DISCRETE SCALE SPACES

ANDERSON CUNHA, RALPH TEIXEIRA and LUIZ VELHO
IMPA - Instituto de Matematica Pura e Aplicada

Abstract. Scale spaces allow us to organize, compare and analyse differently sized structures of an object. In this work, we present and compare five ways of discretizing the Gaussian scale-space: sampling Gaussian distributions; recursively calculating Gaussian approximations; using Splines; approximating by first-order generators; and finally, by a new method we call "Crossed Convolutions".

Key words: Scale spaces, Gaussian, Splines, Crossed convolutions.

1. Introduction

The concept of *scale* is intrinsic to the observation of any physical object. In order to obtain information or features from an object, we must use an appropriate scale where such information is easily observable.

If we want a generic system that does not know *a priori* the desired scale for extracting information from an object (or sequence of images), then it might be necessary to create a representation of such an object in several scales. Some desirable properties for a multi-scale representation are:

- Isotropy or rotation invariance: there are no preferred directions.
- Homogeneity or translation invariance: there are no preferred locations.
- Causality: no structure is created when the scale increases.

Note that these properties (specially the invariance by translations) imply that the size of the support of an object can't change with scale. In particular, the scale space of an image will be a sequence of images, *all of the same size of the original image*. When the scale increases, the images get more blurred, as if we were watching the images from a larger distance, but the support's size is kept fixed. The blurring is a direct consequence of the causality principle.

2. Gaussian Scale-Spaces

Definition 1 *Let* $f : \mathbf{R}^n \to \mathbf{R}$. *The* ***Gaussian scale-space*** *of* f *is the function* $L : \mathbf{R}^n \times \mathbf{R}^+ \to \mathbf{R}$ *given by* $L(\mathbf{x}, \mathbf{t}) = \mathbf{f} * \mathbf{G_t}(\mathbf{x})$, *where* $L(\mathbf{x}, \mathbf{0}) = \mathbf{f}(\mathbf{x})$, $G_t(\mathbf{x}) = \frac{\mathbf{1}}{(\mathbf{4\pi t})^{\frac{\mathbf{n}}{\mathbf{2}}}} \mathbf{e}^{-\frac{\mathbf{1}}{\mathbf{4t}}(\mathbf{x_1^2}+\ldots+\mathbf{x_n^2})}$ *is the* n*-dimensional Gaussian distribution with variance* $2t$ *(standard deviation* $\sigma = \sqrt{2t}$*) and* $\mathbf{x} = (\mathbf{x_1}, \mathbf{x_2}, \ldots, \mathbf{x_n}) \in \mathbf{R^n}$. *The parameter* t *is named* ***scale****.*

The Gaussian scale-space has several interesting properties, most of them inherited from the Gaussian function. Let's list them briefly.

2.1. Properties

Linearity and Translation Invariance: Since we defined the Gaussian scale-space from a convolution, its dependence on f is necessarily linear and translation invariant.
Isotropy: Since the n-dimensional Gaussian is rotationally invariant or isotropic so is the Gaussian scale-space.
Semigroup: The Gaussian kernels G_t have a semigroup property, that is, $G_{t_1+t_2} = G_{t_1} * G_{t_2}$ for any t_1 and $t_2 \in \mathbf{R}_+$. Therefore, $L(x, t_1 + t_2) = L(x, t_1) * G_{t_2}(x)$, indicating *scale homogeneity*, that is, all scales of L are treated in the same way.
Causality: One way of defining causality is to use a **maximum principle**: broadly speaking, the value of a local maximum (for fixed t) of $L(\cdot, t)$ does not increase as the scale t increases, and the value of a local minimum of $L(\cdot, t)$ does not decrease as the scale t increases. With such definition, the Gaussian scale-space is indeed *causal*.
Heat Equation: The Gaussian scale-space $L(\mathbf{x}, \mathbf{t}) = \mathbf{f} * \mathbf{G_t}(\mathbf{x})$ is the solution of the following partial differential equation, known as the **Heat Equation**

$$\begin{cases} L(\mathbf{x}, \mathbf{0}) = \mathbf{f}(\mathbf{x}) \\ \frac{\partial L(\mathbf{x},\mathbf{t})}{\partial t} = \nabla^2 L(\mathbf{x}, \mathbf{t}) \end{cases}$$

where the Laplacian on the right side is taken only with respect to the $\mathbf{x}$ variables ($\mathbf{x} \in \mathbf{R^n}$).
Uniqueness: The Gaussian scale-space is the only non-trivial space that is linear, isotropic, invariant by translations and satisfies the maximum principle stated above. For a proof, see [2].

3. Discretization via Convolution

How to discretize the Gaussian scale-space and keep the properties we listed in the previous section? As a first approach, we could define a discrete scale-space of a signal by applying any family of filters to it.

Definition 2 *Let g_t be a family of discrete filter kernels (with $g_0[n] = \delta[n] = \delta_{0n}$, the discrete Dirac function). The **discrete scale-space** (according to g_t) of a discrete signal $f : \mathbf{Z} \to \mathbf{R}$ is the function $L : \mathbf{Z} \times \mathbf{R}^+ \to \mathbf{R}$ given by*

$$L[n, t] = f * g_t[n]$$

Note that $L[n, 0] = f[n]$.

The definition for multidimensional signals is similar.

Of course, this has the potential to be useless if the family g_t is not at least continuous on t, not to mention that we would like g_t to have *something* to do with the non-discrete Gaussian function! In this section, we present 3 alternatives for the construction of the family g_t, all of them trying to represent Gaussian scale-spaces.

3.1. Sampled Gaussian

Maybe the most straightforward solution to discretize the Gaussian scale-space is to take the family g_t from a direct uniform sampling of the Gaussian function. For example, in the unidimensional case, we could just calculate the value of the Gaussian function at the integers

$$g_t[n] = \frac{1}{\sqrt{4\pi t}} e^{-\frac{n^2}{4t}}$$

For implementation purposes, since this function has infinite support, we might have to decide on a cut-off point for n. It is customary to take this cut-off point at 3 or 4 standard deviations from the mean, that is, to take

$$g_t[n] = \begin{cases} \frac{C}{\sqrt{4\pi t}} e^{-\frac{n^2}{4t}}, \text{ for } |n| < 4\sqrt{\sigma} = 4\sqrt{2t} \\ 0, \text{ otherwise} \end{cases}$$

where the constant C is taken to make g_t normalized, that is, such that

$$\sum_{n=-\infty}^{\infty} g_t[n] = 1$$

This renormalization is specially relevant for small values of t, when the filter has small support.

Now we can define the scale-space of a 1D signal f as in the previous section[1]

$$L_t[n] = f[n] * g_t[n]$$

where $g_t[\cdot]$ is defined as above. Unfortunately, such scale-space does not have the semi-group nor the causality property.

3.2. Recursive Gaussian (Deriche)

Note that the calculation of the sampled Gaussian scale-space is very costly, specially when its variance is big, due to the computation of the convolutions involved. Deriche [4] shows how to approximate the sampled Gaussian kernel by a *recursive kernel* (that is, one whose convolution can be quickly calculated via recursive relations), namely,

$$\begin{aligned} e^{-\frac{1}{2\sigma^2}x^2} \approx\ & e^{-1.783\frac{x}{\sigma}}\left(1.68\cos\left(0.6318\frac{x}{\sigma}\right) + 3.735\sin\left(0.6318\frac{x}{\sigma}\right)\right) - \\ & e^{-1.723\frac{x}{\sigma}}\left(0.6803\cos\left(1.997\frac{x}{\sigma}\right) + 0.2598\sin\left(1.997\frac{x}{\sigma}\right)\right) = g_{2t}(x) \end{aligned}$$

where $\sigma^2 = 2t$.

More explicitly, let $x : \{0, 1, ..., N\} \to \mathbf{R}$ and consider the kernel $g_{2t}[n]$ given by the sampling of the function above. The discrete convolution $y = x * g_{2t}[n]$

[1] From here on, we slightly change the notation putting the variable t on $L[n, t]$ as a subscript.

can be computed using the following recursive relations:

$$\begin{aligned} y^+[n] &= n_{00}^+ x[n] + n_{11}^+ x[n-1] + n_{22}^+ x[n-2] + n_{33}^+ x[n-3] \quad (1) \\ &\quad -d_{11}^+ y^+[n-1] - d_{22}^+ y^+[n-2] - d_{33}^+ y^+[n-3] - d_{44}^+ y^+[n-4] \\ y^-[n] &= n_{11}^- x[n+1] + n_{22}^- x[n+2] + n_{33}^- x[n+3] + n_{44}^- x[n+4] \quad (2) \\ &\quad -d_{11}^- y^-[n+1] - d_{22}^- y^-[n+2] - d_{33}^- y^-[n+3] - d_{44}^- y^-[n+4] \\ y[n] &= y^+[n] + y^-[n] \quad (3) \end{aligned}$$

where $n_{00}^+, n_{11}^+, n_{22}^+, n_{33}^+, d_{11}^+, d_{22}^+, d_{33}^+, d_{44}^+, n_{11}^-, n_{22}^-, n_{33}^-, n_{44}^-, d_{11}^-, d_{22}^-, d_{33}^-$ and d_{44}^- are constants that can be quickly calculated depending only on the particular σ chosen – for example, $n_{00}^+ = 2.3603$ and $d_{44}^+ = e^{-7.012/\sigma}$. For all other expressions, see [4].

Note that y^+ (from formula 1) can be calculated recursively from left to right ($n = 0, 1, ..., N$), while y^- (formula 2) can be computed from right to left ($n = N, N-1 \cdots, 0$). In order to compute each element of $y[n]$, we need 15 additions and 16 multiplications, independent of σ, what makes this method very efficient computationally.

One remaining difficulty is: how to initialize the method? For example, how to choose $y^+[-1]$, $y^+[-2]$, $y^+[-3]$ and $y^+[-4]$ for the first left-to-right iteration, and the corresponding $x[-1], x[-2], x[-3], x[-4]$? We discuss this implementation detail a bit more carefully in [1].

Similarly to the 1D case, we extend this *recursive scale-space* to the multi-dimensional case using tensor products.

3.3. Splines

We could also use splines to construct a discrete scale-space. We will briefly describe this possibility in this subsection. For more details on this possibility (including the formulas we present below), see [1], [5] or [6].

Since B-splines are very similar in shape to Gaussian functions[2], one could use them to construct a scale-space. Besides, B-splines have the advantage of having finite support.

Definition 3 *A (non-discrete)* spline scale-space of order n_2 *of a function* f *is defined by*

$$L(x,t) = f(x) * \beta_t^{n_2}(x)$$

where $\beta_t^{n_2}$ *is a B-spline of order* n_2 *rescaled by a factor of* t*, that is,*

$$\beta_t^n(x) = \frac{1}{t}\beta^n(\frac{x}{t})$$

Sometimes, the function f is given (or is approximately decomposed) in a basis formed by translated B-Splines of some determined order n_1:

$$f(x) = \sum_{k \in \mathbf{Z}} c[k]\beta^{n_1}(x-k)$$

[2] It is worth to point out that this would be true for almost any chosen β^0; this is just one of the simplest choices.

In this case, if $x = k \in \mathbf{Z}$ and $t = \frac{m_1}{m_2} \in \mathbf{Q}_+$ (with $m_1,\, m_2 \in \mathbf{N}^*$), we can compute $L\,(x,t)$ exactly using (see appendix A of [6])

$$L_t\,[k] = m_2 \left(b^{n_1+n_2+1} * B^{n_1}_{m_2} * B^{n_2}_{m_1} * c_{\uparrow m_2}\right)_{\downarrow m_2}[k] \tag{4}$$

where the arrows represent upsampling and downsampling of a signal, b^n is the integer sampling of the B-spline of order n and B^n_m is a **discrete** spline of order n dilated by a factor of m.

For comparison purposes, we consider $n_1 = n_2 = 3$ (cubic B-splines both for the representation of f and for the construction of the scale-space); summarizing, we consider

$$L_t\,[k] = m_2 \left(b^7 * B^3_{m_2} * B^3_{m_1} * c_{\uparrow m_2}\right)_{\downarrow m_2}[k] \tag{5}$$

where:

– $k \in \mathbf{Z}$ and $t = \frac{m_1}{m_2}$ is a rational scale $(m_1, m_2 \in \mathbf{N})$
– $c\,[k]$ are the coefficients of the signal when written in a cubic B-spline basis.
– b^7 is the sampling of the B-spline of order 7, namely

$$b^7 = \left[..., 0, \frac{1}{5040}, \frac{1}{42}, \frac{397}{1680}, \frac{151}{315}, \frac{397}{1680}, \frac{1}{42}, \frac{1}{5040}, 0, ...\right]$$

– B^3_m is the discrete spline of order 3 dilated by a factor m,

If the scale t is an integer ($m_2 = 1$), the spline scale-space can be computed efficiently since 5 is simplified to

$$L_{m_1}\,[k] = \left(b^7 * B^3_{m_1} * c\right)[k]$$

Once more, the multidimensional case is obtained using tensor products.

4. Discretization via Heat Equation

As we have seen, the Gaussian scale-space can be defined as the solution of the heat equation $\frac{\partial L(\mathbf{x},\mathbf{t})}{\partial t} = \nabla^2 L(\mathbf{x},\mathbf{t})$. This suggests a discrete scale-space based on the discretization of this equation instead of a direct discretization of the convolution in the solution. In fact:

Theorem 4 *The discrete heat equation*

$$\frac{\partial L_t\,[\mathbf{n}]}{\partial t} = \mathcal{A} * L_t\,[\mathbf{n}] \tag{6}$$

(where $\mathcal{A}$ is a multiple of the discrete Laplacian operator) defines the only scale-space that is linear, symmetric, translation-invariant, has the semigroup property and satisfies the maximum principle.

Therefore, $\mathcal{A}$ must have the form$\mathcal{A} = K\,[1, -2, 1]$ for one-dimensional scale-spaces, while

$$\mathcal{A}_c = K \begin{bmatrix} c & 1-2c & c \\ 1-2c & -4+4c & 1-2c \\ c & 1-2c & c \end{bmatrix} \qquad 0 \le c \le \frac{1}{2}$$

for the two-dimensional case. It is not hard to see that $\mathcal{A}$ is a multiple of a discrete approximation for the Laplacian operator.

Since the multiplicative constant K corresponds to a simple rescaling of the t parameter, it is common to take $K = \frac{1}{h^2}$ (where h is the distance between grid points) or, more simply, to take $K = 1$. The c parameter, on the other hand, cannot be determined without further constraints; note that c measures somewhat the "importance" of the diagonals of the matrix $\mathcal{A}$. It is common to take $c = 0$ for the simplest discretization of the Laplacian, that has the advantage of satisfying the maximum principle even using 4-connectivity. However, for a more "isotropic" Laplacian, we prefer $c = \frac{1}{6}$.

Whatever the choice of $\mathcal{A}$ is, we call it the *infinitesimal generator* of the scale-space. Based on this continuous scale, discrete space equation, we can come up with some other ideas for discrete scale-spaces.

4.1. First-order generator

We could apply a first-order method in t to solve equation 6; for example, in the 2D case, we take a small step Δt and write

$$L_{t+\Delta t}[m,n] \approx (\delta + \mathcal{A}\Delta t) * L_t[m,n]$$

where δ is the discrete unitary impulse.

Lindeberg [3] suggests then a discrete scale space computed by powers of a *first-order generator* $\delta + \mathcal{A}\Delta t$:

$$L_{t_0+n\Delta t}[m,n] \approx \underbrace{(\delta + \mathcal{A}\Delta t) * (\delta + \mathcal{A}\Delta t) * \cdots * (\delta + \mathcal{A}\Delta t)}_{n \text{ times}} * L_{t_0}[m,n]$$

Using the discrete 2D Laplacian above, it can be shown (like in [3]) that

- $\delta + \mathcal{A}_c\Delta t$ is unimodal only for $c \leq \frac{1}{4}$;
- $\delta + \mathcal{A}_c\Delta t$ it is separable only when $c = \frac{\Delta t}{2}$;
- The causality (maximum principle) is satisfied only if $0 \leq c \leq \frac{1}{4}$.
- $\delta + \mathcal{A}_c\Delta t$ has maximum "rotational symmetry" when $c = \frac{1}{6}$ ([3], ch. 4).

4.2. Crossed Convolutions

While all the scale-spaces presented so far approximate the original Gaussian scale-space, none of them solve exactly equation 6, and therefore none of them have all the properties that such solution would have. Moreover, it is really not absolutely necessary to use first-order approximations for equation 6: one can solve it explicitly. For example, in the 1D case,

$$\begin{aligned} \frac{\partial L_t[n]}{\partial t} &= [1,-2,1] * L_t[n] \\ L_0[n] &= f[n] \end{aligned}$$

can be explicitly solved using a Z-transform (see [2] or [3]). The solution is

$$L_t[n] = f * P_t[n]$$

where $P_t[n]$ is a *symmetric Poisson kernel*[3], that is,

$$P_t[n] = e^{-2t} I_n(2t) = e^{-2t}(\cdots, I_{-n}(2t), \cdots, I_0(2t), \cdots, I_n(2t), \cdots)$$

where $I_n(t)$ is the modified Bessel function[4] of order n.

Note that this is the **only 1D discrete scale-space** with the properties of linearity, invariance by translations, semigroup, symmetry and causality (given by the maximum principle). Also, it is not hard to show that $P_t[n]$ has variance $2t$, the same variance of the Gaussian $G_t(x)$. For these (and other) reasons, it is common to call $P_t[n]$ the *discrete Gaussian distribution*.

For 2D signals, we could just use tensor products. However, can we write an explicit formula for the solution of the discrete heat equation for other discretizations of the Laplacian? The answer is positive:

Theorem 5 *The solution of the discrete 2D heat equation*

$$\begin{aligned} \frac{\partial L_t[m,n]}{\partial t} &= \mathcal{A}_c * L_t[m,n] \\ L_0[m,n] &= f[m,n] \end{aligned} \tag{7}$$

can be written as

$$L_t[m,n] = P^x_{(1-2c)t} * P^y_{(1-2c)t} * P^{x+y}_{ct} * P^{x-y}_{ct} * f[m,n]$$

where

$$\begin{aligned} P^x_\alpha[m,n] &= P_\alpha[m]\,\delta[n];\ P^y_\alpha[m,n] = P_\alpha[n]\,\delta[m] \\ P^{x+y}_\alpha[m,n] &= P_\alpha[m]\,\delta[m-n];\ P^{x-y}_\alpha[m,n] = P_\alpha[m]\,\delta[m+n] \end{aligned}$$

For example, P^x_α is the symmetric Poisson kernel of variance 2α spread only in the "x direction", while P^{x+y}_α is the same symmetric Poisson kernel but this time spread only in the direction $x = y$ (the 45^0 diagonal), with zeroes elsewhere.

Proof: Let

$$g_t(x,y) = \sum_{j,k=-\infty}^{\infty} L_t[j,k]\, x^j y^k$$

be the Z-transform of L_t. Then, from 7

$$\frac{\partial g_t(x,y)}{\partial t} = g_t(x,y).$$
$$t\left\{c\left(xy + xy^{-1} + x^{-1}y + x^{-1}y^{-1}\right) + (1-2c)\left(x + x^{-1} + y + y^{-1}\right) - 4 + 4c\right\}$$

[3] So called because it corresponds to the difference of two independent random variables with the same Poisson distribution.

[4] One convenient way to define the modified Bessel function is to look at the coefficients of the expansion of $e^{\frac{\alpha}{2}\left(z+z^{-1}\right)}$ in a Laurent power series in z, that is $e^{\frac{\alpha}{2}\left(z+\frac{1}{z}\right)} = \sum_{k=-\infty}^{\infty} I_n(\alpha)\, z^n$. A subroutine for the calculation of modified Bessel functions can be found in [7].

We solve this and rearrange the terms conveniently:

$$\begin{aligned} g_t(x,y) &= e^{t\left\{c\left(xy+xy^{-1}+x^{-1}y+x^{-1}y^{-1}\right)+(1-2c)\left(x+x^{-1}+y+y^{-1}\right)-4+4c\right\}} g_0(x,y) = \\ &= e^{(1-2c)t\left(x+\frac{1}{x}-2\right)} e^{(1-2c)t\left(y+\frac{1}{y}-2\right)} e^{ct\left(xy+\frac{1}{xy}-2\right)} e^{ct\left(\frac{x}{y}+\frac{y}{x}-2\right)} g_0(x,y) \end{aligned}$$

Now it is easy to see that each of the 4 exponentials is the Z-transform of one of the symmetric Poisson kernels in the solution (just expand them in Laurent power series and use the definition of the modified Bessel functions).

5. Comparative Analysis

In this section we make a comparative analysis between the discretization methods both qualitatively and quantitatively.

5.1. Theoretical Properties

The table below summarizes the properties of the considered scale-spaces

	semigroup	causality	scale parameter in
sampled Gaussian			$\mathbf{R}$
recursive Gaussian			$\mathbf{R}$
splines			$\mathbf{Q}$
first-order generator	$\surd$	$\surd$	$\Delta t\mathbf{Z}$
crossed convolutions	$\surd$	$\surd$	$\mathbf{R}$

All these methods try to approximate the continuous Gaussian scale-space in different ways. Note that both the sampled gaussian method and the crossed convolutions method incur in additional numerical errors by performing a kernel cut-off to avoid infinite supports; the first-order generator method trades this infinite support by the approximation $L_{t+\Delta t}[m,n] \approx (\delta + \mathcal{A}\cdot\sqcup) * L_t[m,n]$; the recursive gaussian method and the spline method approximate the Gaussian kernel by other kernels that can be computed quicker.

5.2. Computational Efficiency

In the table below we show the algorithmic time and the runtime in seconds of the calculation of each scale-space for the indicated variances. While these values were taken from a Pentium II 300MHz, they are a good basis of comparison between different algorithms. All of them show that calculation of scale-spaces can be quite costly.

	$O(\cdot)$	var=1.0	var=3.0	var=27.0	var=243.0
s. Gaussian	$8\sigma N$	0.22	0.27	0.67	1.87
r. Gaussian	$16N$	$0.57-0.15$	$0.57-0.15$	$0.57-0.15$	$0.57-0.15$
splines	$44Nd$	63.0	0.4	0.43	0.5
1 ord. gen.	$10\sigma^2 N$	0.4	0.95	7.5	67.6
c. conv.	$16\sigma N$	$0.22-0.36$	$0.27-0.4$	$0.67-0.86$	$1.87-2.37$

The first column indicates the number of multiplications for the calculation of each convolution, where N is the number of pixels in the image, σ is the standard deviation of the filter kernel and d is the denominator of the scale $\frac{n}{d}$ in the spline case.

In particular, since spline scales are always rationals of the form $\frac{n}{d} = \sqrt{3.variance}$, for the variances above we used scales $\sqrt{3} \approx \frac{173}{100}, 3, 9$ and 27. Note the extreme influence of the denominator on the spline convolution running time.

For the recursive Gaussian, we show both the times for convolutions with periodic extension and with zero extension; the times indicate that the initialization step takes longer than the recursion itself.

For the crossed convolutions, we display running times for $c = 0$ and $c = \frac{1}{6}$. Note that, for $c = 0$, the times are the same as the sampled Gaussian times (since there are only two convolutions to be performed in this case).

Note also that, since the first-order generator method and the crossed convolutions method both have the semigroup property, we could calculate scale-spaces for larger scales using previously calculated smaller scales – this reduces running time for these algorithms when calculating several scales.

5.3. Numerical Precision

In the table below, we display the squared distance between images for several variances and pairs of methods (the image used in these comparisons was Lenna's image). Signal extensions are by mirroring across borders unless stated otherwise.

			var	*dist*
1	Sampled Gaussian	Crossed Conv. ($c = 0$)	$3,0$	$0,115$
2	Sampled Gaussian	Crossed Conv. ($c = 0$)	$27,0$	$0,003$
3	Sampled Gaussian	Recursive Gaussian	$3,0$	$0,00005$
4	Sampled Gaussian	Recursive Gaussian	$27,0$	$0,00005$
5	Sampled Gaussian	Splines	$3,0$	$0,100$
6	Crossed Conv. ($c = 0$)	Splines	$27,0$	$0,265$
7	Crossed Conv. ($c = 0$)	1st-Order Gen. ($c = 0$)	$27,0$	$0,0044$
8	Crossed Conv. ($c = 0$)	1st-Order Gen. ($c = \frac{1}{6}$)	$27,0$	$0,0029$
9	1st-Order Gen. ($c = \frac{1}{6}$)	Crossed Conv. ($c = \frac{1}{6}$)	$27,0$	$5,074$

We note that all scale-spaces calculated are very close to each other; in particular, the recursive method is an incredible approximation of the sampled Gaussian method. The only case in which two methods show some difference is the crossed convolutions method and the first-order generator method for $c = \frac{1}{6}$.

6. Final remarks

The comparative results in the previous section show that the crossed convolutions method is the only one that retains all the properties of the continuous case. Since its computational cost is still acceptable, it is our personal favorite scale-space.

On the other hand, the recursive Gaussian method gives close results with a runtime that is independent of the chosen variance and should be kept handy specially for large variances.

The spline method, while fast for some scales, does not seem to be appropriate for heavy sampling of scale-spaces; actually, any scale with high denominator greatly slows down the method.

The sampled Gaussian method, while presenting similar results to the others and similar running times, seems to us to be overshadowed by the crossed convolutions method, that has the same running times and several theoretical properties in its favor.

First-order generators take longer and correspond only to a first-order approximation of the solution of the discrete heat equation. Besides, its theoretical properties of causality and semigroup breakdown whenever we need scales that are not multiples of the predefined scale step.

Finally, we must note that all the convolutions could be computed using Discrete Fourier Transforms (FFT algorithms); this might not only speed up all algorithms but, in some cases, avoid numerical errors that come from truncating the supports of the used kernels.

References

1. Anderson Mayrink da Cunha, *Espaços de Escala e Detecção de Arestas*, M.Sc. Dissertation, IMPA, Rio de Janeiro, October 2000. `http://www.visgraf.impa.br/escala.html`
2. Ralph Costa Teixeira, *Introdução aos Espaços de Escala*, Escola de Computação 2000, IME-USP, São Paulo, July 2000. `http://www.visgraf.impa.br/Courses/eescala/index.html`
3. Tony Lindeberg, *Scale Space Theory in Computer Vision.* Kluwer Academic Publishers, Dordrecht, The Netherlands, 1994
4. Rachid Deriche, *"Recursively implementing the gaussian and its derivatives,"* Tech. Report 1893, Programme 4 - Robotique, Image et Vision, INRIA - Institut National en Informatique et en Automatique, April 1993.
5. Michael Unser, "*Splines: a perfect fit for signal and image processing,"* IEEE Signal Processing Magazine, vol. 16, no. 6, pp. 22-38, November 1999. `http://bigwww.epfl.ch/publications/unser9902.html`
6. Yu-Ping Wang and S. L. Lee, *"Scale-Space Derived From B-Splines,"* IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 20, no. 10, pp. 1040-1055, October 1998. `http://wavelets.math.nus.edu.sg/~wyp/download_papers/Preprints.html`
7. William H. Press et al., *Numerical Recipes in C: the art of scientific computing.* second ed., Cambridge University Press, New York, 1992.
8. Florack, L., *"A Spatio-Frequency Trade-Off Scale for Scale-Space Filtering"*, IEEE PAMI v. 22 pp.1050-1055, 2000.