

RWTH Aachen University
Faculty of Mathematics, Computer Science and Natural Sciences
Chair of Computer Science 13 (Computer Vision)
Prof. Dr. Bastian Leibe

Seminar Report

Linear and Nonlinear Filters

Alexander Skretting
Matriculation Number: 445457

Jose Rigel Soeryo Soebandoro
Matriculation Number: 444345

June 2023

Advisor: George Lydakis

Contents

1	Introduction	3
2	Linear Filters	3
2.1	Correlation and Convolution Filtering	3
2.2	Padding (Border Effects)	4
2.3	Separable Filtering	5
2.4	Examples of Linear Filters	6
2.5	Sobel Filter	7
2.6	Band-pass and steerable Filter	7

1 Introduction

2 Linear Filters

With **Linear Filtering**, or specifically linear spatial filtering, the function which is used to pass the image through must be linear and shift invariant. The function \mathbf{L} is considered linear if there exists two constants a and b for any two input pixels $f_1(m, n)$ and $f_2(m, n)$ such that

$$\begin{aligned}\implies g_1(m, n) &= \mathbf{L}[f_1(m, n)] \wedge g_2(m, n) = \mathbf{L}[f_2(m, n)] \\ \implies a \cdot g_1(m, n) + b \cdot g_2(m, n) &= \mathbf{L}[a \cdot f_1(m, n) + b \cdot f_2(m, n)]\end{aligned}$$

also called the *superposition property of linear systems*[BA09]. Whereas a function is considered *shift invariant* when

$$\begin{aligned}\implies g(m, n) &= \mathbf{L}[f(m, n)] \\ \implies g(m - p, n - q) &= \mathbf{L}[f(m - p, n - q)]\end{aligned}$$

p and q implies a spatial shift to both the input and output pixels[BA09]. An intuitive way to think of these two properties is that the function has to behave the same throughout the entire picture.

2.1 Correlation and Convolution Filtering

A common formula for linear filtering is the *Correlation Filtering*[BA09].

$$g(i, j) = \sum_{l \in \mathcal{M}} \sum_{k \in \mathcal{N}} f(i + k, j + l) \cdot h(k, l)$$

or commonly notated as $g = f \otimes h$.

The desired output pixel $g(i, j)$, where i and j specify the coordinates of it, is based on a $M \times N$ sized neighborhood, meaning not only does one pixel define an output pixel, but also a specified number of its neighbors. The influence of each pixel in the neighborhood is defined by the filter coefficient $h(k, l)$, also called its *kernel* or *mask*[Sze22].

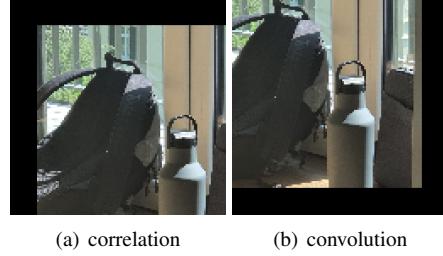
$$\underbrace{\begin{bmatrix} 128 & 34 & 123 \\ 68 & 54 & 73 \\ 100 & 95 & 17 \end{bmatrix}}_{\text{input neighborhood}} \otimes \underbrace{\begin{bmatrix} 0.1 & 0.1 & 0.1 \\ 0.1 & 0.2 & 0.1 \\ 0.1 & 0.1 & 0.1 \end{bmatrix}}_{\text{kernel}} = 75$$

As the above example with a 3×3 kernel, a total of 9 pixels is needed to calculate a single output pixel. Another common variant on the formula is having the signs of the offsets reversed.

$$g = f * h$$

$$g(i, j) = \sum_{l \in \mathcal{M}} \sum_{k \in \mathcal{N}} f(i - k, j - l) \cdot h(k, l)$$

With this formula, $*$ is called the *convolutional operator*, and the kernel h is called the *impulse response function*. An interesting note is that, when the kernel h is convolved with an impulse signal δ (an image with 0 everywhere except the origin), it reproduces the kernel itself $h * \delta = h$, whereas with correlational filtering, it produces the reflected signal (inverted signal in both dimensions).[Sze22] In cases where the kernel is symmetrical on both axis (e.g. box blur), the result of convolutional and correlational is the same.



(a) correlation (b) convolution

Figure 1: Difference between Correlation and Convolution Filtering by translating 32 pixels
An apparent problem from neighborhood filtering is that on the edges, the neighbors simply does not exist in one or two directions (e.g. a 1000×1000 image passed through a 3×3 kernel would produce a 998×998 image). There are a couple method to alleviate the calculation of the nonexistent neighbors.

2.2 Padding (Border Effects)

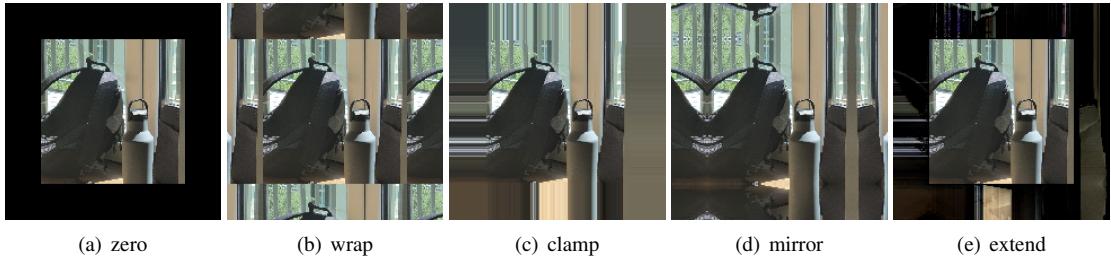


Figure 2: Different types of border padding

- **Zero:** Set all pixels outside the region to zero.
- **Constant:** This is similar to Zero, but instead a specific color is chosen to replace the out of bound pixels.
- **Clamp:** The edge pixels are repeated indefinitely.
- **(Cyclic) Wrap (Repeat or Tile):** The entire image is repeated similar to tiling it infinitely.
- **Mirror:** The image is mirrored accordingly on each of the edges.
- **Extend:** This is a combination of clamping and mirroring, where clamped image is subtracted with the mirrored image.

This of course not only affects the unseen pixels, but the effects can bleed into the pixels inside the frame.

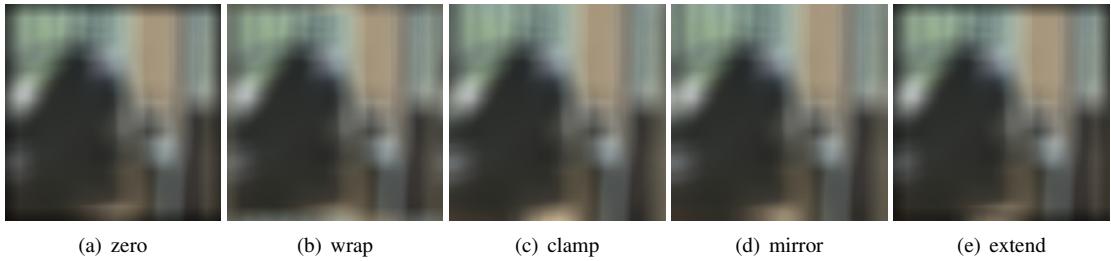


Figure 3: Effects of different paddings with box blur

There are other ways to handle the borders such as two-phase duplication, which is similar to clamp but alternates between the last 2 pixels. A change in the filter function per individual basis can also be implemented, but it might not be practical, as large amounts of additional logic may be needed[BA18].

2.3 Separable Filtering

The time complexity to calculate a single pixel can be said to be $O(M \cdot N)$ as it is dependent on the size of the kernel. For each of the pixel in the kernel, it must be both multiplied and summed together with its neighbors. This can be in practice sped up by separating the horizontal and vertical operations, hence performing a one dimensional convolution horizontally, then one additional operation for the vertical convolution. This kernel in this case is the outer product of two one dimensional kernels[Sze22].

$$\mathbf{K} = v \otimes h = \begin{bmatrix} v_1 h_1 & v_1 h_2 & v_1 h_3 & \dots & v_1 h_n \\ v_2 h_1 & v_2 h_2 & v_2 h_3 & \dots & v_2 h_n \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ v_m h_1 & v_m h_2 & v_m h_3 & \dots & v_m h_n \end{bmatrix}$$

Also notated as $K = vh^T$, where T denotes tranposing h .

A method to check whether a given Kernel is separable is to use *Singular Value Decomposition (SVD)* [Sze22]

$$\mathbf{K} = \sum_i \sigma_i \cdot u_i \cdot v_i^T$$

or more generally

$$\mathbf{K} = \mathbf{U}_{m \times p} \Sigma_{p \times p} \mathbf{V}_{p \times n}^T$$

$$\mathbf{K} = [u_0 \ \dots \ u_{p-1}] \begin{bmatrix} \sigma_0 & & & \\ & \ddots & & \\ & & \sigma_{p-1} & \end{bmatrix} \begin{bmatrix} v_0 \\ \vdots \\ v_{p-1} \end{bmatrix}$$

Note that it requires the number of rows and columns must be the same, it can however be solved by putting 0 as the missing rows/columns, to allow the kernel to be a square matrix. Then the first singular value σ_0 is checked. If it is the only one that is non zero, and the rest is zero, then the kernel is separable. As an example, the sobel filter, which is used for edge detection, can be separated into two separate operations. In this case, this is for the horizontal derivative approximation. [CLLM23]

$$\frac{1}{8} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} = \frac{1}{8} \begin{bmatrix} -1 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

The normalization of the operators are easily calculated by dividing each operator with the absolute sum of the entries.

$$[-1 \ 0 \ 1] \Rightarrow |-1| + |0| + |1| = 2 \Rightarrow \frac{1}{2}$$

Then the normalized operator is

$$\frac{1}{2} [-1 \ 0 \ 1]$$

it applies to the vertical operator as well

$$\frac{1}{4} \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$$

The following are more examples of common separable filtering.

$\frac{1}{K^2} \begin{bmatrix} 1 & 1 & \dots & 1 \\ 1 & 1 & \dots & 1 \\ \vdots & \vdots & \ddots & \vdots \\ 1 & 1 & \dots & 1 \end{bmatrix}$ $\frac{1}{K} [1 \ 1 \ \dots \ 1]$ Box	$\frac{1}{16} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$ $\frac{1}{4} [1 \ 2 \ 1]$ Bilinear	$\frac{1}{256} \begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$ $\frac{1}{16} [1 \ 4 \ 6 \ 4 \ 1]$ Gaussian	$\frac{1}{8} \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$ $\frac{1}{2} [-1 \ 0 \ 1]$ Sobel	$\frac{1}{4} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$ $\frac{1}{2} [-1 \ 2 \ 1]$ Corner
--	---	--	---	---

Note that for Corner filtering, the kernel is not normalized. Normalization is important as otherwise, a pixel may go beyond the allowed value range. In cases where the kernel has a negative value, it means that the pixel value is inverted, since an integer overflow occurs (e.g. $64(-1) = -64 \bmod 256 = 192$ for one channel in an 8 bit pixel). With some filters, it is not always possible to separate it into just two operations, therefore it can be transformed into a series of 1 dimensional operations. However, this may be impractical as it increases the time and space complexity.

2.4 Examples of Linear Filters

- **Box Filter:** each pixel is determined by the average of a $K \times K$ neighborhood of pixels. It is also often called the *moving average*, as the calculation can be done with scanlines, subtracting the old pixel and adding the newest pixels to the running sum. As the pixel is influenced equally by a square neighborhood, this may not be ideal as some ‘square artifacts’ can be seen.
- **Bartlett Filter:** one solution to this is by separably convolving the image with a linear ‘tent’ function (having its peak in the middle). A 3×3 version of this is called the bilinear kernel.
- **Approximate Gaussian Filter:** if the Bartlett kernel is convolved with itself, this will produce an approximate version of the gaussian kernel.

Essentially, the three above filters are different mathematical functions applied both horizontally and vertically.

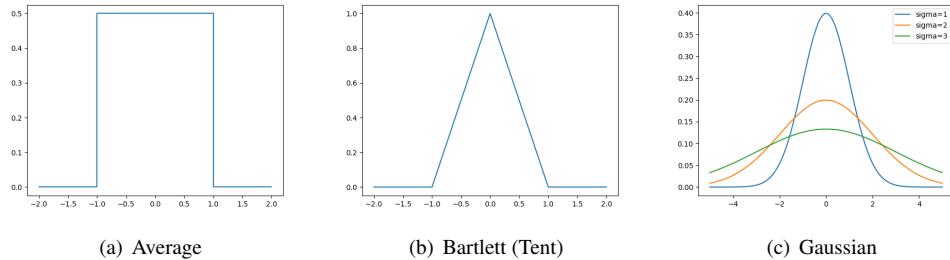


Figure 4: One dimensional graphs of the filters

As its name suggests, the Gaussian Filter is based on the Gaussian function (also commonly known as the normal function). Two dimensionally, it can be expressed as [MW20]

$$G_{2D}(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

It can be seen from the graph that the lower the sigma value, the narrower the neighborhood becomes. From testing, using the gaussian blur, as opposed to the box filter produces a better image, as it preserves rounded corners, however, to produce an equally ‘blurred’ image, a larger kernel is required for Gaussian, which can be argued to then be slower than using the box filter. These filters are mainly used to smoothen(blur) the image, where the low frequencies are let through as well as reducing the effect of the high frequencies (high frequencies and low frequencies both referring to the brightness/darkness of each pixel). In practical applications, this is done to reduce high frequency noises. A creative application to smoothing the images, is that the smoothen images can be used as a mask to sharpen the image by adding the difference between the original and blurred image. This method is called *unsharp masking*[Sze22].

$$g_{sharp} = f + \gamma(f - h_{plur} * f)$$

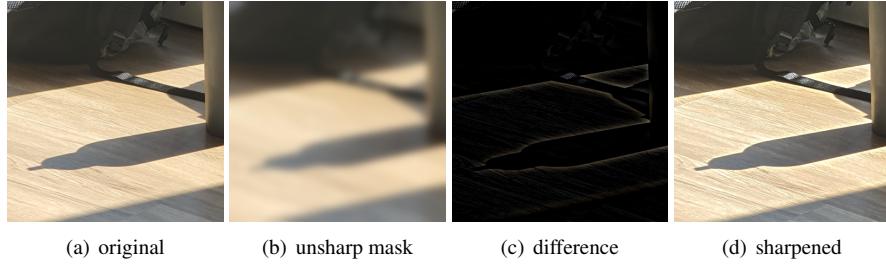


Figure 5: Unsharp masking with Gaussian blur

The result of the image traces a bright outline throughout the image between the darker(shadow) and lighter part. It can also be seen, that the difference between the blurred image and the original results in an approximate edge detection. A better way to extract edges by means of linear filtering is by the use of the *Sobel filter*. [Sze22]

2.5 Sobel Filter

As per the example in the previous section, the horizontal Sobel operator is calculated by combining the *horizontal central difference* (which calculates the second horizontal derivative of the pixel and places it in the center itself) and a vertical linear tent filter to smooth the result.

The two sobel operator emphasizes the horizontal or vertical edges respectively, although as seen in the images, it can also highlight diagonal edges.

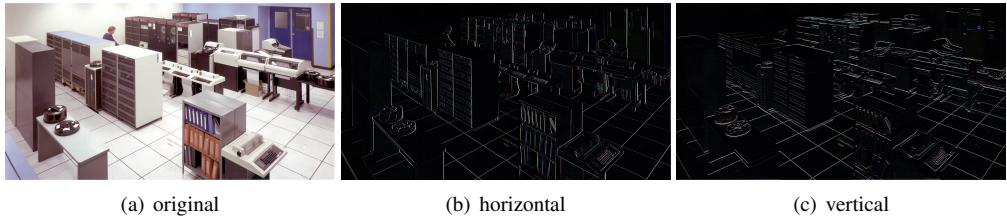


Figure 6: A picture of the Data Acquisition System at the Ames Research Center Vertical and Horizontal Sobel filtering[NAS90]

2.6 Band-pass and steerable Filter

In the above example, the sobel filter is calculated either for the horizontal or vertical axis. Nevertheless, it is possible to calculate filters that uses a specified *directional derivative* $\nabla_{\hat{u}} = \frac{\partial}{\partial \hat{u}}$ and a *unit direction* $\hat{u} = (\cos \theta, \sin \theta)$ [Sze22].

$$\hat{u} \cdot \nabla(G * f) = \nabla_{\hat{u}}(G * f) = (\nabla_{\hat{u}} G) * f$$

Then it is also preferable to smooth it with the Gaussian filter.

$$G_{\hat{u}} = uG_x + vG_y = u\frac{\partial G}{\partial x} + v\frac{\partial G}{\partial y}$$

This is called the *directional* or *oriented* filter. Another filter is called the *Laplacian operator*, which is the second (undirected)derivative.[Sze22]

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

and again, it can be smoothed with the Gaussian filter. It is equivalent to the *Laplacian of Gaussian*(LoG) filter.

$$\nabla^2 G(x, y, \sigma) = \left(\frac{x^2 + y^2}{\sigma^4} - \frac{2}{\sigma^2}\right) G(x, y, \sigma)$$

The aforementioned filters are more sophisticated as they require the second or first derivatives and they can filter both the low and high frequencies. These filters are called *band-pass filters*. Since the filters are based on the gradient fields, which are undirected, it can be ‘steered’ by freely choosing a unit direction, hence the name *steerable* filter. The advantage of this is that multiple variations of directions can easily be calculated with minimal cost as the derivative must only be calculated once. In case of higher order derivatives, the calculation must be done repeatedly with the same unit vector. This can although be calculated with a relatively small number of functions[Sze22]. Consider the example below for the second-order derivatives, which requires 3 basis functions.

$$G_{\hat{u}\hat{u}} = u^2 G_{xx} + 2uv G_{xy} + v^2 G_{yy} \text{ with } \hat{u} = (u, v)$$

References

- [BA09] Alan C. Bovik and Scott T. Acton. Chapter 10 - basic linear filtering with application to image enhancement. In Al Bovik, editor, *The Essential Guide to Image Processing*, pages 225–239. Academic Press, Boston, 2009.
- [BA18] Donald G. Bailey and Anoop S. Ambikumar. Border handling for 2d transpose filter structures on an fpga, Nov 2018.
- [CLLM23] Qiong Chang, Xiang Li, Yun Li, and Jun Miyazaki. Multi-directional sobel operator kernel on gpus. *Journal of Parallel and Distributed Computing*, 177:160–170, 2023.
- [MW20] Siddharth Misra and Yaokun Wu. Chapter 10 - machine learning assisted segmentation of scanning electron microscopy images of organic-rich shales with feature extraction and feature ranking. In Siddharth Misra, Hao Li, and Jiabo He, editors, *Machine Learning for Subsurface Characterization*, pages 289–314. Gulf Professional Publishing, 2020.
- [NAS90] NASA, Feb 1990. [Online; accessed June 24, 2023].
- [Sze22] Richard Szeliski. 3.2 *Linear filtering*, page 119–131. Springer, 2nd edition, 2022.