Fast and Accurate Gaussian Derivatives based on B-Splines

Henri Bouma[1⋆], Anna Vilanova[1], Javier Oliván Bescós[2], Bart M. ter Haar Romeny[1], and Frans A. Gerritsen[1,2]

[1] Biomedical Image Analysis, Technische Universiteit Eindhoven, The Netherlands. henri.bouma@tno.nl

[2] Advanced Development, Healthcare Informatics, Philips Medical Systems, Best, The Netherlands.

Abstract. Gaussian derivatives are often used as differential operators to analyze the structure in images. In this paper, we will analyze the accuracy and computational cost of the most common implementations for differentiation and interpolation of Gaussian-blurred multi-dimensional data. We show that – for the computation of multiple Gaussian derivatives – the method based on B-splines obtains a higher accuracy than the truncated Gaussian at equal computational cost.

1 Introduction

Computer vision aims at the automatic interpretation of structures in an image. The low-level image structure is often analyzed with differential operators, which are used to calculate (partial) derivatives. In mathematical analysis, the derivative expresses the slope of a continuous function at a point ($\frac{\partial f(x)}{\partial x} = \lim_{h \downarrow 0} \frac{f(x+h)-f(x)}{h}$). In digital image analysis, signals can be modelled as a sum of of shifted dirac-delta impulses with varying amplitude. Differentiation of these discrete signals is ill-posed, since the derivatives do not continuously depend on the sampled input data [1]. Even a linear interpolation will not make the differentiation well-posed. The problem of an ill-posed differentiation on an image F is solved through a replacement of the derivative by a (well-posed) convolution with the derivative of a regularizing test function ϕ [2, 3].

$$\begin{aligned}(\partial_{i_1 \ldots i_n} F * \phi)(x) &= (-1)^n \int_{-\infty}^{\infty} F(\xi)\, \partial_{i_1 \ldots i_n} \phi(x+\xi)\, \mathrm{d}\xi \\ &= \int_{-\infty}^{\infty} F(\xi)\, \partial_{i_1 \ldots i_n} \phi(x-\xi) \mathrm{d}\xi = (F * \partial_{i_1 \ldots i_n} \phi)(x) \qquad (1)\end{aligned}$$

The Gaussian is the only regularizing function that is smooth, self-similar, causal, separable and rotation invariant [3, 4]. The convolution of an image with a Gaussian is called blurring, which allows the analysis at a higher scale where small structures (e.g., noise) are removed.

⋆ Henri Bouma recently joined TNO, The Hague, The Netherlands.

Thanks to the mentioned properties, the Gaussian derivatives are often applied in the fields of image processing and computer vision as differential operators [5]. They are used to implement differential invariant operators – such as edge detectors, shape descriptors and motion estimators. In the medical field, the Gaussian derivatives are used to compute features in *huge* multi-dimensional images for a computer-aided interpretation of the data, sometimes even at multiple scales [6]. This processing requires an efficient and accurate implementation of the Gaussian derivatives.

The naive approach to obtain the blurred derivatives of an image, is to convolve a multi-dimensional image with a multi-dimensional truncated Gaussian (derivative) kernel. The same result can be obtain with lower computational cost by using separability, because the rotation-invariant multivariate Gaussian is equal to a product of univariate Gaussians. However, the cost of both approaches increases as the scale gets larger. Therefore, many techniques are proposed for an efficient implementation at large scales or at multiple scales.

The FFT [7] allows the replacement of an expensive convolution in the spatial domain by a cheaper multiplication in the Fourier domain. Usually, the cost of an FFT is only acceptable for large scales [8]. A recursive implementation [9–11] of the Gaussian (derivative) is even cheaper than the FFT [12], and the costs are – like the FFT – independent of the scale. However, this implementation lacks high accuracy, especially for small scales [13] and derivatives cannot be computed between voxels (e.g., for rendering) or locally at some voxels (e.g., to save time and memory for the computation of isophote curvature on a sparse surface). The low-pass pyramid technique [14, 15] uses down-sampling at coarser scales to reduce the computational cost. Especially analysis at multiple or higher scales can benefit from this approach.

However, the use of large-scale Gaussian derivatives can be avoided because the Gaussian is a self-similar convolution operation. This means that a cascade application of two Gaussian kernels with standard deviation σ_1 and σ_2, results in a broader Gaussian function with $\sigma_{tot} = \sqrt{\sigma_1^2 + \sigma_2^2}$ (semi-group property). Therefore, Lindeberg [16] proposed to first blur an image once with a large Gaussian $G(\sigma_1)$, and then obtain all partial derivatives at lower cost with smaller Gaussian derivative kernels $g(\sigma_2)$. In this paper, we will compare the accuracy and computational cost of several approaches to obtain these derivatives.

Figure 1 shows four ways to obtain a Gaussian derivative. One way is to convolve an image in one pass with a truncated Gaussian derivative for each partial derivative. The second way is the approach of Lindeberg [16] that first blurs an image once and then obtains all the partial derivatives with small truncated Gaussian derivative kernels. Due to truncation, the Gaussian is not continuous and smooth anymore, although the error of this exponential function rapidly approaches zero. In the third way, which is similar to the second way, the small Gaussian derivative is replaced by a B-spline derivative [14, 17, 18]. The higher-order B-spline β converges to a Gaussian as a consequence of the central-limit theorem. An advantage of the B-spline of order n is that it is a compact kernel that guarantees C^{n-1} continuity. The fourth way to compute the Gaussian derivatives makes a separation between blurring and differentiation. After blurring the image – an operation that can benefit from the mentioned optimizations – the derivative is computed *without* blurring.

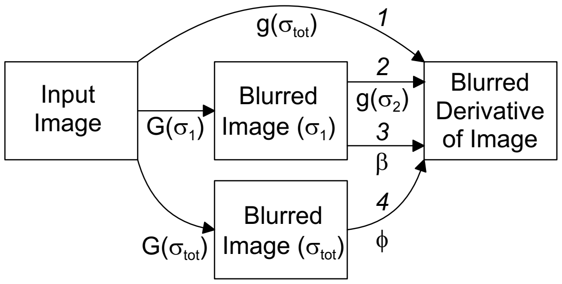


Fig. 1. Four ways to obtain the blurred derivative of an image. The first way performs one convolution with the derivative of a Gaussian $g(\sigma_{tot})$. The second and third way convolve the image with a Gaussian $G(\sigma_1)$ for most of the blurring and then with a smaller derivative of a Gaussian $g(\sigma_2)$ or a B-spline derivative β; where $\sigma_{tot} = \sqrt{\sigma_1^2 + \sigma_2^2}$. The fourth way convolves the image with a Gaussian $G(\sigma_{tot})$ for all the blurring and then with the derivative of an interpolator ϕ for differentiation.

Many operators have been proposed to compute the derivative in an image (e.g., the Roberts, Prewitt and Sobel operators [19–21]). However, they do not compute the derivative without adding extra blur and they are very inaccurate.

The unblurred derivative of an image can be computed as a convolution with the derivative of an interpolation function ϕ (4th way in Fig. 1). A quantitative comparison of several interpolation methods can be found in papers by Meijering *et al.* [23, 24], Jacobs *et al.* [25] and Lehmann *et al.* [26]. The comparisons show that for each of the methods for differentiation and interpolation there is a trade-off between accuracy, continuity and kernel size, and that *B-spline interpolation* [27, 28] appears to be superior in many cases. Therefore, we used the derivative of a B-spline interpolator to implement the unblurred derivative (4th way in Fig. 1).

In the last decades, we have seen a growing competition between Gaussian- and spline-based image-analysis techniques, which are both frequently used. To our knowledge, a comparison between the truncated Gaussian and the approaches based on B-spline approximation and B-spline interpolation (Fig. 1) for a fast and accurate implementation of Gaussian derivatives has not been published before. In this paper, we will compare the accuracy (Sec. 2) and computational cost (Sec. 3) of the four strategies.

2 Accuracy of Methods

In this section, the true Gaussian derivatives are compared to their approximations to analyze the accuracy of these approximations on one-dimensional data. Thanks to the separability of the Gaussian $\mathcal{G}$, this analysis is also valid for higher dimensions.

$$\mathcal{G}(x,\sigma) = \frac{1}{\sigma\sqrt{2\pi}}\, e^{-\frac{x^2}{2\sigma^2}} \tag{2}$$

The error ϵ of an approximation $\tilde{y}$ of the true continuous signal y is computed as the normalized RMS-value, which is directly related to the energy.

$$\epsilon = \frac{\sqrt{\int_{-\infty}^{\infty} |\tilde{y}(x) - y(x,\sigma)|^2 \, \mathrm{d}x}}{\sqrt{\int_{-\infty}^{\infty} |y(x,\sigma)|^2 \mathrm{d}x}} \tag{3}$$

For the normalized RMS-value, the error of the impulse response of a Gaussian-derivative kernel that is truncated at $x = a\sigma$ is independent of the standard deviation. For example, the normalized RMS-error of a Gaussian is $\epsilon = \sqrt{1 - \mathrm{erf}(a)}$, and for a first-order Gaussian derivative: $\epsilon = \sqrt{1 + 2\,a\,e^{-a^2}/\sqrt{\pi} - \mathrm{erf}(a)}$.

2.1 Aliasing and Truncation of the Gaussian

Small-scale Gaussian operators may suffer from sampling artifacts. According to the Nyquist theorem, the sampling frequency must be at least twice the bandwidth in order to avoid overlap of the copied bands (aliasing) [5]. If the copies do not overlap, then perfect reconstruction is possible. For a small amount of blurring (e.g., $\sigma < 1$ pixel) limitation of the bandwidth of the reconstructed signal is not guaranteed and serious aliasing artifacts may be the result. Band-limitation is enforced by a convolution with a sinc function [13]. If high-frequencies are (almost) absent in the stop band, then aliasing is negligible and the signals with and without band-limitation ($g * \phi_{sinc}(x)$ and g(x) respectively) are (approximately) equal. Figure 2a shows that sampling causes serious aliasing artifacts for a small-scale zeroth-order derivative of a Gaussian, and it shows that a first- or second- order derivative requires even more blurring for the same reduction of the aliasing effects. To avoid aliasing artifacts, second-order derivatives are often computed at $\sigma = 2.0$ pixels. Therefore, we will mainly focus further analysis on this amount of blurring ($\sigma_{tot} = 2.0$ px).

For a fixed kernel size N, small scales will lead to aliasing artifacts but large scales will lead to truncation artifacts. The optimal trade-off between aliasing and truncation is selected by minimizing the difference between a band-limited Gaussian ($g * \phi_{sinc}$) and a truncated Gaussian. Figure 2b shows that the error is minimal at $\sigma \approx (N/6.25)^{0.50} \approx \sqrt{N/6}$. If the truncated kernel is applied to a blurred input signal – e.g. blurred by the PSF or pre-filtering with σ_1 – so that the total blurring is $\sigma_{tot} = \sqrt{\sigma_1^2 + \sigma_2^2} = 2.0$ px, the optimal scale can even be reduced to approximately $\sigma_2 \approx (N/9.9)^{0.56} \approx \sqrt{N/10}$, as shown in Figure 2c. The scale with a minimal error is used to implement the second approach in Figure 1.

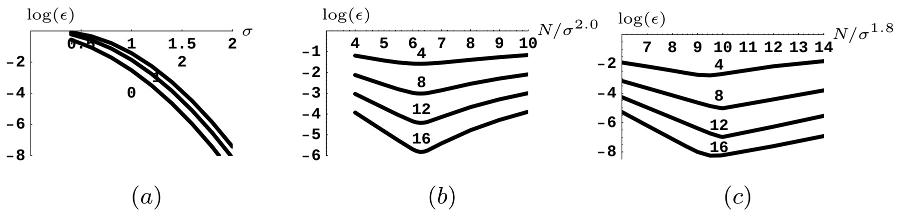


Fig. 2. (a) The normalized RMS-error ϵ due to aliasing for a zeroth-, first- and second-order derivative of a Gaussian at $\sigma = [0.5 - 2.0]$. (b) The difference between a band-limited Gaussian and a truncated Gaussian is minimal at $\sigma = (N/6.25)^{0.50}$, where the kernel size $N = [4, 8, 12, 16]$. (c) On a blurred signal, the normalized RMS-error is minimal at $\sigma = (N/9.9)^{0.56}$.

2.2 B-Spline Approximation

The B-spline approximator is used to implement the third approach in Figure 1. A high-order B-spline [18], or a cascade application of kernels [17, 14], will converge to a Gaussian (central-limit theorem). The B-spline approximator $\beta^n(x)$ of order n is:

$$\beta^n(x) := \frac{1}{n!} \sum_{i=0}^{n+1} \binom{n+1}{i} (-1)^i \; \mu^n\left(x - i + \frac{n+1}{2}\right) \tag{4}$$

where $\mu^n(x)$ is x^n for $x \geq 0$ and zero for other values, and where $\binom{n+1}{i}$ is the binomial coefficient. The derivatives of the B-spline can be obtained analytically in a recursive fashion based on the following property:

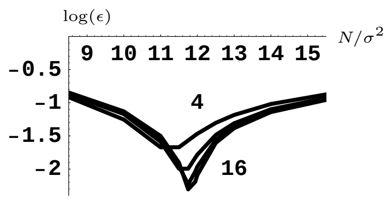
$$\frac{\partial \beta^n(x)}{\partial x} = \beta^{n-1}\left(x + \frac{1}{2}\right) - \beta^{n-1}\left(x - \frac{1}{2}\right) \tag{5}$$

The z-transform [29] is commonly used in digital signal processing to represent filters in the complex frequency domain. For example, for cubic ($n = 3$) spline filtering, the z-transform is:

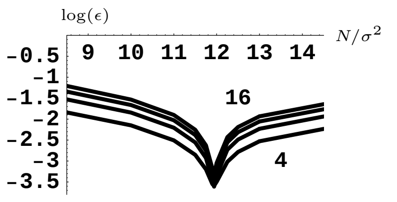
$$B^3(z) = \frac{1\,z^{-1} + 4\,z^0 + 1\,z^1}{6} \quad \Leftrightarrow \quad y_i = \frac{1}{6}x_{i-1} + \frac{4}{6}x_i + \frac{1}{6}x_{i+1} \tag{6}$$

The output y_i of this digital filter only depends on the inputs x, which makes it a *finite impulse response* (FIR) filter.

The derivative of a B-spline approximator $\beta^n(x)$ can be used as a small-scale Gaussian derivative. Figure 3a shows the normalized RMS-error between a Gaussian and a B-spline is minimal for the standard deviation $\sigma = \sqrt{N/12}$ [30]. Although the B-spline converges to a Gaussian for higher orders, the error is not reduced for higher orders (Fig. 3b) when it is applied to a blurred signal (to obtain $\sigma_{tot} = 2.0$ px). The scale with a minimal error is used to analyze the accuracy of this approach.



(*a*)



(*b*)

Fig. 3. The normalized RMS-error between a Gaussian and a B-spline approximator is minimal at $\sigma = \sqrt{N/12}$, for kernel size $N = 4, 8, 12, 16$. (b) The same relation can be found on a blurred signal.

2.3 B-Spline Interpolation

The B-spline interpolator is used to implement the fourth approach in Figure 1. In order to perform B-spline interpolation of the blurred image H with the approximating B-spline kernels (β^n in Eq. 4), an inverse operation B^n_{inv} is required.

$$\tilde{h} = H * \phi = H \, * \, B^n_{inv} \, * \, \beta^n \tag{7}$$

The inverse operator can easily be calculated in the z-domain as $B^n(z)^{-1}$. To obtain a stable filter, this inverse operator can be decomposed by its negative roots with magnitude smaller than one [27]. For example, the root of the inverse of a cubic B-spline (Equations 6 and 8) is $\lambda = -2 + \sqrt{3}$.

$$B^3(z)^{-1} = \frac{1}{B^3(z)} = \frac{6}{z^1 + 4 + z^{-1}} = -6\lambda \frac{1}{(1 - \lambda z^{-1})} \frac{1}{(1 - \lambda z^1)} \tag{8}$$

Multiplication of two parts in the z-domain is equivalent to a cascade convolution with both parts in the spatial domain. The last part in Equation (8), with z^1, can be applied backward, so that it also becomes a z^{-1} operation. This results in a stable and fast filter, which should be applied forward and backward:

$$\frac{1}{1 - \lambda z^{-1}} \quad \Leftrightarrow \quad y_i = x_i + \lambda \, y_{i-1} \tag{9}$$

The output y_i of this digital filter does not only depend on the input x_i, but also on the output y_{i-1}, which makes it a recursive – or *infinite impulse response* (IIR) – filter. The recursive inverse operation makes the B-spline interpolator computationally more expensive than the B-spline approximator at equal order n. For more information about B-Spline interpolation, we refer to the work of Unser *et al.* [27, 28].

2.4 Comparison of Accuracy

An experiment was performed to estimate the normalized RMS-error between the impulse response of a continuous Gaussian derivative ($\sigma_{tot} = 2.0$ px to avoid aliasing) and each of the four approaches (Fig. 1). Measuring for each approach the error of the impulse response gives an indication of the accuracy in general, because a discrete image can be modelled as a sum of impulses with varying amplitude. The first approach, which is based on a one-pass truncated Gaussian of $\sigma = 2.0$ pixels, used an unblurred impulse as input signal. The second and third approach, which are based on a small-scale truncated Gaussian and on a B-spline approximator, used a sampled Gaussian as an input signal to obtain a total blurring of $\sigma = 2.0$ pixels. The fourth approach, which is based on B-spline interpolation, used a sampled Gaussian of $\sigma = 2.0$ pixels as input signal.

Truncation of the one-pass Gaussian is often performed at 3σ or 4σ, which corresponds to a kernel size of 12 or 16 pixels for $\sigma = 2.0$ pixels. Figure 4 shows that for these kernel sizes the normalized RMS-error in the second-order derivative is $5.0 \cdot 10^{-2}$ or $2.4 \cdot 10^{-3}$ respectively. The results show that B-spline approximation requires much smaller kernels to obtain the same accuracy as the truncated Gaussian (4 or 6 px respectively). The figure also shows that B-spline interpolation and cascade application of small-scale Gaussians may be interesting if higher accuracies are required, but for most applications the approach based on B-spline approximation will be sufficiently accurate.

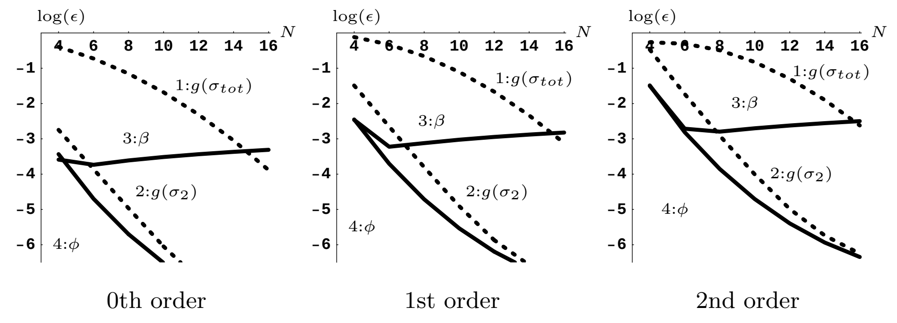


Fig. 4. The normalized RMS-error ϵ in estimating the zeroth-, first- and second-order Gaussian derivative ($\sigma = 2.0$px) for the four approaches based on the one-pass truncated Gaussian ($g(\sigma_{tot})$, dashed), the cascade application Gaussians ($g(\sigma_2)$, dashed), the B-spline approximator (β, solid) and the B-spline interpolator (ϕ, solid). The first approach requires much larger kernels than the others to obtain the same accuracy.

3 Computational Cost

Our comparison of computational cost will focus on the calculation of first- and second-order derivatives at a low scale ($\sigma = 2.0$ px) in three-dimensional (3D) data, because these derivatives are frequently used in the medical field. For these parameters, we will show that – in most cases – it is beneficial to use the B-spline approximator. For larger scales, more derivatives or higher-dimensionality it will be even more beneficial to make a separation between the blurring and differentiation. Therefore, our analysis can easily be extended to the computation of an arbitrary number of derivatives at higher scales on multi-dimensional data.

Figure 4 showed that the truncated Gaussian requires 12 or 16 pixels to obtain the same accuracy as the B-spline approximator of 4 or 6 pixels respectively. For these sizes the B-spline approximator (*B-spl.A*) is more accurate than the cascaded Gaussians ($\mathcal{G}(\sqrt{N/10})$) and computationally cheaper than the B-spline interpolator (*B-spl.I*) because no inverse is required (Eq. 7). Therefore, we will focus on the comparison of the B-spline approximator with the truncated Gaussian. Despite its small kernel, the B-spline is not always cheaper than the truncated Gaussian because it requires preprocessing to obtain the same amount of blur. The computational cost of this global blurring step can be reduced – especially for large scales – by using a recursive implementation [11].

The estimation of the computational cost C is based on the number of multiplications, which is equal to the kernel size. Three approaches are distinguished to analyze the performance for different purposes (Table 1). In the first approach, all volume-elements (voxels) are processed in a 3D volume. In the second, the derivatives are computed at some voxel-locations, and in the third, interpolation and differentiation is allowed at arbitrary (sub-voxel) locations in the volume. Finally, our estimation of the computational cost is verified with an experiment.

Table 1. The computational cost C in a 3D volume of d derivatives based on the truncated Gaussian (kernel size k) and B-spline approximation (order n).

	Blur	All Voxels	Some Voxels	Some Points
Trunc. Gauss	–	$3\,d\,(k+1)$	$d\,(k+1)^3$	$d\,(k)^3$
B-spline approx.	$3\,(k+1)$	$3\,d\,(n)$	$d\,(n)^3$	$d\,(n+1)^3$

3.1 Cost of Differentiation on All Voxels

The computation of Gaussian derivatives on all voxels allows the use of a separable implementation with discrete one-dimensional filters. The continuous B-spline of order n with kernel size $n+1$ is zero at the positions $-(n+1)/2$ and $(n+1)/2$. Therefore, the number of non-zero elements in a discrete B-spline kernel is n. The truncated Gaussian with kernel size k is not zero at its end points and therefore it requires $k+1$ elements in the discrete kernel to avoid the loss of accuracy.

For a 'fast' computation ($n = 3$, $k = 12$) of three first-order derivatives on all voxels, the B-spline approximator is 1.8 times faster than the truncated Gaussian despite the required preprocessing. For the nine first- and second-order derivatives, the B-spline is 2.9 times faster. For a 'more-accurate' computation ($n = 5$, $k = 16$) of three or nine derivatives, the B-spline approximator is 1.6 resp. 2.5 times faster than the truncated Gaussian (horizontal lines in Fig. 5).

3.2 Cost of Differentiation on Some Voxels

If only a small percentage p of the volume needs to be processed (e.g, to compute shape descriptors on the surface of an object) – or if storage of multiple derivatives of the whole image consumes too much memory – the non-separable implementation may be more efficient to compute the derivatives than the separable implementation. However, in 3D data, the cost of a non-separable *local* operation increases with a *power* of three instead of a *factor* of three (Tab. 1).

The non-separable implementation is more efficient than the separable for the 'fast' B-spline approximator ($n = 3$) if less than $p = 33\%$ of the volume is processed, and for the 'more-accurate' ($n = 5$) if less than $p = 12\%$ is processed (Fig. 5).

Figure 5 also shows that the B-spline implementation ($n = 3$, $d = 3$) is more efficient than the truncated Gaussian if more than $p = 0.6\%$ of the voxels is processed ($d = 9$ reduces the trade-off point to $p = 0.2\%$). For example, the B-spline ($n = 3$, $d = 9$) is 8 times faster than the truncated Gaussian at $p = 2.0\%$.

If we would have assumed that the blurring for the B-splines was incorporated in the preprocessing, then the B-spline approximator would even have been 81 times faster than the truncated Gaussian for each voxel.

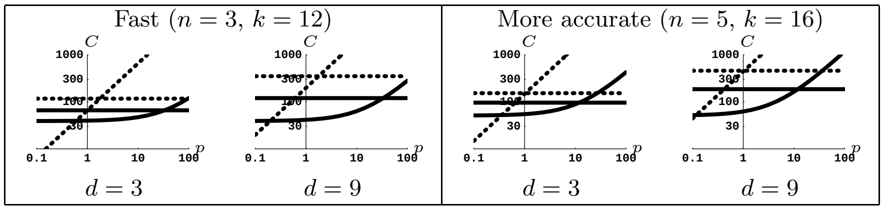


Fig. 5. The curves show the computational cost C for processing a percentage p of the voxels with a non-separable implementation in a 3D volume with a truncated Gaussian (dashed) and the B-spline (solid) for d derivatives. The horizontal lines show the cost of processing all voxels with a separable implementation. The plots show that the B-spline is expected to be more efficient if more than $p = 0.6\%$ of the data is processed.

3.3 Cost of Interpolation and Differentiation on Arbitrary Points

To interpolate and differentiate at arbitrary (sub-voxel) points in the volume continuous kernels are needed and a separable implementation cannot be used. The n-th order B-spline has a continuous kernel size of $n + 1$ (Table 1).

Figure 6 shows that if more than $p = 0.8\%$ of the data is processed the B-spline is more efficient than the truncated Gaussian. For example, if nine derivatives are computed at a number of points that equals $p = 10\%$ of the voxels, the B-spline (n=3) is more than 16 times faster than the truncated Gaussian ($k = 12$).

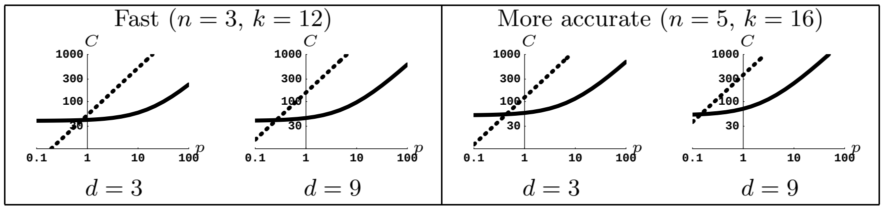


Fig. 6. The computational cost C for processing arbitrary points as a percentage p of the voxels in a 3D volume with a truncated Gaussian (dashed) and the B-spline (solid) for d derivatives. The plots show that the B-spline is expected to be more efficient if more than $p = 0.8\%$ of the data is processed.

3.4 Validation of Cost of Differentiation on Voxels

To validate our estimation of the computational cost, we measured the time that was required to compute the nine first- and second-order derivatives on a 3D volume of 512x512x498 voxels with a Pentium Xeon 3.2 GHz processor. In this experiment, we compared the implementations based on the truncated Gaussian ($k = 12$) and the B-spline approximator ($n = 3$) as an example to show that our assumptions are valid. The measured results in Figure 7 are in good agreement with our analysis. The measurements show that the B-spline is more efficient if more than 0.3% of the data is processed (estimated 0.2%). The B-spline appears to be 6 times faster than the truncated Gaussian if 2% of the volume is processed with a non-separable implementation (estimated 8 times faster). And if all voxels are processed with a separable implementation the B-spline appears to be 2.1 times faster (estimated 2.9 times faster).

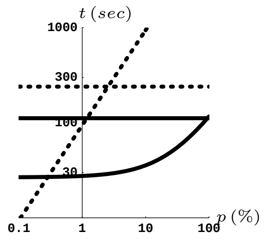


Fig. 7. The measured computation time t in seconds for processing a percentage p of the voxels in a 3D volume (512x512x498 voxels) with a truncated Gaussian ($k = 12$, dashed) and a B-spline approximator ($n = 3$, solid) for 9 derivatives. The horizontal lines show the cost of processing all voxels with a separable implementation. The plot shows that, for equivalent accuracy, the B-spline is more efficient if more than $p = 0.3\%$ of the data is processed.

4 Conclusions

We analyzed the accuracy and computational cost of several common implementations for differentiation and interpolation of Gaussian blurred multi-dimensional data. An efficient implementation is extremely important for all fields that use Gaussian derivatives to analyze the structure in data. A comparison between an implementation based on the truncated Gaussian and alternative approaches based on B-spline approximation and B-spline interpolation has not been published before, to the best of our knowledge.

If the vesselness or isophote curvature of a data set needs to be computed (requiring six or nine derivatives respectively), the B-spline approach will perform much faster than the approach based on truncated Gaussians. These operators are very important in the field of medical imaging for shape analysis. Our analysis shows that, for the computation of first- and second-order Gaussian derivatives on three-dimensional data, the B-spline approximator is faster than the truncated Gaussian at equal accuracy, provided that more than 1% of the data is processed. For example, if 2% of a 3D volume is processed, B-spline approximation is more than 5 times faster than the truncated Gaussian at equal accuracy. Our analysis can be extended easily to an arbitrary number of derivatives on multi-dimensional data.

Higher accuracy will not always lead to better results. However, in many cases, the same accuracy can be obtained at lower computational cost, as was shown in this paper. Another advantage of the B-spline of order n is that C^{n-1} continuity is guaranteed, whereas the truncated Gaussian is not even C^0 continuous.

References

1. J. Hadamard: Sur les problèmes aux Dérivées Partielles et leur Signification Physique. Bulletin, Princeton University **13** (1902) 49–62
2. L. Schwartz: Théorie des distributions. In: Actualités Scientifiques et Industrielles, Institut de Mathématique, Université de Strasbourg. Vol. 1,2. (1951) 1091–1122
3. L.M.J. Florack: Image Structure. Kluwer Academic Publ., The Netherlands (1997)
4. R. Duits, L.M.J. Florack, J. de Graaf and B.M. ter Haar Romeny: On the axioms of scale-space theory. J. Mathematical Imaging and Vision **20**(3) (2004) 267–298
5. B.M. ter Haar Romeny: Front-End Vision and Multi-Scale Image Analysis. Kluwer Academic Publ., The Netherlands (2003)
6. Y. Masutani, H. MacMahon and K. Doi: Computerized detection of pulmonary embolism in spiral CT angiography based on volumetric image analysis. IEEE Trans. Medical Imaging **21**(12) (2002) 1517–1523
7. M. Frigo and S.G. Johnson: An FFT compiler. Proc. IEEE **93**(2) (2005) 216–231
8. L.M.J. Florack: A spatio-frequency trade-off scale for scale-space filtering. IEEE Trans. Pattern Analysis and Machine Intelligence **22**(9) (2000) 1050–1055
9. M. Abramowitz and I.A. Stegun: Handbook of Mathematical Functions. Dover, New York, USA (1965)
10. R. Deriche: Fast algorithms for low-level vision. IEEE Trans. Pattern Analysis and Machine Intelligence **12**(1) (1990) 78–87

11. L.J. van Vliet, I.T. Young and P.W. Verbeek: Recursive Gaussian derivative filters. In: Proc. Int. Conf. Pattern Recognition (ICPR). Vol. 1. (1998) 509–514
12. I.T. Young and L.J. van Vliet: Recursive implementation of the Gaussian filter. Signal Processing, Elsevier **44** (1995) 139–151
13. R. van den Boomgaard and R. van der Weij: Gaussian convolutions, numerical approximations based on interpolation. In: Proc. Scale Space. LNCS 2106 (2001) 205–214
14. P.J. Burt and E.H. Adelson: The Laplacian pyramid as a compact image code. IEEE Trans. Communications **31**(4) (1983) 532–540
15. J.L. Crowley and R.M. Stern: Computation of the difference of low-pass transform. IEEE Trans. Pattern Analysis and Machine Intelligence **6**(2) (1984) 212–222
16. T. Lindeberg: Discrete derivative approximations with scale-space properties: A basis for low-level feature extraction. J. Mathematical Imaging and Vision **3**(4) (1993) 349–376
17. M. Wells: Efficient synthesis of Gaussian filters by cascaded uniform filters. IEEE Trans. Pattern Analysis and Machine Intelligence **8**(2) (1986) 234–239
18. Y.P. Wang and S.L. Lee: Scale space derived from B-splines. IEEE Trans. Pattern Analysis and Machine Intelligence **20**(10) (1998) 1040–1055
19. I.E. Abdou and W.K. Pratt: Quantitative design and evaluation of enhancement/thresholding edge detectors. Proc. IEEE **67**(5) (1979) 753–763
20. V.S. Nalwa and T.O. Binford: On detecting edges. IEEE Trans. Pattern Analysis and Machine Intelligence **8**(6) (1986) 699–714
21. V. Torre and T.A. Poggio: On edge detection. IEEE Trans. Pattern Analysis and Machine Intelligence **8**(2) (1986) 147–163
22. S.R. Marschner and R.J. Lobb: An evaluation of reconstruction filters for volume rendering. In: Proc. IEEE Visualization. (1994) 100–107
23. E.H.W. Meijering, W.J. Niessen and M.A. Viergever: The sinc-approximating kernels of classical polynomial interpolation. In: Proc. IEEE Int. Conf. Image Processing. Vol. 3. (1999) 652–656
24. E.H.W. Meijering, W.J. Niessen and M.A. Viergever: Quantitative evaluation of convolution-based methods for medical image interpolation. Medical Image Processing **5**(2) (2001) 111–126
25. M. Jacob, T. Blu and M. Unser: Sampling of periodic signals: A quantitative error analysis. IEEE Trans. Signal Processing **50**(5) (2002) 1153–1159
26. T.M. Lehmann, C. Gönner and K. Spitzer: Survey: Interpolation methods in medical image processing. IEEE Trans. Medical Imaging **18**(11) (1999) 1049–1075
27. M. Unser, A. Aldroubi and M. Eden: B-spline signal processing: Part I: Theory, and Part II: Efficient design and applications. IEEE Trans. Signal Processing **41**(2) (1993) 821–848
28. M. Unser: Splines, a perfect fit for signal and image processing. IEEE Signal Processing Magazine (1999) 22–38
29. E.I. Jury: Theory and Application of the Z-Transform Method. John Wiley and Sons, New York, USA (1964)
30. M. Unser, A. Aldroubi and M. Eden: On the asymptotic convergence of B-spline wavelets to Gabor functions. IEEE Trans. Inf. Theory **38**(2) (1992) 864–872