# Image & Video Processing

**Assignment 2 - Local Operations**

Albert Cerfeda

## Contents

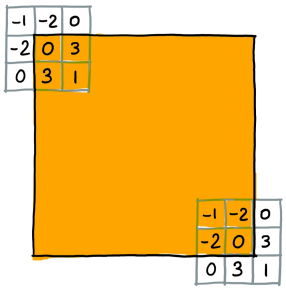



2 April 2023
Università della Svizzera italiana
Faculty of Informatics
Switzerland

# 1 Spatial Filtering [2 points]

**Maximum and minimum values of the convoluted image.**

We know that that each pixel is encoded with in a 6-bit integer format. That is, each pixel has $2^6$ possible values and the minimum and maximum values representable are 0 and $2^6 - 1 = 63$ respectively.
To compute the max and min values of the convoluted image, let us consider the case where the filter is getting applied to the corners of the image. Let us employ the padding technique for dealing with pixels outside the bounds of the image (i.e we add artificial pixels with value 0).
If we consider an image where every pixel has value 63 as a result of the convolution the pixel in the top-left corner will have a value of $(0) * 63 + (3) * 63 + (3) * 63 + (1) * 63 = 441$ while the pixel in the bottom-right corner will have a value of $(-1) * 63 + (-2) * 63 + (-2) * 63 + (0) * 63 = -315$.




**Post-filtering pixel transformation.**
Knowing that each convoluted pixel ranges from $-315$ to 441, we can apply a linear transformation to the pixel values. By mapping each pixel intensity by mapping it $x \mapsto (x + 315)(\frac{63}{441-(-315)})$ we will map the pixel values to the range $[0, 63]$.

**Separability of filter** $H$

# 2 Combining linear operations [2 points]

We know that the unsharp masking operation can be represented with the following formula: i.e we add back to

$$g_{\text{sharp}} = f + \gamma\,(f - h_{\text{blur}} \star f)$$

Figure 1: Unsharp masking operation

the image the details obtained by subtracting the blurred image from the original image.

To compute the kernel reponsible for blurring the image, we can use the Gaussian filter. The gaussian filter computes a weighted average of the pixels in the neighborhood. The weights fall off as the distance increases from the center pixel, according to the gaussian distribution. We can control such gaussian distribution by changing the value of the standard deviation $\sigma$ (i.e control the amount of blur). The size for the kernel matters: to give a good approximated representation of the gaussian function we choose the kernel size to be $4\sigma + 1$.

We notice how we can translate the unsharp masking operation [Figure 1] into a linear operation between kernels [Figure 2]. Notice how the kernel for the original image is just a zero matrix with a 1 in the center. The kernel for the blurred image is the gaussian filter. The difference between the two kernels is the image details.
Parameter $\gamma$ serves as a control parameter for determining the amount of sharpening.
The code for computing the unsharp masking kernel is rather simple [Figure 3]. The results are shown in [Figure 4].

# 3 Morphological operations A [2 points]

Morphological operations are a set of operations that are applied to binary images, in order to alter the represented shapes.
The exercise requires to choose one structuring element and perform multiple morphological operations on the binary in order to obtain the desired result.
By choosing structural element $c$) we can obtain the desired result by performing erosion and dilation in succession. Results are shown in Figure 5:

$$\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix} + \gamma \underbrace{\left( \underbrace{\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}}_{\text{Original pixel}} - \underbrace{\begin{bmatrix} 0.00 & 0.01 & 0.02 & 0.01 & 0.00 \\ 0.01 & 0.06 & 0.10 & 0.06 & 0.01 \\ 0.02 & 0.10 & 0.16 & 0.10 & 0.02 \\ 0.01 & 0.06 & 0.10 & 0.06 & 0.01 \\ 0.00 & 0.01 & 0.02 & 0.01 & 0.00 \end{bmatrix}}_{\text{Blurred pixel}} \right)}_{\text{Image details}}$$

(a) Kernel computation for unsharp masking.

$$\sigma = 1, \gamma = 8$$

$$\begin{bmatrix} -0.02 & -0.11 & -0.18 & -0.11 & -0.02 \\ -0.11 & -0.48 & -0.79 & -0.48 & -0.11 \\ -0.18 & -0.79 & 7.70 & -0.79 & -0.18 \\ -0.11 & -0.48 & -0.79 & -0.48 & -0.11 \\ -0.02 & -0.11 & -0.18 & -0.11 & -0.02 \end{bmatrix}$$

(b) Resulting unsharp masking kernel

Figure 2: Computing the unsharp masking kernel

```
% ...
% Create gaussian blur filter with a gamma parameter of 1
sigma = 1;
gamma = 5;
size = 4*sigma + 1;
blur_f = fspecial('gaussian', size, sigma);
% Identity kernel - leaves the pixel untouched
identity_f = zeros(size);
identity_f(ceil(size/2), ceil(size/2)) = 1;
% Unsharp filter
unsharp_filter = identity_f + gamma.*(identity_f - blur_f);
im_sharp = imfilter(im, unsharp_filter);
```

Figure 3: Matlab code computing the unsharp masking kernel.

(a) Original image

(b) Sharpened image

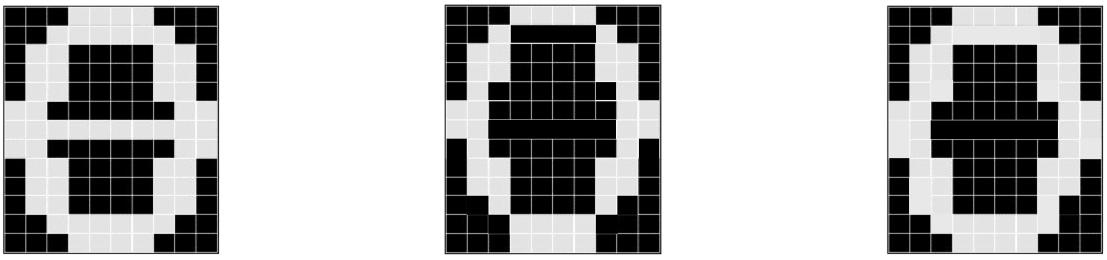
Figure 4: Unsharp masking results.

# 4 Morphological operations B [2 points]

# 5 Linear Motion Blur Filter [4 points]

# 6 Iterative filtering [4 points]

## 6.1 Image Stylization [4 points]



(a) Original binary image (b) Erosion (c) Erosion + Dilation

Figure 5: Morphological operations using structuring element c)