



Published in Image Processing On Line on 2013–12–17.  
 Submitted on 2013–04–29, accepted on 2013–11–02.  
 ISSN 2105–1232 © 2013 IPOL & the authors CC–BY–NC–SA  
 This article is available online with supplementary materials,  
 software, datasets and online demo at  
<https://doi.org/10.5201/ipol.2013.87>

# A Survey of Gaussian Convolution Algorithms

Pascal Getreuer

CMLA, ENS Cachan ([getreuer@cmla.ens-cachan.fr](mailto:getreuer@cmla.ens-cachan.fr))

*Communicated by* Luis Álvarez     *Demo edited by* Pascal Getreuer

## Abstract

Gaussian convolution is a common operation and building block for algorithms in signal and image processing. Consequently, its efficient computation is important, and many fast approximations have been proposed. In this survey, we discuss approximate Gaussian convolution based on finite impulse response filters, DFT and DCT based convolution, box filters, and several recursive filters. Since boundary handling is sometimes overlooked in the original works, we pay particular attention to develop it here. We perform numerical experiments to compare the speed and quality of the algorithms.

## Source Code

ANSI C source code to produce the same results as the demo is accessible on the [IPOL web page of this article](#)<sup>1</sup>. Future software releases and updates will be posted at <http://dev.ipol.im/~getreuer/code>.

**Keywords:** filtering, convolution

## 1 Introduction

This work surveys algorithms for the computation and fast approximation of Gaussian convolution,

$$u(x) = (G_\sigma * f)(x) := \int_{\mathbb{R}^d} G_\sigma(x - y)f(y) dy, \quad G_\sigma(x) = (2\pi\sigma^2)^{-d/2} \exp\left(-\frac{\|x\|_2^2}{2\sigma^2}\right), \quad (1)$$

where  $f$  is the input signal,  $u$  is the filtered signal, and where  $G_\sigma$  is the Gaussian (see figure 1) with standard deviation  $\sigma$ . Gaussian convolution is a building-block operation used in many signal and image processing algorithms. To name a few prominent examples, Gaussian convolution is used in Gabor filtering [9, 25], Canny edge detection [6], and SIFT feature detection [18].

While we focus on Gaussian convolution, many of the ideas here can be applied more broadly, such as other kernels and spatially-varying filtering [10, 20, 22, 28].

<sup>1</sup><https://doi.org/10.5201/ipol.2013.87>

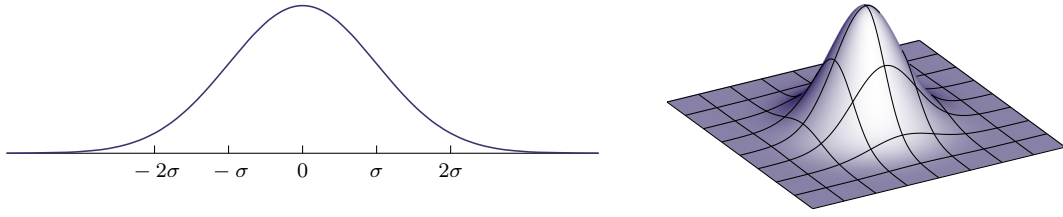


Figure 1: The Gaussian in one and two dimensions.

### 1.1 Notations

Notations  $\lfloor x \rfloor$  and  $\lceil x \rceil$  denote respectively the floor and ceiling functions,  $(\cdot)^*$  denotes complex conjugation,  $\|v\|_p := (\sum_n |v_n|^p)^{1/p}$  denotes the  $\ell^p$  norm of sequence  $v$ , and  $\|v\|_\infty := \sup_n |v_n|$  the  $\ell^\infty$  norm of  $v$ . We use the following definition of the continuous Fourier transform,

$$\hat{f}(\xi) := \int_{-\infty}^{\infty} f(x)e^{-2\pi i x \xi} dx, \quad \check{g}(x) := \int_{-\infty}^{\infty} g(\xi)e^{2\pi i x \xi} d\xi, \tag{2}$$

where  $\hat{\cdot}$  denotes the forward transform and  $\check{\cdot}$  the inverse transform. For discrete filters, define the Z-transform

$$\mathcal{Z}\{h\}(z) := \sum_{n=-\infty}^{\infty} h_n z^{-n}. \tag{3}$$

### 1.2 Properties

Compared to other smoothing kernels, the Gaussian has a unique combination of properties that make it especially attractive. A few well-known properties are listed below.

- **Separability.** The Gaussian is separable:

$$\exp\left(-\frac{x_1^2 + x_2^2}{2\sigma^2}\right) = \exp\left(-\frac{x_1^2}{2\sigma^2}\right) \exp\left(-\frac{x_2^2}{2\sigma^2}\right). \tag{4}$$

- **Semigroup property.** Gaussians have the semigroup property that convolution of two Gaussians is another Gaussian [14, 15],

$$G_{\sigma_1} * G_{\sigma_2} = G_\sigma, \quad \sigma = \sqrt{\sigma_1^2 + \sigma_2^2}. \tag{5}$$

Thus the convolution  $u = G_\sigma * f$  may be implemented as two convolutions,  $w = G_{\sigma_1} * f$  and  $u = G_{\sigma_2} * w$ , possibly using two different algorithms.

- **Extrema properties.** In one dimension, Gaussians are the only filters among a broad class that do not create or enhance local extrema or create new zero-crossings in the second derivative [4, 5, 14, 19].
- **Locality in space and frequency.** Gaussians are optimally localized in space and frequency in the sense of the Heisenberg–Weyl inequality [1, 2]: for any nonzero function  $\psi \in L^2(\mathbb{R})$ ,  $\psi : \mathbb{R} \rightarrow \mathbb{C}$ , and any fixed but arbitrary constants  $x_0, \xi_0 \in \mathbb{R}$ ,

$$\left(\int_{\mathbb{R}} (x - x_0)^2 |\psi(x)|^2 dx\right) \left(\int_{\mathbb{R}} (\xi - \xi_0)^2 |\hat{\psi}(\xi)|^2 d\xi\right) \geq \frac{\|\psi\|_2^4}{16\pi^2}. \tag{6}$$

and equality is attained only when  $\psi$  is a Gaussian or modulated Gaussian.

## 2 Discretization and Boundary Handling

For a discrete one-dimensional signal  $(f_n)$ ,  $n \in \mathbb{Z}$ , Gaussian convolution is

$$u_n = \sum_{m=-\infty}^{\infty} g_m f_{n-m} \tag{7}$$

where  $(g_n)$  is a discretization of the Gaussian. We discuss several possibilities for how to define  $(g_n)$ .

### 2.1 Discretizations of the Gaussian

**Sampling** A simple discretization of the Gaussian is by sampling,

$$g_n^{\text{sampled}} = G_\sigma(n), \quad n \in \mathbb{Z}. \tag{8}$$

**Normalized samples** The samples are often normalized to ensure that they have unit sum,

$$g_n = \frac{1}{S} G_\sigma(n), \quad S = \sum_{n=-\infty}^{\infty} G_\sigma(n). \tag{9}$$

Both  $g_n^{\text{sampled}}$  and  $g_n$  are straightforward and preserve separability.

**Lemma 1.** *For any  $\sigma > 0$ , the sum  $S$  in (9) is strictly greater than 1.*

*Proof.* Using the Poisson summation formula<sup>2</sup> and  $\hat{G}_\sigma(\xi) = e^{-2\pi^2\sigma^2\xi^2}$ ,

$$S = \sum_{n=-\infty}^{\infty} G_\sigma(n) = \sum_{k=-\infty}^{\infty} \hat{G}_\sigma(k) = 1 + 2 \sum_{k=1}^{\infty} e^{-2\pi^2\sigma^2k^2} > 1. \quad \square$$

*Remark 1.* The sum  $S$  is typically very nearly but not exactly one, and the difference  $S - 1$  decreases rapidly as  $\sigma$  increases (see figure 2). For  $\sigma = 1.5$ , the difference is  $S - 1 \approx 1.0295 \times 10^{-19}$ . This difference is so small that  $S$  cannot be distinguished from 1 in double-precision floating-point representation.<sup>3</sup> Truncating the sum in the proof above yields that the difference is  $S - 1 \approx 2e^{-2\pi^2\sigma^2}$ , which due to the sum's exponential decay is a very accurate approximation for  $\sigma \geq 0.5$ .

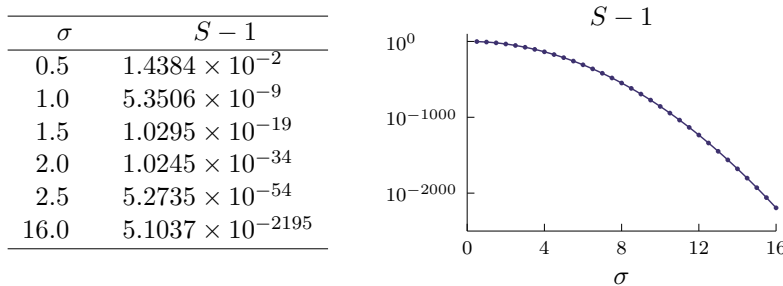


Figure 2: The sum  $S = \sum_n G_\sigma(n)$  is very nearly one for  $\sigma \geq 1.5$ . Computed using the mpmath library [26].

*Remark 2.* The sum  $S$  is closely related to Ramanujan's theta function, for which a few exact values are known [16, 21]. For instance  $S = \frac{1}{4}\pi^{1/4}(2 + 2^{3/4})/\Gamma(\frac{3}{4}) \approx 1.000006974684712$  if  $\sigma = \sqrt{2/\pi}$ .

<sup>2</sup>Poisson summation formula:  $\sum_{n=-\infty}^{\infty} f(n)e^{-i2\pi n\xi} = \sum_{n=-\infty}^{\infty} \hat{f}(\xi - n)$ .

<sup>3</sup>The least IEEE 754 double-precision number greater than one is  $1 + 2^{-52} \approx 1 + 2.22 \times 10^{-16}$ .

**Preserving the semigroup property** Lindeberg [7] notes that the semigroup property is not preserved with discretization (8) or (9) and proposes the discretization

$$g_n^{\text{Lindeberg}} = e^{-t} I_n(t), \quad t = \sigma^2, \tag{10}$$

which does preserve the semigroup property and the  $I_n$  are the modified Bessel functions. In analogy to Gaussian convolution arising from the continuous heat equation,  $u = g^{\text{Lindeberg}} * f$  satisfies

$$\partial_t u_n = \frac{1}{2}(u_{n-1} - 2u_n + u_{n+1}). \tag{11}$$

**Interpolation** Another reasonable discretization is to define Gaussian convolution using a linear interpolation  $F(x) = \sum_n \varphi(x - n) f_n$  of  $(f_n)$  as

$$\begin{aligned} u_n &= \int_{\mathbb{R}^d} G_\sigma(n - x') F(x') dx' \\ &= \sum_{n' \in \mathbb{Z}^d} g_{n-n'}^{\text{interp}} f_{n'}, \quad g_n^{\text{interp}} := (G_\sigma * \varphi)(n). \end{aligned} \tag{12}$$

If nearest neighbor interpolation is used, then  $\varphi(x) = 1$  for  $|x| < \frac{1}{2}$  and zero otherwise and

$$g_n^{\text{nearest}} = \int_{-1/2-n}^{+1/2-n} G_\sigma(x) dx = \frac{1}{2} \left[ \operatorname{erf} \left( \frac{n + 1/2}{\sqrt{2}\sigma} \right) - \operatorname{erf} \left( \frac{n - 1/2}{\sqrt{2}\sigma} \right) \right] \tag{13}$$

where erf is the error function. With Shannon–Whittaker interpolation,  $\varphi$  is the sinc function  $\sin(\pi x)/(\pi x)$  and

$$\begin{aligned} g_n^{\text{sinc}} &= (G_\sigma * \varphi)(n) = (\hat{G}_\sigma \cdot \hat{\varphi})(n), \quad \hat{G}_\sigma(\xi) = e^{-2\pi^2\sigma^2\xi^2}, \quad \hat{\varphi}(\xi) = \begin{cases} 1 & \text{if } |\xi| < \frac{1}{2} \\ 0 & \text{if } |\xi| > \frac{1}{2} \end{cases} \\ &= \int_{-1/2}^{1/2} e^{-2\pi^2\sigma^2\xi^2} \cos(2\pi\xi n) d\xi = G_\sigma(n) \operatorname{Re} \left\{ \operatorname{erf} \left( \frac{\pi\sigma}{\sqrt{2}} + i \frac{n}{\sqrt{2}\sigma} \right) \right\}. \end{aligned} \tag{14}$$

That is,  $g^{\text{sinc}}$  is a bandlimited version of  $G_\sigma$  sampled on the integers.

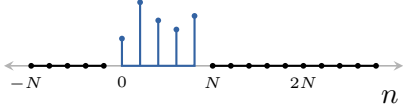
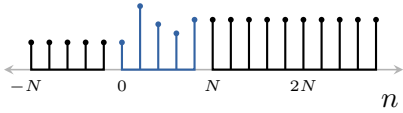
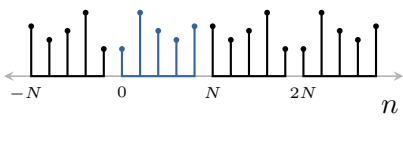
**Comparison** Table 1 shows the  $\ell^1$  differences between the sampled and normalized discretization defined by (9) and the other discretizations discussed above. For  $\sigma \geq 1$ , the difference among these discretizations is small and decreases as  $\sigma$  increases. Moreover, most of the Gaussian convolution algorithms discussed in this work are approximate, and discretization differences are negligible compared to approximation errors. From here on, we use discretization (9), except for DFT/DCT-based convolution where the sinc discretization  $g^{\text{sinc}}$  (14) is used.

$\sigma$	$\ g^{\text{sampled}} - g\ _1$	$\ g^{\text{Lindeberg}} - g\ _1$	$\ g^{\text{nearest}} - g\ _1$	$\ g^{\text{sinc}} - g\ _1$
1	$5.3506 \times 10^{-9}$	$1.5245 \times 10^{-1}$	$3.2996 \times 10^{-2}$	$7.1919 \times 10^{-3}$
2	$1.0245 \times 10^{-34}$	$3.2195 \times 10^{-2}$	$9.6135 \times 10^{-3}$	$2.6753 \times 10^{-9}$
4	$1.3771 \times 10^{-137}$	$7.3356 \times 10^{-3}$	$2.4908 \times 10^{-3}$	$5.1225 \times 10^{-35}$
8	$4.4952 \times 10^{-549}$	$1.8269 \times 10^{-3}$	$6.2826 \times 10^{-4}$	$6.8854 \times 10^{-138}$
16	$5.1037 \times 10^{-2195}$	$4.5615 \times 10^{-4}$	$1.5742 \times 10^{-4}$	$2.2476 \times 10^{-549}$

Table 1: Differences among Gaussian discretizations. Computed using the mpmath library [26].

## 2.2 Boundary Handling

Care is needed at the boundaries when filtering a signal of finite-length,  $f_0, \dots, f_{N-1}$ . For some  $m$ , the samples  $f_{m-n}$  in (7) lie outside of the domain of definition. We apply a *boundary extension* to extrapolate  $f$  to all  $n \in \mathbb{Z}$ . Several common choices of extension are

- zero-padding  $\tilde{f}_n = \begin{cases} f_n & \text{if } 0 \leq n < N, \\ 0 & \text{otherwise,} \end{cases}$  
- constant  $\tilde{f}_n = f_{\min(\max(n,0), N-1)}$ , 
- symmetric  $\tilde{f}_n = \begin{cases} f_n & \text{if } n = 0, \dots, N-1, \\ \tilde{f}_{-1-n} & \text{if } n < 0, \\ \tilde{f}_{2N-1-n} & \text{if } n \geq N. \end{cases}$  

We select the (half-sample) symmetric extension since this is usually a reasonable choice for images, our application of interest. Signal extension will be denoted by tilde  $\sim$  accent. For error analysis, we assume at least that  $\|\tilde{f}\|_\infty = \|f\|_\infty$ , which is true for the above extensions.

Several of the Gaussian approximation methods discussed in this survey are based on recursive (infinite impulse response) filters, which have the form

$$u_n = b_0 f_n + b_1 f_{n-1} + \dots + b_p f_{n-p} - a_1 u_{n-1} - a_2 u_{n-2} - \dots - a_q u_{n-q}, \quad \xrightarrow{z} \quad U(z) = \frac{b_0 + b_1 z^{-1} + \dots + b_p z^{-p}}{1 + a_1 z^{-1} + \dots + a_q z^{-q}} F(z). \quad (15)$$

A chicken-and-egg problem with recursive filters is how to initialize them on the boundaries. The first sample  $u_0$  depends recursively on outputs  $u_{-1}, \dots, u_{-q}$ . How to obtain the first output sample when it depends on other output samples?

To untangle this problem, we first note that a recursive filter can be expressed nonrecursively as convolution with its impulse response  $h$ . The impulse response of (15) can be obtained as shown in algorithm 1 (let  $b_n = 0$  for  $n > p$  so that the  $b_n$  term in the loop has effect only for  $n = 0, \dots, p$ ).

---

### Algorithm 1

---

```

 $h_n = 0$  for all  $n < 0$ 
for  $n = 0, 1, 2, \dots$  do
   $h_n = b_n - a_1 h_{n-1} - a_2 h_{n-2} - \dots - a_q h_{n-q}$ 

```

---

We assume that the  $h_n$  decay, which is true provided the filter is stable. To compute the first  $q$  output samples  $u_0, \dots, u_{q-1}$ , we write the filter as  $u = h * \tilde{f}$  and truncate the infinite sum,

$$u_m = \sum_{n=-m}^{\infty} h_{n+m} \tilde{f}_{-n} \approx \sum_{n=-m}^{k-1} h_{n+m} \tilde{f}_{-n}, \quad m = 0, \dots, q-1. \quad (16)$$

The approximation error due to this truncation is bounded,

$$\left| \sum_{n=k}^{\infty} h_{n+m} \tilde{f}_{-n} \right| \leq \|f\|_\infty \sum_{n=k}^{\infty} |h_n|, \quad m = 0, \dots, q-1. \quad (17)$$

Provided the filter is stable,  $\sum_{n=0}^{\infty} |h_n|$  is finite, which implies that  $k$  can be selected such that  $\sum_{n \geq k} |h_n|$  is arbitrarily small. Therefore,  $u_m$  is computable with any desired accuracy. Together with

algorithm 1, algorithm 2 computes  $u_0, \dots, u_{q-1}$  with error less than  $tol \|f\|_\infty$ .

---

**Algorithm 2**


---

**input** : signal  $f$ , filter coefficients  $(b_i)$  and  $(a_i)$ , accuracy  $tol$   
**output**:  $u_m = h * f$ ,  $m = 0, \dots, q - 1$ , with boundary handling  
Compute  $h_0, \dots, h_{q-1}$  with algorithm 1  
 $s \leftarrow \sum_{n=0}^{\infty} |h_n|$   
 $u_m \leftarrow \sum_{n=-m}^{-1} h_{n+m} \tilde{f}_{-n}$ ,  $m = 0, \dots, q - 1$   
**for**  $n = 0, 1, 2, \dots$  **do**  
     $u_m \leftarrow u_m + h_{n+m} \tilde{f}_{-n}$ ,  $m = 0, \dots, q - 1$   
     $s \leftarrow s - |h_n|$   
    **if**  $s \leq tol$  **then stop**  
     $h_{n+q} = b_{n+q} - a_1 h_{n+q-1} - a_2 h_{n+q-2} - \dots - a_q h_n$

---

The variable  $s$  tracks the absolute sum of the remaining filter samples,  $s = \sum_{m>n} |h_m|$ , and the loop stops once  $s \leq tol$ . To initialize  $s$ , the infinite sum  $\sum_{n=0}^{\infty} |h_n|$  must somehow be evaluated or accurately precomputed. If  $h$  is nonnegative, the infinite sum is computed exactly as

$$\sum_{n=0}^{\infty} h_n = H(1) = \frac{b_0 + \dots + b_p}{1 + a_1 + \dots + a_q}. \quad (18)$$

**Complexity** In the following sections, several algorithms use boundary initialization where the filter  $h$  is related to approximating the Gaussian. As the standard deviation  $\sigma$  increases,  $h$  generally becomes proportionally wider and more terms in algorithm 2 must be added to achieve the tolerance. This cost can be ignored when  $N$  is reasonably large compared to  $\sigma$ , since then the majority of the computation time is spent filtering interior samples.

## 2.3 Multidimensional Convolution

Thanks to separability, Gaussian convolution in multiple dimensions is a tensor product of one-dimensional convolutions, provided the domain is a Cartesian product. Convolution of images and volumes may be implemented using one-dimensional convolutions (see algorithm 3), which simplifies algorithm design and implementation. We focus hereafter on one dimension.

---

**Algorithm 3:** 2D Gaussian convolution decomposed into 1D convolutions.

---

**input** : image  $f$  defined on  $\{0, M - 1\} \times \{0, N - 1\}$   
**output**:  $u = G_\sigma * f$   
**for**  $n = 0, \dots, N - 1$  **do**  
     $\left[$  Compute  $(u_{m,n})_{m=0}^{M-1} \leftarrow G_\sigma * (f_{m,n})_{m=0}^{M-1}$  with any 1D method  
**for**  $m = 0, \dots, M - 1$  **do**  
     $\left[$  Compute  $(u_{m,n})_{n=0}^{N-1} \leftarrow G_\sigma * (u_{m,n})_{n=0}^{N-1}$  with any 1D method

---

### 3 Methods

#### 3.1 FIR Filtering

The simplest implementation of Gaussian convolution is approximation by a finite impulse response (FIR) filter, where the Gaussian is truncated to  $|n| \leq r$ ,

$$H(z) = \frac{1}{s(r)} \sum_{n=-r}^r G_\sigma(n) z^{-n}, \quad s(r) = \sum_{n=-r}^r G_\sigma(n). \quad (19)$$

The size of the radius  $r$  determines the tradeoff between accuracy and speed.

**Theorem 1.** *Define the truncated filter  $g_n^{\text{trunc}} = G_\sigma(n)/s(r)$  for  $|n| \leq r$  and zero otherwise, then the error made in approximating  $(g * \tilde{f})$  by  $(g^{\text{trunc}} * \tilde{f})$  is bounded as*

$$\|g * \tilde{f} - g^{\text{trunc}} * \tilde{f}\|_\infty \leq 2 \operatorname{erfc}\left(\frac{r}{\sqrt{2}\sigma^2}\right) \|f\|_\infty, \quad (20)$$

where  $\operatorname{erfc}(t) := 1 - \operatorname{erf}(t)$  is the complementary error function.

*Proof.* The  $\ell^1$  distance between  $g$  and  $g^{\text{trunc}}$  is

$$\|g - g^{\text{trunc}}\|_1 = \sum_{|n| \leq r} \left(\frac{1}{s(r)} G_\sigma(n) - \frac{1}{s(\infty)} G_\sigma(n)\right) + \sum_{|n| > r} \frac{1}{s(\infty)} G_\sigma(n). \quad (21)$$

The first sum can be rewritten by factoring and using  $s(\infty) - s(r) = \sum_{|n| > r} G_\sigma(n)$ ,

$$\begin{aligned} \sum_{|n| \leq r} \left(\frac{1}{s(r)} G_\sigma(n) - \frac{1}{s(\infty)} G_\sigma(n)\right) &= \left(\frac{1}{s(r)} - \frac{1}{s(\infty)}\right) s(r) \\ &= \frac{s(\infty) - s(r)}{s(\infty)} \\ &= \frac{1}{s(\infty)} \sum_{|n| > r} G_\sigma(n) \end{aligned} \quad (22)$$

Combining (21) and (22), we obtain the bound

$$\begin{aligned} \|g - g^{\text{trunc}}\|_1 &= \frac{2}{s(\infty)} \sum_{|n| > r} G_\sigma(n) \\ &= \frac{4}{s(\infty)} \sum_{n=r+1}^{\infty} G_\sigma(n) \\ &\leq \frac{4}{s(\infty)} \int_r^{\infty} \frac{e^{-t^2/(2\sigma^2)}}{\sqrt{2\pi}\sigma^2} dt \\ &= \frac{2}{s(\infty)} \operatorname{erfc}\left(\frac{r}{\sqrt{2}\sigma^2}\right), \end{aligned} \quad (23)$$

where  $\frac{2}{s(\infty)} < 2$  by lemma 1. The conclusion (20) then follows from Young's inequality.  $\square$

The practical value of theorem 1 is that it tells how large the radius  $r$  must be for a desired level of accuracy. Selecting  $r$  as

$$r = \lceil \sqrt{2} \operatorname{erfc}^{-1}(\operatorname{tol}/2) \sigma \rceil \quad (24)$$

ensures error less than  $\operatorname{tol} \|f\|_\infty$ , where for instance  $\sqrt{2} \operatorname{erfc}^{-1}(\operatorname{tol}/2) \approx 3.4808$  if  $\operatorname{tol} = 10^{-3}$ .

Boundary handling is relatively straightforward for FIR-based convolution. In algorithm 4,  $\tilde{f}$  denotes the symmetric extension of  $f$  as defined in section 2.2.

---

**Algorithm 4:** Gaussian convolution with truncated FIR approximation

---

**input** : signal  $f = (f_0, f_1, \dots, f_{N-1})$ , standard deviation  $\sigma$ , accuracy  $tol$

**output:**  $u \approx g_\sigma * f$

$r = \lceil \sqrt{2} \operatorname{erfc}^{-1}(tol/2)\sigma \rceil$

$g_m^{\text{trunc}} = \frac{1}{s(r)} \exp(-\frac{m^2}{2\sigma^2})$ ,  $m = -r, \dots, r$

**for**  $n = 0, 1, \dots, N - 1$  **do**

$u_n \leftarrow g_0^{\text{trunc}} f_n$

**for**  $m = 1, \dots, r$  **do**

$u_n \leftarrow u_n + g_m^{\text{trunc}}(\tilde{f}_{n+m} + \tilde{f}_{n-m})$

---

**Complexity** For a length- $N$  input, the computational complexity of FIR Gaussian approximation is  $\mathcal{O}(Nr)$ . If  $tol$  is fixed,  $r$  is proportional to  $\sigma$  and the complexity is  $\mathcal{O}(N\sigma)$ . FIR filtering is reasonable for small  $\sigma$  but costly for large values of  $\sigma$ .

## 3.2 DFT/DCT Convolution

It is well known that while direct convolution of two length- $N$  signals costs  $\mathcal{O}(N^2)$  operations, convolution via the discrete Fourier transform (DFT) costs only  $\mathcal{O}(N \log N)$  operations, leveraging the convolution-multiplication property and fast Fourier transform (FFT) algorithms [8].

### 3.2.1 DFT-based Convolution

Define periodic (cyclic) convolution of length- $N$  signals  $h$  and  $f$ ,

$$u_m = (h \overset{\text{per}}{*} f)_m := \sum_{n=0}^{N-1} h_n f_{(m-n) \bmod N}, \quad m = 0, 1, \dots, N - 1. \quad (25)$$

By the convolution-multiplication property of the DFT, (25) can be computed as

$$(h \overset{\text{per}}{*} f)_m = \mathcal{F}^{-1}(\mathcal{F}(h) \cdot \mathcal{F}(f)), \quad \mathcal{F}(f)_k := \sum_{n=0}^{N-1} f_n e^{-i2\pi nk/N}, \quad (26)$$

where  $\cdot$  denote elementwise multiplication and  $\mathcal{F}$  is the DFT. Unfortunately, the convolution (25) corresponds to periodic boundary extension, which is generally undesirable for images. A solution is to pad the input signal as  $f^{\text{pad}} = (f_0, f_1, \dots, f_{N-1}, f_{N-1}, \dots, f_1, f_0)$ , then its periodization corresponds to half-sample symmetric boundary extension,  $h * \tilde{f} = h \overset{\text{per}}{*} f^{\text{pad}}$ .

It is convenient to use the sinc-interpolated Gaussian discretization  $g^{\text{sinc}}$  since its DFT can be computed in closed form. Recall that

$$g_n^{\text{sinc}} = (G_\sigma * \varphi)(n) = (\hat{G}_\sigma \cdot \hat{\varphi})(n), \quad \hat{G}_\sigma(\xi) = e^{-2\pi^2 \sigma^2 \xi^2}, \quad \hat{\varphi}(\xi) = \begin{cases} 1 & \text{if } |\xi| < \frac{1}{2}, \\ 0 & \text{if } |\xi| > \frac{1}{2}. \end{cases}$$

Since the Gaussian has infinite support, we wrap it by summing periodic translates

$$h_n = \sum_{m=-\infty}^{\infty} g_{n-2Nm}^{\text{sinc}}, \quad n = 0, 1, \dots, 2N - 1, \quad (27)$$



such that  $h *^{\text{per}} f^{\text{pad}}$  is equal to  $g^{\text{sinc}} * \tilde{f}$ . We then obtain the length- $2N$  DFT

$$\begin{aligned} \mathcal{F}(h)_k &:= \sum_{n=0}^{2N-1} h_n e^{-i2\pi nk/(2N)} = \sum_{n=-\infty}^{\infty} (G_\sigma * \varphi)(n) e^{-i2\pi nk/(2N)} \\ &= \sum_{n=-\infty}^{\infty} (\hat{G}_\sigma \cdot \hat{\varphi})\left(\frac{k}{2N} - n\right) = \begin{cases} \hat{G}_\sigma\left(\frac{k}{2N}\right) & \text{if } 0 \leq k \leq N, \\ \hat{G}_\sigma\left(\frac{k}{2N} - 1\right) & \text{if } N \leq k < 2N, \end{cases} \end{aligned} \quad (28)$$

where the third equality follows from the Poisson summation formula and we extrapolate  $\mathcal{F}(g^{\text{sinc}})_N = \lim_{\xi \uparrow \frac{1}{2}} (\hat{G}_\sigma \cdot \hat{\varphi})(\xi) = \lim_{\xi \downarrow \frac{1}{2}} (\hat{G}_\sigma \cdot \hat{\varphi})(\xi - 1)$  to overcome the discontinuity in  $\hat{\varphi}$ .

The DFT-based Gaussian convolution is summarized in algorithm 5.

---

**Algorithm 5:** DFT-based Gaussian convolution

---

**input** : signal  $f = (f_0, f_1, \dots, f_{N-1})$ , standard deviation  $\sigma$

**output**:  $u = g_\sigma^{\text{sinc}} * f$  with symmetric boundary handling

$f^{\text{pad}} = (f_0, f_1, \dots, f_{N-1}, f_{N-1}, \dots, f_1, f_0)$

$F = \mathcal{F}(f^{\text{pad}})$

$U_k = \begin{cases} F_k \exp(-2\pi^2 \sigma^2 (\frac{k}{2N})^2), & k = 0, \dots, N \\ F_k \exp(-2\pi^2 \sigma^2 (\frac{k}{2N} - 1)^2), & k = N + 1, \dots, 2N - 1 \end{cases}$

$u^{\text{pad}} = \mathcal{F}^{-1}(U)$

$u = (u_0^{\text{pad}}, u_1^{\text{pad}}, \dots, u_{N-1}^{\text{pad}})$

---

### 3.2.2 DCT-based Convolution

Since the Gaussian is an even function, the discrete cosine transform (DCT) domain may be used instead of the DFT (algorithm 5) for greater computational efficiency. The data does not need to be padded in this case because symmetric boundaries are implied by the transforms, reducing the computational and memory costs. Martucci [11] showed that convolution with half-sample symmetric boundary handling can be implemented through DCT transforms as

$$h * \tilde{f} = \mathcal{C}_{2e}^{-1}(\mathcal{C}_{1e}(h) \cdot \mathcal{C}_{2e}(f)), \quad (29)$$

where  $\mathcal{C}_{1e}$  and  $\mathcal{C}_{2e}$  are the DCT-I and DCT-II transforms of the same period lengths,

$$\mathcal{C}_{1e}(h)_k = h_0 + (-1)^k h_N + 2 \sum_{n=1}^{N-1} h_n \cos(\pi nk/N), \quad k = 0, \dots, N, \quad (30)$$

$$\mathcal{C}_{2e}(f)_k = 2 \sum_{n=0}^{N-1} f_n \cos\left(\pi\left(n + \frac{1}{2}\right)k/N\right), \quad k = 0, \dots, N - 1. \quad (31)$$

Since (27) has the symmetry  $h_{2N-n} = h_n$ , its DCT-I transform reduces to  $\mathcal{C}_{1e}(h) = \mathcal{F}(h)$  (28).

---

**Algorithm 6:** DCT-based Gaussian convolution

---

**input** : signal  $f = (f_0, f_1, \dots, f_{N-1})$ , standard deviation  $\sigma$

**output**:  $u = g_\sigma^{\text{sinc}} * f$  with symmetric boundary handling

$$F_k = \mathcal{C}_{2e}(f)_k := 2 \sum_{n=0}^{N-1} f_n \cos\left(\pi\left(n + \frac{1}{2}\right)k/N\right), \quad k = 0, \dots, N-1$$

$$U_k = F_k \exp\left(-2\pi^2\sigma^2\left(\frac{k}{2N}\right)^2\right), \quad k = 0, \dots, N-1$$

$$u_n = \mathcal{C}_{2e}^{-1}(U)_n := \frac{1}{2N} \left( U_0 + 2 \sum_{k=1}^{N-1} U_k \cos\left(\pi\left(n + \frac{1}{2}\right)k/N\right) \right), \quad n = 0, \dots, N-1$$

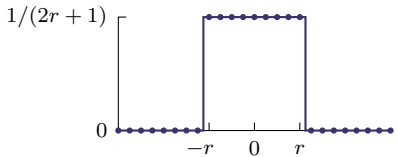

---

Compared to algorithm 5, the advantage of DCT-based convolution is that no padding is needed. The computation can be performed in-place by reusing one length- $N$  memory buffer for  $f, F, U, u$ .

**Complexity** The DCT-II transform of size  $N$  can be evaluated in  $\mathcal{O}(N \log N)$  operations<sup>4</sup> with fast cosine transform algorithms or with FFT algorithms through an equivalent DFT.

### 3.3 Box

The box (or boxcar) filter is a recursive filter with a box-shaped impulse response,

$$H(z) = \frac{1}{2r+1} \frac{z^r - z^{-r-1}}{1 - z^{-1}}. \quad (32)$$


The virtue of the box filter is that its computational cost is independent of its radius  $r$ . Gaussian convolution can be approximated by  $K$  passes of box filtering where  $K$  is usually 3, 4, or 5 (see algorithm 7). Wells [3] suggests to select  $r$  according to  $\sigma^2 = \frac{1}{12}K((2r+1)^2 - 1)$ .

---

**Algorithm 7:** Box filtering approximation of Gaussian convolution

---

**input** : signal  $f = (f_0, f_1, \dots, f_{N-1})$ , standard deviation  $\sigma$ , passes  $K$

**output**:  $u^K \approx g_\sigma * f$  with symmetric boundary handling

$$r = \lfloor \frac{1}{2} \sqrt{\frac{12}{K} \sigma^2 + 1} \rfloor$$

$$u^0 \leftarrow f / (2r+1)^K$$

**for**  $k = 0, \dots, K-1$  **do**

$$\left[ \begin{array}{l} s \leftarrow \sum_{n=-r}^r \tilde{u}_n^k \\ u_0^{k+1} \leftarrow s \\ \quad \mathbf{for} \ n = 1, \dots, N-1 \ \mathbf{do} \\ \quad \left[ \begin{array}{l} s \leftarrow s + \tilde{u}_{n+r}^k - \tilde{u}_{n-r-1}^k \\ u_n^{k+1} \leftarrow s \end{array} \right. \end{array} \right.$$


---

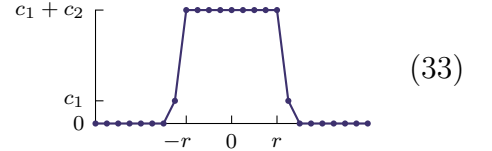
**Complexity** For a length- $N$  input and  $K$  passes, the complexity of box filtering is  $\mathcal{O}(NK + r)$ . The cost per interior pixel is  $K$  additions,  $K$  subtractions, and one multiplication.

<sup>4</sup> $N$  need not be an integer power of two, there exist  $\mathcal{O}(N \log N)$  algorithms for any  $N \geq 1$  [8].

**Improvements** Since  $r$  is integer-valued, a limitation of box filtering is that the set of possible approximated standard deviations is quantized. With  $K = 3$ , the possible standard deviations by Wells' formula are  $\sigma \in \{2, 6, 12, 20, 30, \dots\}$ . There are several methods to overcome this limitation:

1. **Semigroup property.** For a desired standard deviation of  $\sigma$ , box filtering is first performed with  $\sigma_1 \leq \sigma$ , then the result is convolved with  $G_{\sigma_2}$  (using any method) where  $\sigma_2 = \sqrt{\sigma^2 - \sigma_1^2}$ .
2. **Extended box filter** (algorithm 8). Gwosdek, Grewenig, Bruhn, and Weickert [30] extend the box filter to allow a fractional radius,

$$u_n = u_{n-1} + c_1(f_{n+r+1} - f_{n-r-2}) + c_2(f_{n+r} - f_{n-r-1}).$$



$$(33)$$

Gaussian convolution is approximated by  $K$  passes of extended box filtering with the parameters

$$r = \left\lfloor \frac{1}{2} \sqrt{\frac{12\sigma^2}{K} + 1} - \frac{1}{2} \right\rfloor, \quad c_1 = \frac{\alpha}{2\alpha + 2r + 1}, \quad c_2 = \frac{1 - \alpha}{2\alpha + 2r + 1}, \quad \text{where } \alpha = (2r + 1) \frac{r(r + 1) - 3\frac{\sigma^2}{K}}{6(\frac{\sigma^2}{K} - (r + 1)^2)}. \quad (34)$$

---

**Algorithm 8:** Extended box filtering approximation of Gaussian convolution

---

**input** : signal  $f = (f_0, f_1, \dots, f_{N-1})$ , standard deviation  $\sigma$ , passes  $K$

**output:**  $u^K \approx g_\sigma * f$  with symmetric boundary handling

Set  $r, c_1, c_2$  according to (34)

$u^0 \leftarrow f$

**for**  $k = 0, \dots, K - 1$  **do**

$s \leftarrow (c_1 + c_2) \sum_{n=-r}^r \tilde{u}_n^k + c_1(\tilde{u}_{r+1}^k + \tilde{u}_{-r-1}^k)$

$u_0^{k+1} \leftarrow s$

**for**  $n = 1, \dots, N - 1$  **do**

$s \leftarrow s + c_1(\tilde{u}_{n+r+1}^k - \tilde{u}_{n-r-2}^k) + c_2(\tilde{u}_{n+r}^k - \tilde{u}_{n-r-1}^k)$

$u_n^{k+1} \leftarrow s$

3. **Stacked integral images (SII)** [27, 29]. A weighted sum of box filters is computed,

$$u_n = \sum_{k=1}^K w_k (s_{n+r_k} - s_{n-r_k-1}), \quad s_n = \sum_{m=-pad}^n \tilde{f}_m, \quad pad = 1 + \max_k r_k. \quad (35)$$

The  $k$ th term of the sum effectively computes a box filter of radius  $r_k$  since  $s_{n+r_k} - s_{n-r_k-1} = \sum_{m=n-r_k}^{n+r_k} \tilde{f}_m$ . Extended box filtering is a special case with  $K = 2$ . The radii  $r_1, \dots, r_K$  and weights  $w_1, \dots, w_K$  are selected to optimize the approximation of the Gaussian. Table 2 lists effective values suggested by Elboher and Werman [29] for standard deviation  $\sigma_0 = 100/\pi$ . These parameters can be rescaled for other  $\sigma$  as described in algorithm 9.

$K$	$r_k^0$	$w_k^0$
3	76, 46, 23	0.1618, 0.5502, 0.9495
4	83, 56, 37, 19	0.0976, 0.3376, 0.6700, 0.9649
5	85, 61, 44, 30, 16	0.0739, 0.2534, 0.5031, 0.7596, 0.9738

Table 2: Radii and weights that optimize the approximation of the Gaussian for SII [29]

---

**Algorithm 9:** SII approximation of Gaussian convolution

---

**input** : signal  $f = (f_0, f_1, \dots, f_{N-1})$ , standard deviation  $\sigma$ , number of boxes  $K$   
**output**:  $u \approx g_\sigma * f$  with symmetric boundary handling  
Read  $r_1^0, \dots, r_K^0$  and  $w_1^0, \dots, w_K^0$  from table 2  
 $r_k = \lceil \frac{\sigma}{\sigma_0} r_k^0 \rceil$ ,  $w_k = w_k^0 / \sum_{j=1}^K w_j^0 (2r_j + 1)$ ,  $k = 1, \dots, K$   
 $pad = 1 + \max_k r_k$   
 $s_{-pad-1} = 0$   
**for**  $n = -pad, \dots, N + pad - 1$  **do**  
     $s_n = s_{n-1} + \tilde{f}_n$   
**for**  $n = 0, \dots, N - 1$  **do**  
     $u_n = \sum_{k=1}^K w_k (s_{n+r_k} - s_{n-r_k-1})$

---

### 3.4 Deriche

Deriche [10] constructs approximations for the right half of the Gaussian  $n \geq 0$  of the form

$$G_\sigma(n) \approx h_n^{(K)} := \frac{1}{\sqrt{2\pi\sigma^2}} \sum_{k=1}^K \alpha_k e^{-n\lambda_k/\sigma} \xrightarrow{z} H^{(K)}(z) = \frac{1}{\sqrt{2\pi\sigma^2}} \sum_{k=1}^K \frac{\alpha_k}{1 - e^{-\lambda_k/\sigma} z^{-1}}. \quad (36)$$

$H^{(K)}$  is an order- $K$  recursive filter, allowing efficient implementation. The left half of the Gaussian  $n < 0$  is obtained using a similar anticausal filter. These left- and right-half filters are added (a parallel filter combination) to construct an approximately Gaussian impulse response (algorithm 10).

Deriche optimized the  $\alpha_k$  and  $\lambda_k$  parameters to find the  $\ell^2$ -best fit to the Gaussian over the domain  $n = 0, \dots, 1000$  with  $\sigma = 100$ . This optimization is numerically difficult, but Deriche succeeded with routine E04FCF of the NAG library [31]. We fortunately do not need to repeat this optimization and only need the resulting values, which are

$$\begin{aligned}
K = 2: \quad & \alpha_1 = 0.48145 + i0.971, \\
& \lambda_1 = 1.26 + i0.8448, \\
K = 3: \quad & \alpha_1 = -0.44645 + i0.5105, \quad \alpha_3 = 1.898, \\
& \lambda_1 = 1.512 + i1.475, \quad \lambda_3 = 1.556, \\
K = 4: \quad & \alpha_1 = 0.84 + i1.8675, \quad \alpha_3 = -0.34015 - i0.1299, \\
& \lambda_1 = 1.783 + i0.6318, \quad \lambda_3 = 1.723 + i1.997,
\end{aligned} \quad (37)$$

where  $\alpha_{2k} = \alpha_{2k-1}^*$ ,  $\lambda_{2k} = \lambda_{2k-1}^*$ . The recursive filter  $H^{(K)}(z)$  can be expressed in the form of (15) by algebraically combining the sum into a single fraction,

$$H^{(K)}(z) = \frac{1}{\sqrt{2\pi\sigma^2}} \sum_{k=1}^K \frac{\alpha_k}{1 - e^{-\lambda_k/\sigma} z^{-1}} = \frac{\sum_{k=0}^{K-1} b_k^+ z^{-k}}{1 + \sum_{k=1}^K a_k z^{-k}}. \quad (38)$$

For example for  $K = 3$ ,

$$\begin{aligned} b_0^+ &= \frac{1}{\sqrt{2\pi\sigma^2}}(\alpha_1 + \alpha_2 + \alpha_3), & a_1 &= \beta_1 + \beta_2 + \beta_3, \\ b_1^+ &= \frac{1}{\sqrt{2\pi\sigma^2}}(\alpha_1(\beta_2 + \beta_3) + \alpha_2(\beta_1 + \beta_3) + \alpha_3(\beta_1 + \beta_2)), & a_2 &= \beta_1\beta_2 + \beta_1\beta_3 + \beta_2\beta_3, \\ b_2^+ &= \frac{1}{\sqrt{2\pi\sigma^2}}(\alpha_1\beta_2\beta_3 + \alpha_2\beta_1\beta_3 + \alpha_3\beta_1\beta_2), & a_3 &= \beta_1\beta_2\beta_3, \end{aligned} \quad (39)$$

where  $\beta_k = -e^{-\lambda_k/\sigma}$ . The conjugacy of the pairs  $\alpha_2 = \alpha_1^*$ ,  $\lambda_2 = \lambda_1^*$  allows to express the coefficients in purely real arithmetic [10].

The left (anticausal) half of the Gaussian is obtained by spatial reversal and subtracting the sample at  $n = 0$  so that it is not produced twice,

$$\frac{\sum_{k=1}^K b_k^- z^k}{1 + \sum_{k=1}^K a_k z^k} = H^{(K)}(z^{-1}) - h_0^{(K)} = \frac{\sum_{k=1}^K (b_k^+ - a_k b_0^+) z^k}{1 + \sum_{k=1}^K a_k z^k}. \quad (40)$$

---

**Algorithm 10:** Deriche approximation of Gaussian convolution [10]

---

**input** : signal  $f = (f_0, f_1, \dots, f_{N-1})$ , standard deviation  $\sigma$ , order  $K$ , accuracy  $tol$

**output:**  $u \approx g_\sigma * f$  with symmetric boundary handling

Use (37) and (38) to determine  $b_0^+, \dots, b_{K-1}^+$  and  $a_1, \dots, a_K$

$b_k^- = b_k^+ - a_k b_0^+$  for  $k = 1, \dots, K$

Compute  $q_0^+, \dots, q_{K-1}^+$  with accuracy  $tol$  using algorithm 2 *Causal filter*

**for**  $n = K, \dots, N - 1$  **do**

$q_n^+ \leftarrow \sum_{k=0}^{K-1} b_k^+ f_{n-k} - \sum_{k=1}^K a_k q_{n-k}^+$

Compute  $q_{N-1}^-, \dots, q_{N-K}^-$  with accuracy  $tol$  using algorithm 2 *Anticausal filter*

**for**  $n = N - K - 1, \dots, 1, 0$  **do**

$q_n^- \leftarrow \sum_{k=0}^{K-1} b_k^- f_{n+k} - \sum_{k=1}^K a_k q_{n+k}^-$

$u = q^- + q^+$

---

**Complexity** Ignoring boundary initialization, the computational complexity is  $\mathcal{O}(N)$ . The computational cost is  $4K$  multiplications and  $(4K - 1)$  additions per interior pixel.

### 3.5 Alvarez–Mazorra

Alvarez and Mazorra [12] derive a recursive filter from a finite difference discretization of the heat equation. Let  $K$  be the number of passes,  $\lambda = q^2/(2K)$ , and  $\nu = \frac{1}{2\lambda}(1 + 2\lambda - \sqrt{1 + 4\lambda})$ , then the filter is

$$H(z) = (\nu/\lambda)^K \left( \frac{1}{1 - \nu z^{-1}} \frac{1}{1 - \nu z} \right)^K. \quad (41)$$

The filter is a cascade of the causal filter  $u'_n = f_n + \nu u'_{n-1}$  and anticausal filter  $u''_n = u'_n + \nu u''_{n+1}$ , both applied  $K$  times, and a scale factor  $(\nu/\lambda)^K$ .

In the original work [12], Alvarez and Mazorra set  $q = \sigma$ . While the method with  $q = \sigma$  converges to Gaussian filtering with parameter  $\sigma$  as  $K \rightarrow \infty$ , this choice tends to undersmooth for small  $K$ . Here we introduce an adjustment to compensate for this effect (see algorithm 11). For each value of  $q = 1, 1.5, \dots, 500$  and  $K = 2, 3, \dots, 20$ , we compute the impulse responses  $h_{q,K}$  of (41) and estimate the best-fitting Gaussian  $\sigma^* = \arg \min_\sigma \|h_{q,K} - G_\sigma\|_1$ . Regression over  $(\sigma^*, K)$  leads to an estimate for the value of  $q$  that best approximates a desired Gaussian standard deviation  $\sigma$ :

$$q = \sigma \left( 1 + \frac{0.3165K + 0.5695}{(K + 0.7818)^2} \right). \quad (42)$$

The experiments section compares Alvarez–Mazorra with (42) to the original method with  $q = \sigma$ .

**Boundary handling** Alvarez and Mazorra’s discussion is on the infinite grid and does not include boundary handling. Here we develop its use with half-sample symmetric boundaries. We apply the approximation discussed in section 2.2 to compute the left endpoint of the causal recursion  $u'_n = f_n + \nu u'_{n-1}$ ,

$$u'_0 \approx \sum_{m=0}^{M-1} \nu^m \tilde{f}_{-m}, \quad (43)$$

where from (17) it follows that the  $\ell^\infty$  error is less than  $\|f\|_\infty \nu^M / (1 - \nu)$ . For the right endpoint of the anticausal recursion  $u''_n = u'_n + \nu u''_{n+1}$ , we note that  $\tilde{f}$  is half-sample symmetric and the filter  $(1 - \nu z^{-1})^{-1} (1 - \nu z)^{-1}$  is whole-sample symmetric, therefore their convolution  $u''$  is also half-sample symmetric [11]. This symmetry implies  $u''_{N-1} = u''_N$ , so together with  $u''_{N-1} = u'_{N-1} + \nu u''_N$ , it allows to solve for  $u''_{N-1}$  as

$$u''_{N-1} = \frac{u'_{N-1}}{(1 - \nu)} = \frac{u_{N-1} + \nu u'_{N-2}}{(1 - \nu)}. \quad (44)$$

---

**Algorithm 11:** Alvarez–Mazorra approximation of Gaussian convolution [12]

---

**input** : signal  $f = (f_0, f_1, \dots, f_{N-1})$ , standard deviation  $\sigma$ , passes  $K$ , accuracy  $tol$

**output:**  $u \approx g_\sigma * f$  with symmetric boundary handling

Compute  $q$  with (42)

$\lambda = q^2 / (2K)$ ,  $\nu = \frac{1+2\lambda-\sqrt{1+4\lambda}}{2\lambda}$ ,  $M = \lceil \log_\nu((1 - \nu)tol) \rceil$

$u \leftarrow (\nu/\lambda)^K f$

**for**  $k = 0, 1, \dots, K - 1$  **do**

$u_0 \leftarrow \sum_{m=0}^{M-1} \nu^m \tilde{u}_{-m}$  *Causal filter*  $u'_n \leftarrow u_n + \nu u'_{n-1}$

**for**  $n = 1, 2, \dots, N - 2$  **do**

$u_n \leftarrow u_n + \nu u_{n-1}$

$u_{N-1} \leftarrow (u_{N-1} + \nu u_{N-2}) / (1 - \nu)$  *Anticausal filter*  $u''_n \leftarrow u'_n + \nu u''_{n+1}$

**for**  $n = N - 2, \dots, 1, 0$  **do**

$u_n \leftarrow u_n + \nu u_{n+1}$

---

**Complexity** Ignoring boundary initialization, Alvarez–Mazorra filtering has complexity  $\mathcal{O}(NK)$ . For each interior pixel, the cost is  $(2K + 1)$  multiplications and  $2K$  additions.

### 3.6 Vliet–Young–Verbeek

Another recursive approximation of Gaussian convolution, proposed by Young and van Vliet [13] and refined by van Vliet, Young, and Verbeek [17], has the form

$$H(z) = \prod_{k=1}^K \frac{d_k - 1}{d_k - z^{-1}} \cdot \prod_{k=1}^K \frac{d_k - 1}{d_k - z} = G(z)G(z^{-1}), \quad G(z) = \frac{b_0}{1 + a_1 z^{-1} + \dots + a_K z^{-K}}, \quad (45)$$

which is the cascade of causal filter  $G(z)$  and anticausal filter  $G(z^{-1})$ . The denominator  $\prod (d_k - z^{-1})$  can be algebraically expanded to obtain  $b_0, a_1, \dots, a_K$ . Van Vliet et al. select the poles  $d_k$  to minimize

the  $L^2$  distance from the Gaussian with standard deviation  $\sigma_0 = 2$ ,

$$\begin{aligned} K = 3: & \quad d_1 = 1.41650 + i1.00829, \quad d_3 = 1.86543 \\ K = 4: & \quad d_1 = 1.13228 + i1.28114, \quad d_3 = 1.78534 + i0.46763 \\ K = 5: & \quad d_1 = 0.86430 + i1.45389, \quad d_3 = 1.61433 + i0.83134, \quad d_5 = 1.87504 \end{aligned} \quad (46)$$

where  $d_{2k} = d_{2k-1}^*$ . For a general standard deviation  $\sigma$ , the poles are scaled according to  $d_k^{1/q} \mapsto d_k$ , where  $q$  is selected such that the filter variance equals  $\sigma^2$ ,

$$\text{var}(h) = \sum_{k=1}^K \frac{2d_k^{1/q}}{(d_k^{1/q} - 1)^2} = \sigma^2. \quad (47)$$

The value of  $q$  is accurately estimated by initializing  $q = \sigma/\sigma_0$  and performing several iterations of Newton's method.

**Boundary handling** Van Vliet et al. [17] do not discuss boundary handling. Triggs and Sdika [24] developed constant extension boundary handling when  $K = 3$ . Here we develop handling for half-sample symmetric extension: the left endpoints  $q_0, \dots, q_{K-1}$  can be computed with algorithm 2. Similarly to the boundary handling for Alvarez–Mazorra, the right endpoints  $u_{N-K}, \dots, u_{N-1}$  are obtained by solving the linear system

$$u_{N-m} = b_0 q_{N-m} - \sum_{k=1}^K a_k \tilde{u}_{N-m+k}, \quad m = 1, \dots, K. \quad (48)$$

For example, with  $K = 3$  and half-sample symmetric boundaries, the solution is

$$\begin{pmatrix} u_{N-3} \\ u_{N-2} \\ u_{N-1} \end{pmatrix} = b_0 \begin{pmatrix} 1 & a_1 & a_2 + a_3 \\ 0 & 1 + a_3 & a_1 + a_2 \\ a_3 & a_2 & 1 + a_1 \end{pmatrix}^{-1} \begin{pmatrix} q_{N-3} \\ q_{N-2} \\ q_{N-1} \end{pmatrix}. \quad (49)$$

This leads to an efficient algorithm (algorithm 12) with one forward pass and one backward pass through the data. Computation may be performed in-place by substituting all occurrences of  $q$  and  $u$  with  $f$ .

---

**Algorithm 12:** Vliet–Young–Verbeek approximation of Gaussian convolution [17]

---

**input** : signal  $f = (f_0, f_1, \dots, f_{N-1})$ , standard deviation  $\sigma$ , filter order  $K$ , accuracy  $tol$

**output:**  $u \approx g_\sigma * f$  with symmetric boundary handling

Obtain  $(d_k)$  from equations (46) and (47)

Compute  $q_0, \dots, q_{K-1}$  with accuracy  $tol$  using algorithm 2

*Causal filter*

**for**  $n = K, \dots, N - 1$  **do**

$$\lfloor q_n \leftarrow c_0 f_n - \sum_{k=1}^K a_k q_{n-k}$$

Compute  $u_{N-K}, \dots, u_{N-1}$  by solving (48)

*Anticausal filter*

**for**  $n = N - K - 1, \dots, 1, 0$  **do**

$$\lfloor u_n \leftarrow b_0 q_n - \sum_{k=1}^K a_k u_{n+k}$$


---

**Complexity** Ignoring boundary initialization, the complexity is  $\mathcal{O}(N)$ . The cost per interior pixel is  $2(K + 1)$  multiplications and  $3K$  additions.

## 4 Experiments

### 4.1 Impulse Responses

In the following experiments, we compute the response of the methods to the unit impulse when  $\sigma = 5$ , which should be approximately Gaussian.

**Box filtering** The impulse response of  $K$  passes of box filtering (algorithm 7) is the  $(K - 1)$ th B-spline. The response is more Gaussian-like as  $K$  increases (figure 3). Extended box filtering (algorithm 8) has similar impulse responses, the key difference being that the box radius may be fractional.

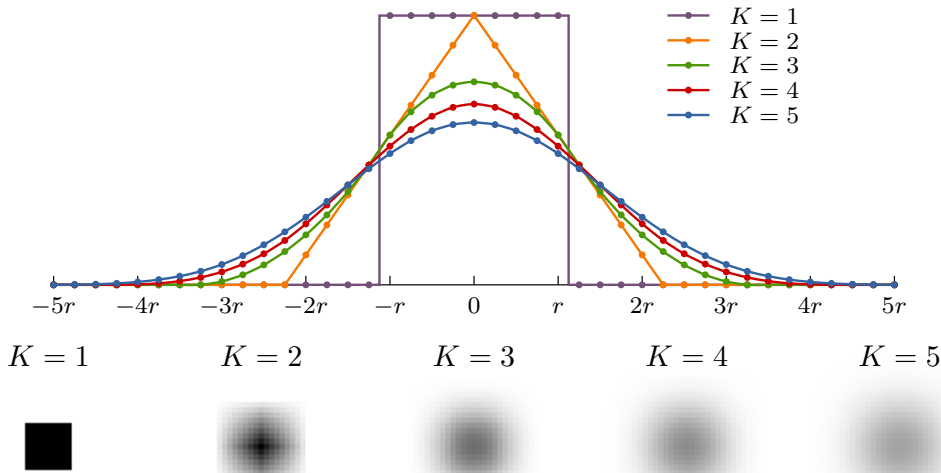


Figure 3: Box filter impulse responses with  $K$  passes in one and two dimensions.

**SII** Stacked integral images (SII) yields a piecewise constant impulse response (figure 4).

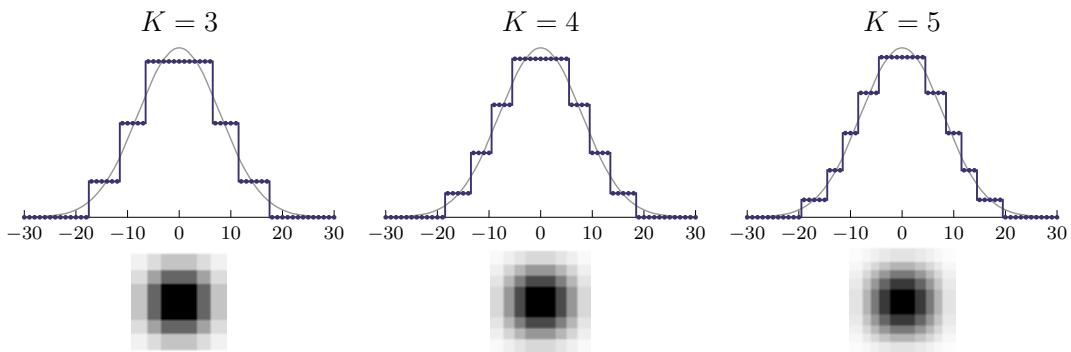


Figure 4: SII impulse responses in one and two dimensions.

**Alvarez–Mazorra** For  $K = 1$ , the impulse response is  $\frac{\nu\lambda}{1-\nu^2}\nu^{|n|}$  and has a sharp peak at  $n = 0$ . The impulse response looks much more Gaussian-like for  $K = 3$  and higher (see figure 5).

Figure 6 compares the original Alvarez–Mazorra method with  $q = \sigma$  to the proposed adjustment (42) for  $\sigma = 5$ ,  $K = 3$ . While using  $q = \sigma$  produces a Gaussian-like shape, it is too concentrated. The impulse response using (42) is more accurate. Further comparisons are made in the next section.



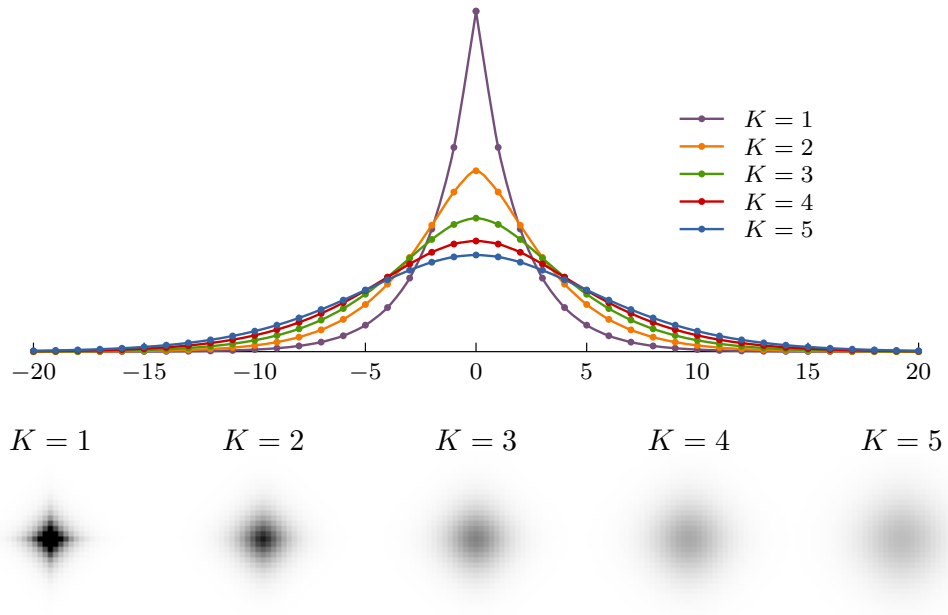


Figure 5: Alvarez–Mazorra impulse responses with  $K$  passes in one and two dimensions.

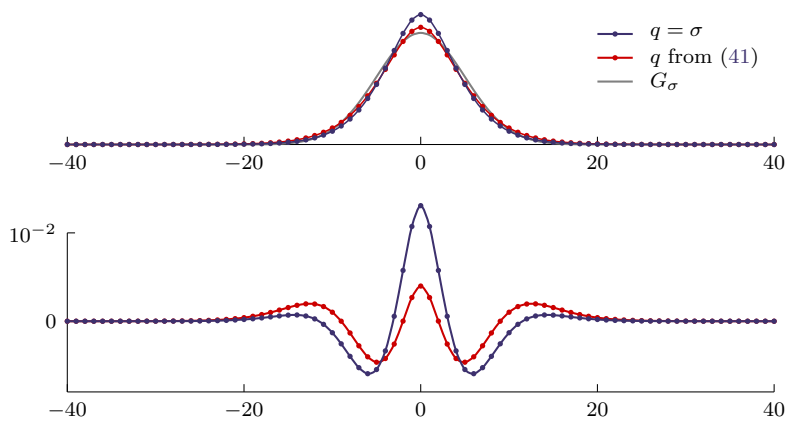


Figure 6: Effect of  $q$  in the Alvarez–Mazorra method. Top: impulse responses. Bottom: impulse response errors.

**Deriche** For  $K \geq 3$ , the difference between the Deriche impulse responses and  $G_\sigma$  is less than  $10^{-3}$ . Figure 7 plots these differences.

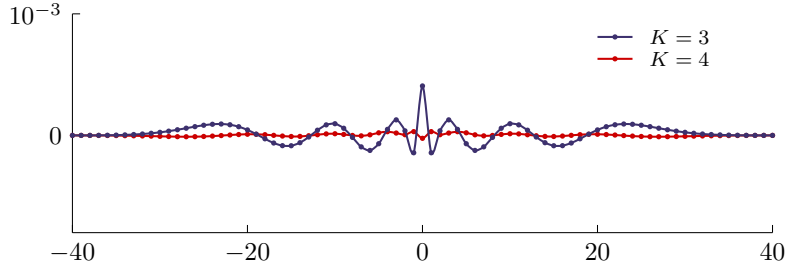


Figure 7: Deriche impulse errors.

**Vliet–Young–Verbeek** The Vliet–Young–Verbeek impulse responses are also very accurate, though not as accurate as Deriche impulse responses of the same order  $K$  (see figure 8).

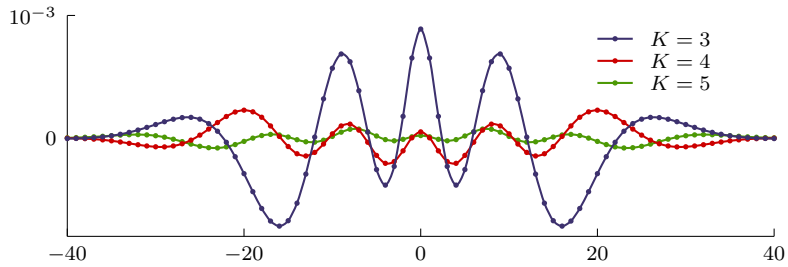


Figure 8: Vliet–Young–Verbeek impulse errors.

## 5 Comparison

This section compares the accuracy and speed of the algorithms. Accuracy is quantified by the  $\ell^\infty$  operator norm. Given linear operators  $L^{\text{exact}}$  and  $L$  representing the exact and approximate Gaussian convolutions, the  $\ell^\infty$  operator norm is  $\|L^{\text{exact}} - L\|_\infty$ , defined as the smallest constant such that

$$\|L^{\text{exact}} f - Lf\|_\infty \leq \|L^{\text{exact}} - L\|_\infty \|f\|_\infty \quad \text{for any } f \in \ell^\infty. \quad (50)$$

The exact convolution  $L^{\text{exact}}$  is approximated using the FIR method with  $tol = 10^{-15}$ .

We measure the accuracy and speed in double-precision arithmetic using the C source code included with this article running on a recent laptop.<sup>5</sup> For DCT-based convolution, the FFTW library [23] is used to compute the transforms. Note that speed measurements are sensitive to the platform and code optimizations, so these results should be considered as rough indications.

### 5.1 Accuracy and Speed for $\sigma = 5$

Table 3 compares accuracy and speed under the parameters  $N = 1000$ ,  $\sigma = 5$ , and boundary initialization accuracy  $tol = 10^{-6}$ . As shorthands, “box” refers to the basic iterated box filtering approximation with Wells’ formula, “ebox” denotes extended box filtering (section 3.3), “AM















































Algorithm	$\ell^\infty$ operator norm	Time (ms)
FIR, $tol = 10^{-2}$	 $3.8034 \times 10^{-3}$	 54
DCT	 $2.9092 \times 10^{-15}$	 61
box, $K = 3$	 $1.2921 \times 10^{-1}$	 12
box, $K = 4$	 $6.5507 \times 10^{-2}$	 16
box, $K = 5$	 $8.9585 \times 10^{-2}$	 19
ebox, $K = 3$	 $5.1577 \times 10^{-2}$	 21
ebox, $K = 4$	 $3.7858 \times 10^{-2}$	 28
ebox, $K = 5$	 $2.7937 \times 10^{-2}$	 35
SII, $K = 3$	 $2.0229 \times 10^{-1}$	 7
SII, $K = 4$	 $1.8654 \times 10^{-1}$	 9
SII, $K = 5$	 $1.7999 \times 10^{-1}$	 10
AM orig., $K = 3$	 $1.1278 \times 10^{-1}$	 34
AM orig., $K = 4$	 $8.7869 \times 10^{-2}$	 46
AM orig., $K = 5$	 $7.2186 \times 10^{-2}$	 57
AM, $K = 3$	 $7.8317 \times 10^{-2}$	 34
AM, $K = 4$	 $5.9480 \times 10^{-2}$	 46
AM, $K = 5$	 $4.8207 \times 10^{-2}$	 57
Deriche, $K = 2$	 $3.4845 \times 10^{-2}$	 15
Deriche, $K = 3$	 $4.4986 \times 10^{-3}$	 18
Deriche, $K = 4$	 $6.2498 \times 10^{-4}$	 20
VYV, $K = 3$	 $2.1031 \times 10^{-2}$	 16
VYV, $K = 4$	 $6.7471 \times 10^{-3}$	 19
VYV, $K = 5$	 $2.3703 \times 10^{-3}$	 21

Table 3: Accuracy and speed under the parameters  $N = 1000$ ,  $\sigma = 5$ , and boundary initialization accuracy  $tol = 10^{-6}$ .

orig.” denotes the original Alvarez–Mazorra method and “AM” with the proposed regression on  $q$  (section 3.5), and “VYV” denotes Vliet–Young–Verbeek (section 3.6).

These results show that at one extreme, SII and box filtering are the fastest Gaussian convolution algorithms, but with low accuracy. At the other extreme, DCT-based convolution is very accurate but slow. The algorithms with the most successful speed/accuracy tradeoff are Deriche and VYV. Deriche appears to be slightly better in both accuracy and speed, though VYV has the advantage that it may be computed in-place, unlike Deriche.

## 5.2 Accuracy for varying $\sigma$

The images in figure 9 repeat the preceding test with varying  $\sigma$ . We plot  $\sigma \in [0.5, 25]$  on the horizontal axis and the  $\ell^\infty$  operator error norm logarithmically on the vertical axis. Line color is used to indicate parameter  $K$  (orange  $\Rightarrow 2$ , green  $\Rightarrow 3$ , red  $\Rightarrow 4$ , blue  $\Rightarrow 5$ ).

For many algorithms, the error is much higher for  $\sigma < 2$ , then decreases and becomes steady for  $\sigma \geq 2$ . FIR filtering is the exception: its error is much lower for small  $\sigma$ , then becomes larger, though it is always bounded by  $tol$  as guaranteed by theorem 1.

DCT-based convolution computes  $g^{\text{sinc}} * f$  exactly, so aside from numerical imprecision, the error is entirely due to discrepancy between  $g$  and  $g^{\text{sinc}}$ . For small  $\sigma$ , this difference is significant, but for  $\sigma \geq 2$ , the difference is extremely small and the convolution is highly accurate.

The error plot for box filtering is oscillatory, the valleys corresponding to the quantized set of  $\sigma$  values that it can approximate. SII and extended box filtering have more consistent error for different  $\sigma$ , though the error is much lower with extended box filtering.

Arguably, some algorithms are simpler to implement than others. Extended box filtering and AM are attractive as having reasonable accuracy and speed and relatively simple algorithms.

<sup>5</sup>2.40GHz Intel® Core™ 2 Duo T7700 with 2GB RAM

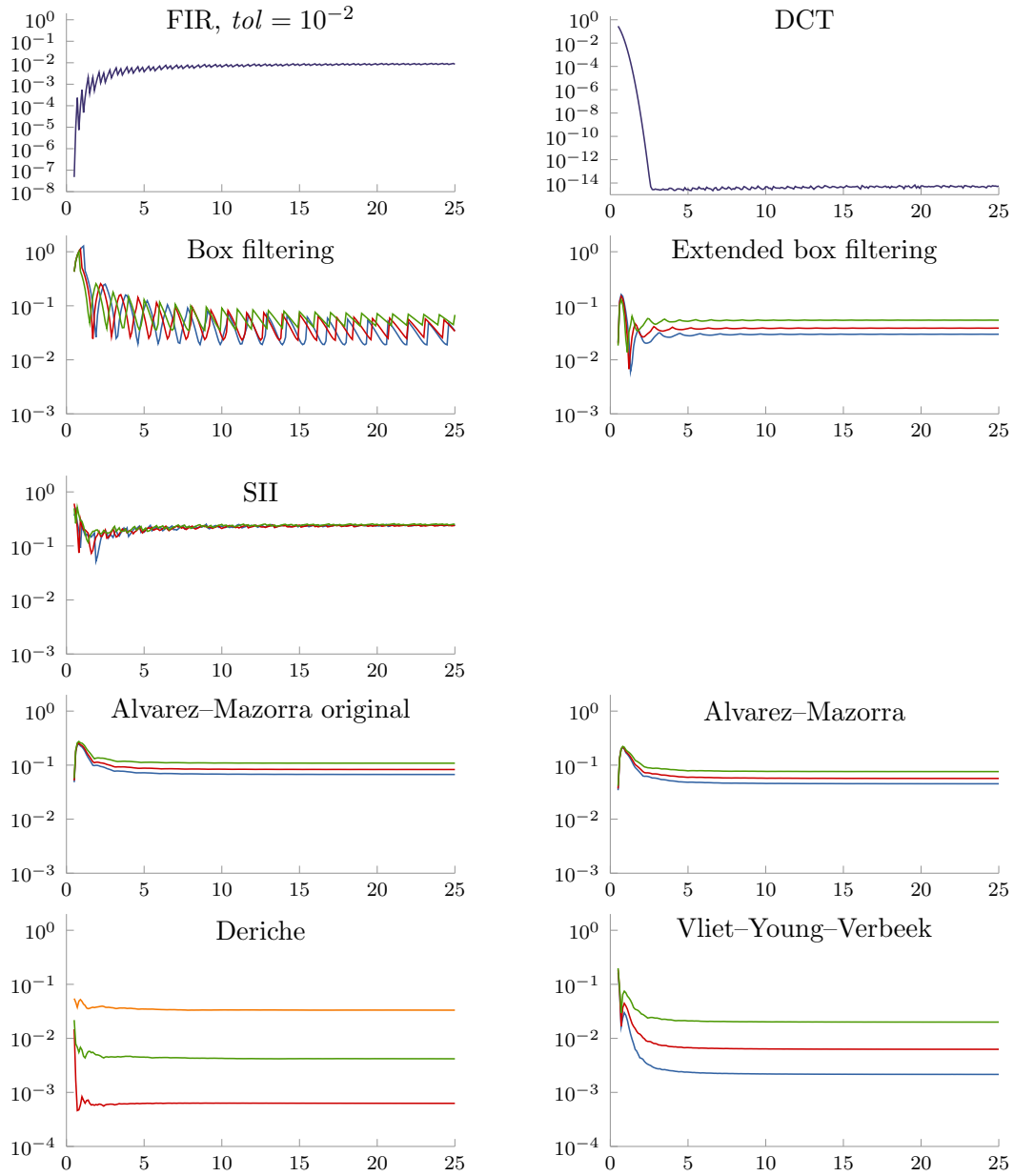


Figure 9:  $\ell^\infty$  operator error vs.  $\sigma \in [0.5, 25]$  for each algorithm. Line color indicates  $K$  (orange  $\Rightarrow 2$ , green  $\Rightarrow 3$ , red  $\Rightarrow 4$ , blue  $\Rightarrow 5$ ).

### 5.3 Speed for varying $\sigma$

For all algorithms except FIR, the value of  $\sigma$  has negligible effect on run time, testing over the range  $[0.5, 25]$ . In principle, the box filtering methods, Deriche, AM, and VYV depend on  $\sigma$  in the cost of boundary initialization, but no effect was measured, indicating that the majority of the cost is filtering the interior samples.

The notable exception is FIR convolution. With fixed  $tol$ , its computation time increases linearly with  $\sigma$  since the filter radius is selected as  $r = \lceil \sqrt{2} \operatorname{erfc}^{-1}(tol/2)\sigma \rceil$ . Figure 10 shows FIR computation time vs.  $\sigma$  with  $tol = 10^{-2}$  and  $N = 1000$ .

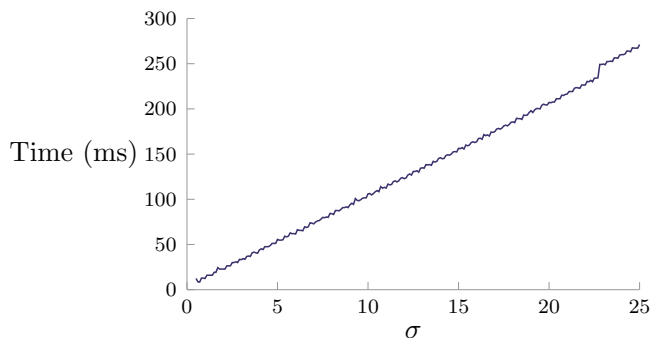


Figure 10: Run time vs.  $\sigma$  for FIR convolution.

### 5.4 Images

For visual purposes, Gaussian convolution can be approximated with low accuracy. For example, figure 11 shows that for  $\sigma = 5$ , the SII method with  $K = 3$  produces a result that appears similar to the exact convolution. Even a single pass of box filtering is a visually convincing approximation.

Accuracy may be more important when the Gaussian convolution is an intermediate step in a processing chain. Figure 12 plots nine level lines for each of the convolved images. The level lines are significantly different with the lower-accuracy methods.

As a color example, we perform Gaussian convolution with  $\sigma = 5$  on a color image by independently filtering its RGB channels (figure 13).

## 6 Conclusion

There is no single Gaussian convolution algorithm that is clearly best; the right choice is a consideration of aspects like accuracy, speed, memory, and ease of implementation. The results from this survey suggest the following recommendations (where  $T$  is a threshold roughly equal to 2):

- For high accuracy, use FIR for  $\sigma < T$  and Deriche or Vliet–Young–Verbeek for  $\sigma \geq T$ .
- For the best accuracy, use FIR for  $\sigma < T$  and DCT for  $\sigma \geq T$ .
- For the best speed, use SII or box filtering.
- For ease of implementation, use extended box filtering or Alvarez–Mazorra.

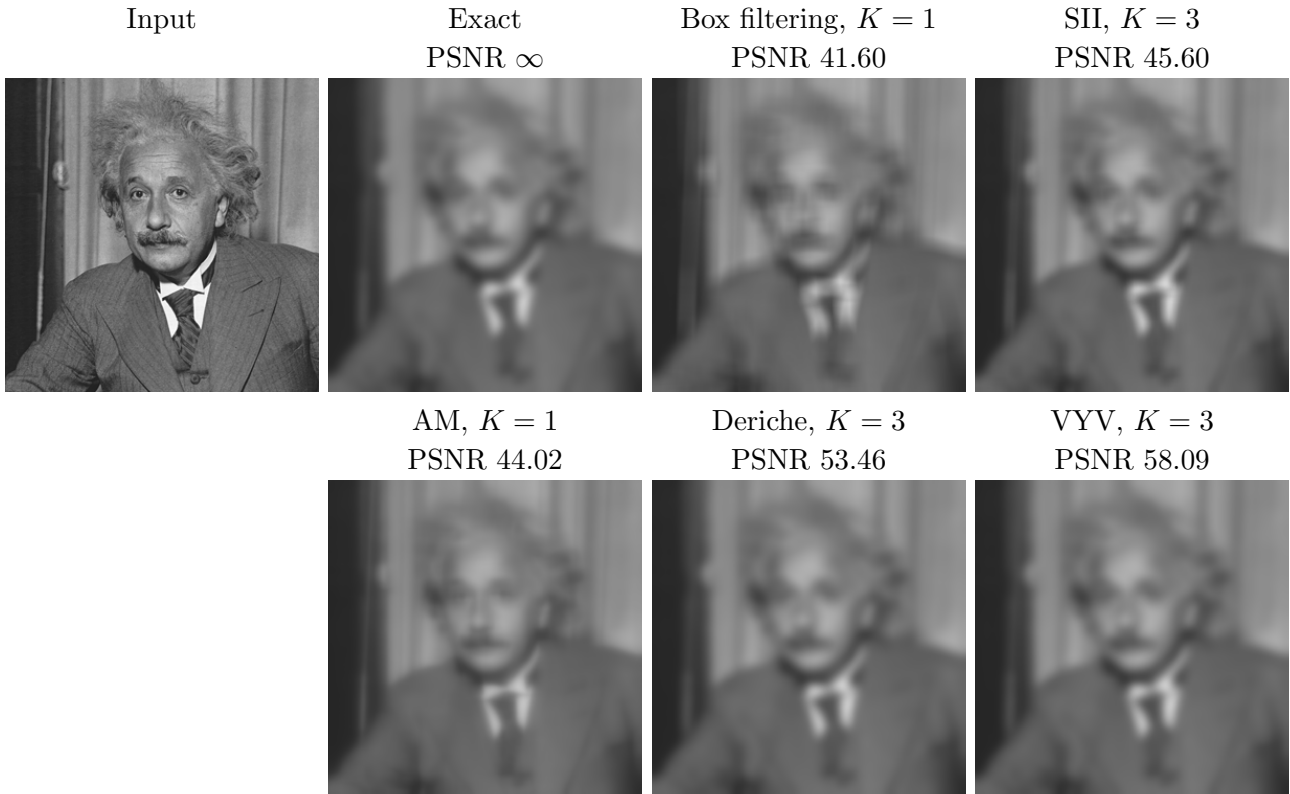


Figure 11: Results of different Gaussian convolution algorithms on a gray-scale image.

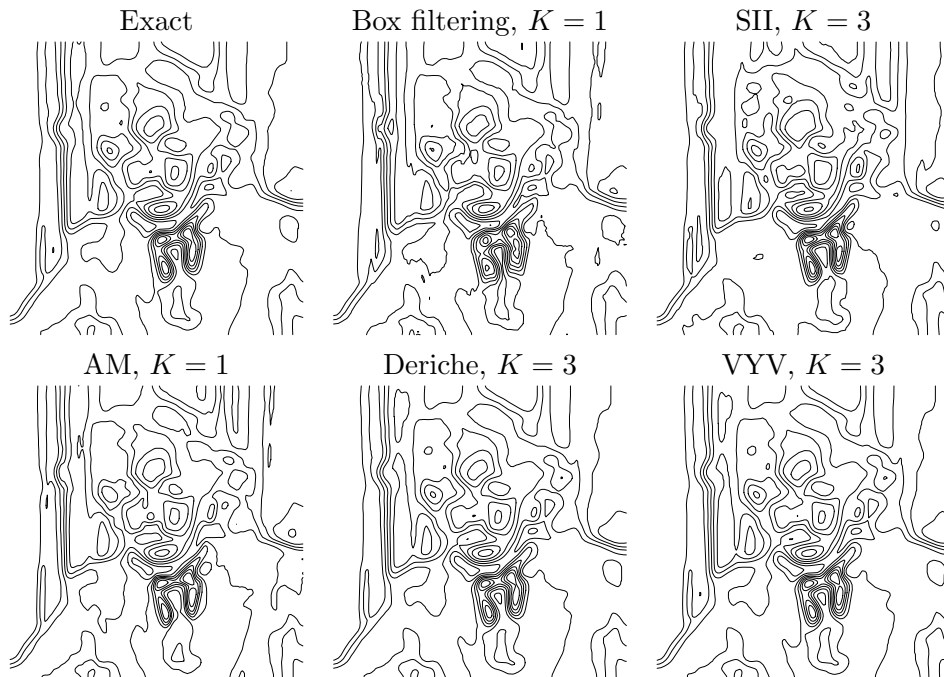


Figure 12: Nine levels for each of the convolved images shown in figure 11.

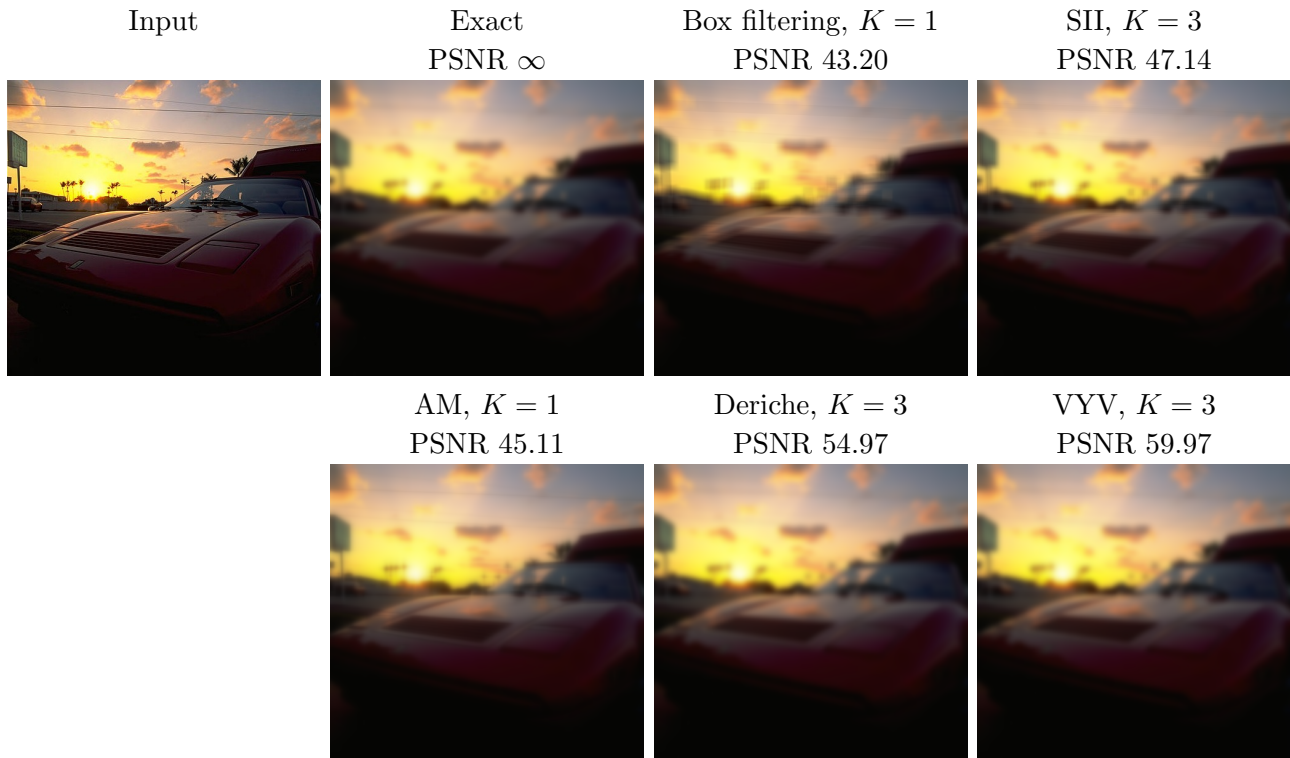


Figure 13: Results of different Gaussian convolution algorithms on a color image.

## Image Credits



Standard test image



Photograph courtesy Philip Greenspun (<http://philip.greenspun.com>)

## References

- [1] W. Heisenberg, “Über den anschaulichen inhalt der quantentheoretischen kinematic und mechanic,” *Zeit. Physik*, vol. 43, no. 172, 1927; *The Physical Principles of the Quantum Theory*, Dover Publications, 1930, ISBN:0486601137.
- [2] H. Weyl, *Gruppentheorie und quantenmechanik*, S. Hirzel, Leipzig, 1928; Dover Publications, 1950, ISBN:3534066685.
- [3] W.M. Wells, “Efficient synthesis of Gaussian filters by cascaded uniform filters,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 2, pp. 234–239, 1986. <http://dx.doi.org/10.1109/TPAMI.1986.4767776>
- [4] J. Babaud, A.P. Witkin, M. Baudin, R.O. Duda, “Uniqueness of the Gaussian kernel for scale-space filtering,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 1, pp. 26–33, 1986. <http://dx.doi.org/10.1109/TPAMI.1986.4767749>
- [5] A.L. Yuille, T.A. Poggio, “Scaling theorems for zero crossings,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 1, pp. 15–25, 1986. <http://dx.doi.org/10.1109/34.41383>

- [6] J. Canny, “A computational approach to edge detection,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 8, no. 6, pp. 679–698, 1986. <http://dx.doi.org/10.1109/TPAMI.1986.4767851>
- [7] T. Lindeberg, “Scale-space for discrete signals,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, no. 3, 1990. <http://dx.doi.org/10.1109/34.49051>
- [8] P. Duhamel, M. Vetterli, “Fast Fourier transforms: A tutorial review and a state of the art,” *Signal Processing*, vol. 19, no. 4, pp. 259–299, 1990. [http://dx.doi.org/10.1016/0165-1684\(90\)90158-U](http://dx.doi.org/10.1016/0165-1684(90)90158-U)
- [9] K.R. Rao, J. Ben-Ariem “Lattice architectures for multiple-scale Gaussian convolution, image processing, sinusoid-based transforms and Gabor filtering,” *Analog Integrated Circuits and Signal Processing*, vol. 4, no. 2, pp. 141–160, 1993. <http://dx.doi.org/10.1007/BF01254865>
- [10] R. Deriche, “Recursively implementing the Gaussian and its derivatives,” INRIA Research Report 1893, France, 1993. <http://hal.inria.fr/docs/00/07/47/78/PDF/RR-1893.pdf>
- [11] S. Martucci, “Symmetric convolution and the discrete sine and cosine transforms,” *IEEE Transactions on Signal Processing* SP-42, pp. 1038–1051, 1994. <http://dx.doi.org/10.1109/78.295213>
- [12] L. Alvarez, L. Mazorra, “Signal and image restoration using shock filters and anisotropic Diffusion,” *SIAM Journal on Numerical Analysis*, vol. 31, no. 2, pp. 590–605, 1994. <http://www.jstor.org/stable/2158018>
- [13] I.T. Young, L.J. van Vliet, “Recursive implementation of the Gaussian filter,” *Signal Processing*, vol. 44, no. 2, pp. 139–151, 1995. [http://dx.doi.org/10.1016/0165-1684\(95\)00020-E](http://dx.doi.org/10.1016/0165-1684(95)00020-E)
- [14] T. Lindeberg, “On the axiomatic foundations of linear scale-space: combining semigroup structure with causality vs. scale invariance,” In: J. Sporring et al. (eds.), *Gaussian Scale-Space Theory*, Kluwer Academic Publishers, pp. 75–98, 1997. ISBN:0792345614.
- [15] L. Evans, *Partial differential equations*, Graduate studies in mathematics, vol. 19, American Mathematical Society, 1998. ISBN: 0821807722.
- [16] B.C. Berndt, *Ramanujan’s notebooks, part V*, Springer–Verlag, New York, 1998, ISBN:0387949410.
- [17] L.J. van Vliet, I.T. Young, P.W. Verbeek, “Recursive Gaussian derivative filters,” *Proceedings of the 14th International Conference on Pattern Recognition*, vol. 1, pp. 509–514, 1998. <http://dx.doi.org/10.1109/ICPR.1998.711192>
- [18] D.G. Lowe, “Object recognition from local scale-invariant features,” *Proceedings of the International Conference on Computer Vision*, vol. 2. pp. 1150–1157, 1999. <http://dx.doi.org/10.1109/ICCV.1999.790410>
- [19] J. Weickert, S. Ishikawa, A. Imiya, “Linear scale-space has first been proposed in Japan,” *Journal of Mathematical Imaging and Vision*, vol. 10, no. 3, pp. 237–252. <http://dx.doi.org/10.1023/A:1008344623873>
- [20] I.T. Young, L.J. van Vliet, M. van Ginkel, “Recursive Gabor filtering,” *IEEE Transactions on Signal Processing*, vol. 50, no. 11, pp. 2798–2805, 2002. <http://dx.doi.org/10.1109/TSP.2002.804095>



- [21] J. Yi, “Theta-function identities and the explicit formulas for theta-function and their applications,” *Journal of Mathematical Analysis and Applications*, vol. 292, pp. 381–400, 2004. <http://dx.doi.org/10.1016/j.jmaa.2003.12.009>
- [22] S. Tan, J.L. Dale, A. Johnston, “Performance of three recursive algorithms for fast space-variant Gaussian filtering,” *Real-Time Imaging*, vol. 9, pp. 215–228, 2003. [http://dx.doi.org/10.1016/S1077-2014\(03\)00040-8](http://dx.doi.org/10.1016/S1077-2014(03)00040-8)
- [23] M. Frigo, S. G. Johnson, “The design and implementation of FFTW3,” *Proceedings of the IEEE*, vol. 93, no. 2, pp. 216–231, 2005. <http://dx.doi.org/10.1109/JPROC.2004.840301>
- [24] B. Triggs, M. Sdika, “Boundary conditions for Young–van Vliet recursive filtering,” *IEEE Transactions on Signal Processing*, vol. 54, no. 6, pp. 2365–2367, 2006. <http://dx.doi.org/10.1109/TSP.2006.871980>
- [25] G. Amayeh, A. Tavakkoli, G. Bebis, “Accurate and efficient computation of Gabor features in real-time applications,” *Advanced in Visual Computing, Lecture Notes in Computer Science*, vol. 5875/2009, pp. 243–252, 2009, [http://dx.doi.org/10.1007/978-3-642-10331-5\\_23](http://dx.doi.org/10.1007/978-3-642-10331-5_23)
- [26] F. Johansson et al., “mpmath: a Python library for arbitrary-precision floating-point arithmetic (version 0.14),” 2010. <http://code.google.com/p/mpmath>
- [27] A. Bhatia, W.E. Snyder, G. Bilbro, “Stacked integral image,” *IEEE International Conference on Robotics and Automation (ICRA)*, pp. 1530–1535, 2010. <http://dx.doi.org/10.1109/ROBOT.2010.5509400>
- [28] K.N. Chaudhury, A. Muñoz-Barrutia, M. Unser, “Fast space-variant elliptical filtering using box splines,” *IEEE Transactions on Image Processing*, vol. 19, no. 9, pp. 2290–2306, 2010. <http://dx.doi.org/10.1109/TIP.2010.2046953>
- [29] E. Elboher, M. Werman, “Efficient and accurate Gaussian image filtering using running sums,” *Computing Research Repository*, vol. abs/1107.4958, 2011. <http://arxiv.org/abs/1107.4958>
- [30] P. Gwosdek, S. Grewenig, A. Bruhn, J. Weickert, “Theoretical foundations of Gaussian convolution by extended box filtering,” *International Conference on Scale Space and Variational Methods in Computer Vision*, pp. 447–458, 2011. [http://dx.doi.org/10.1007/978-3-642-24785-9\\_38](http://dx.doi.org/10.1007/978-3-642-24785-9_38)
- [31] The NAG Library, *The Numerical Algorithms Group (NAG)*, Oxford, United Kingdom. <http://www.nag.co.uk>