n, :)];
            end
            if m < 0
                copy = [copy(:, end+m+1:end) copy(:, 1:end+m)];
            end

            %Add weighted copy to result
            out = out + mask(n + (N+1)/2, m + (M+1)/2) * copy;
        end
    end
end
```

Listing 1: IPfilter.m: A spacial filter for exercise 1a

```
%Computes the gradient magnitude of image using the specified method (1=
    Prewitt, 2=Sobel)
function out = IPgradient(image, method)
    image = double(image);
    image = padarray(image, [1,1], 'replicate');

    %Create mask
    if method == 1
        mask = [-1,0,1;-1,0,1;-1,0,1]; %Prewitt
    else
      mask = [-1,0,1;-2,0,2;-1,0,1]; %Sobel
    end

    %Calculate gradient along axes
    Gx = IPfilter(mask, image);
    Gy = IPfilter(rot90(mask), image);

    %Calculate gradient magnitude
    out = sqrt(Gx.*Gx + Gy.*Gy);
    out = uint8(out(2:size(out,1)-1, 2:size(out,2)-1));
end
```

Listing 2: IPgradient.m: The IPgradient function for exercise 1b

```matlab
%Reset workspace
close all;
clear all;

%Load input image
moon = imread('../images/blurrymoon.tif');

%Show image and its gradient
figure;
colormap(gray(256));
subplot(1,3,1);
imshow(moon);
title('Blurry moon');
subplot(1,3,2);
imshow(IPgradient(moon, 1));
title('Prewitt Gradient');
subplot(1,3,3);
imshow(IPgradient(moon, 2));
title('Sobel Gradient');
```

Listing 3: script2$_1.m : AscriptusingtheIPgradientfunctionoftheimage$blurrymoon.tif

```matlab
%Creates a MxN lowpass Gaussian mask with cutoff frequency D0
function H = IPgaussian(D0, M, N)
    H = zeros(M, N);
    middle = [(M+1)/2, (N+1)/2];

    for i = 1: M
        for j = 1: N
            d = abs(i - middle(1)) + abs(j - middle(2));
            H(i, j) = exp((-d) / (2 * D0));
        end
    end

    %Normalize mask
    H = H / sum(sum(H));
end
```

Listing 4: IPgaussian.m: A gaussian mask generating function for exercise 2a

```matlab
%Perform frequency domain filtering on image x with mask H
function y = IPftfilter(x,H)
    H = rot90(H,2);

    %Pad image and mask to same size
    s = size(x) + size(H) - 1;
    xp = x; xp(s(1),s(2)) = 0;
    Hp = H; Hp(s(1),s(2)) = 0;
    Hp = double(Hp); xp = double(xp);

    %Perform multiplication in frequency domain
    y = ifft2(fft2(Hp) .* fft2(xp));

    %Remove padding
```

```
    y = uint8(real(y));
    b = ceil((s - size(x)) / 2);
    y = y(b(1):s(1)-b(1), b(2):s(2)-b(2));
end
```

Listing 5: IPftfilter.m: A fourier transform filter function for exercise 2b

```
%Reset workspace
close all;
clear all;

%Initialize data
chars = imread('../images/characters.tif');
D0 = [0.5,1,3,8,10,20,50,100];
c = 3;

%Show original image
figure;hold on;
colormap(gray(256));
subplot(ceil((length(D0)+1) / c), c, 1);
imshow(chars);
title('characters.tif');

%Show filtered images for several values of D0
for i = 1:length(D0)
    H = IPgaussian(D0(i), 80, 80);
    r = IPftfilter(chars, H);
    subplot(ceil((length(D0)+1) / c), c, i+1);
    imshow(r);
    title(['D0=' num2str(D0(i)) ', M=80, N=80']);
end
```

Listing 6: script2$_2.m : AscriptusingIPftfilterontheimage$characters.tif