# Discrete Scale Spaces

Anderson Cunha, Ralph Teixeira and Luiz Velho

January 28th, 2001

**Abstract**

Scale spaces allow us to organize, compare and analyse differently sized structures of an object. The linear scale space of a monochromatic image is the solution of the heat equation using that image as an initial condition. Alternatively, this linear scale space can also be obtained applying Gaussian filters of increasing variances to the original image. In this work, we compare (by looking at theoretical properties, running time and output differences) five ways of discretizing this 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". In particular, we explicitly present a correct way of initializing the recursive method to approximate Gaussian 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. For example, in cartography, if we want to find a house in a neighborhood, the scale 1:10.000 is appropriate. However, if we want the location of this same spot in the planet, we have to use a 1:1.000.000 scale.

An appropriate choice of scale is also imperative to locate phenomena in time. Certain physical phenomena are observable in time scales of micro-seconds (think of the information flow inside a computer), while others occur in time scale of billions of years (like the life cycle of stars).

Note that the concept of scale is closely linked to the idea of *resolution* (or detail). In general, the resolution is the inverse of scale: whenever we examine events of small (fine) scale, we use high resolutions.

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. In image processing, several such *multi-scale representations* have been created to face this difficulty, like Quad-Trees, Pyramids, Wavelets and Scale-Spaces.

Some desirable properties for a multi-scale representation are:

- Isotropy or rotation invariance: there are no preferred directions, that is, the representation of a structure is independent of its orientation.
- Homogeneity or translation invariance: there are no preferred locations, that is, the representation of a structure does not depend on its localization.
- Causality: no structure is created when the scale increases. More specifically, any feature present in a certain scale comes from details that are present in smaller scales; the representation of an object in a coarser scale has no more details than the same object represented on a smaller (finer) scale.

Note that these properties (specially the invariance by translations) imply that the size of the support of an object can't change with scale; this is unlike the idea of scale in cartographic maps, where a larger scale usually implies in smaller object support. 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, that forces the image to lose detail as the scale increases.

The structure of this paper goes as follows:

In section 2 we will formally define (non-discrete) linear (Gaussian) scale-spaces and present their main properties.

In section 3, we will discretize such linear scale-spaces by discretizing the Gaussian filter kernel in different ways; in particular, we present a correct way of initializing an algorithm by Deriche [4] to recursively calculate scale-spaces when the image extension is assumed to be periodic.

In section 4, we discretize linear scale-spaces from a slightly different point-of-view: we discretize a Heat Equaton, always keeping an eye to see if the non-discrete properties transfer to the discrete case. In particular, we present a new method (Crossed Convolutions) that solves the discrete version of the heat equation exactly, and therefore preserves many of the desirable properties of the non-discrete case.

Finally, in section 5, we compare the results of our implementations of the different methods (looking at theoretical differences, running time and output differences of the many scale-spaces).

# 2 Gaussian Scale-Spaces

**Definition 1** *Let $f : \mathbb{R}^n \to \mathbb{R}$. The **Gaussian scale-space** of $f$ is the function $L : \mathbb{R}^n \times \mathbb{R}^+ \to \mathbb{R}$ given by*

$$L(\mathbf{x}, t) = f * G_t(\mathbf{x})$$

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

Figure 1 shows samples of Lenna's image scale-space at scales $t = 0, 0.25, 0.5, 1, 2, 4, 8, 16, 32$ (note that $t = 0$ corresponds to the original image).

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

## 2.1 Properties

### 2.1.1 Linearity and Translation Invariance

Since we defined the Gaussian scale-space from a convolution, its dependence on $f$ is necessarily linear and translation invariant.

### 2.1.2 Isotropy

Since the $n$-dimensional Gaussian is rotationally invariant or isotropic (see figure 2), so is the Gaussian scale-space.

Figure 1: Lenna's image scale-space.

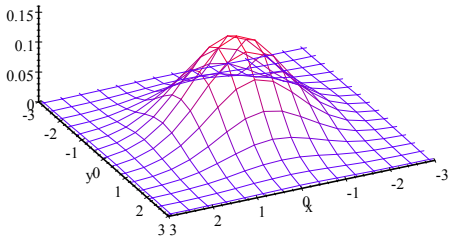


Figure 2: 2D Gaussian, $t = 0.5$.

### 2.1.3 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 $t_2 \in \mathbb{R}_+$. Therefore, $L(x, t_1 + t_2) = L(x, t_1) * G_{t_2-t_1}(x)$, indicating *scale homogeneity*, that is, all scales of $L$ are treated in the same way.

### 2.1.4 Causality

The causality property requires that the scale-space be somehow smoothed with increasing scale, but it can be formalized in many ways. 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*.

### 2.1.5 Heat Equation

The Gaussian scale-space $L(\mathbf{x}, t) = f * G_t(\mathbf{x})$ is the solution of the following partial differential equation, known as the **Heat Equation**

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

where the Laplacian on the right side is taken only with respect to the $\mathbf{x}$ variables ($\mathbf{x} \in \mathbb{R}^n$).

### 2.1.6 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 $f : \mathbb{Z} \to \mathbb{R}$ is the function $L : \mathbb{Z} \times \mathbb{R}^+ \to \mathbb{R}$ given by*

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

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

The definition for multidimensional signals is similar. For example, for images we have:

**Definition 3** *Let $g_t$ be a family of discrete filter kernels (with $g_0[m, n] = \delta[m, n]$). The **discrete scale-space** (according to $g_t$) of an image $f : \mathbb{Z}^2 \to \mathbb{R}$ is the function $L : \mathbb{Z}^2 \times \mathbb{R}^+ \to \mathbb{R}$ given by*

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

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

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 (if possible, keeping the properties described in the continuous case).

### 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. In particular, note that, when $t \to 0$, $G_t(0) \to \infty$ and we don't want the same behavior for $g_t[0]$. With the renormalization, $g_t[0] = 1$ for $t < 1/32$ (in fact, $g_t[n] = \delta[n]$ for $t < 1/32$).

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 (both properties are defined in the discrete case following the corresponding non-discrete definitions).

We extend the sampled Gaussian scale-space to the multi-dimensional case using tensor products, that is, using

$$g_t[n_1, n_2, ...] = g_t[n_1] . g_t[n_2] ....$$

where each of the $g_t$ on the right side is defined according to the previous discussion.

Figure 3 shows samples of this discrete scale-space at variances 0 (original image), 1, 10 and 100.

### 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

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

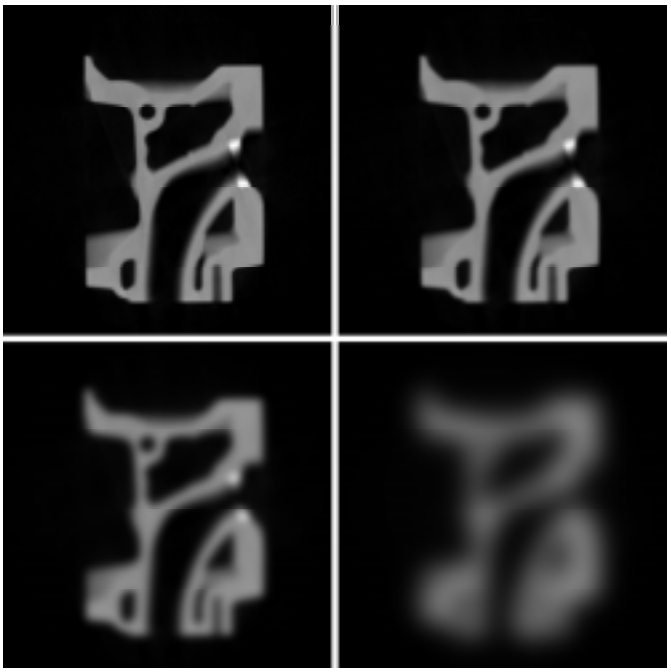

Figure 3: Sampled Gaussian scale-space.

calculated via recursive relations), namely, using the approximation

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

where $\sigma^2 = 2t$.

More explicitly, let $x : \{0, 1, ..., N\} \to \mathbb{R}$ and consider the kernel $g_{2t}\left[n\right]$ obtained using the approximation above instead of $e^{-n^2/4t}$. The discrete convolution $y = x * g_{2t}\left[n\right]$ can be quickly computed using the following recursive relations:

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

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

$$y\left[n\right] = y^+\left[n\right] + y^-\left[n\right] \tag{3}$$

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 $\sigma$ 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 $y\left[n\right]$, we need 15 additions and 16 multiplications, independent of $\sigma$, 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 the next subsection.

Similarly to the 1D case, we extend this *recursive scale-space* to the multi-dimensional case using tensor products. Figure 4 shows samples of this scale-space at variances 0, 1, 10 and 100.

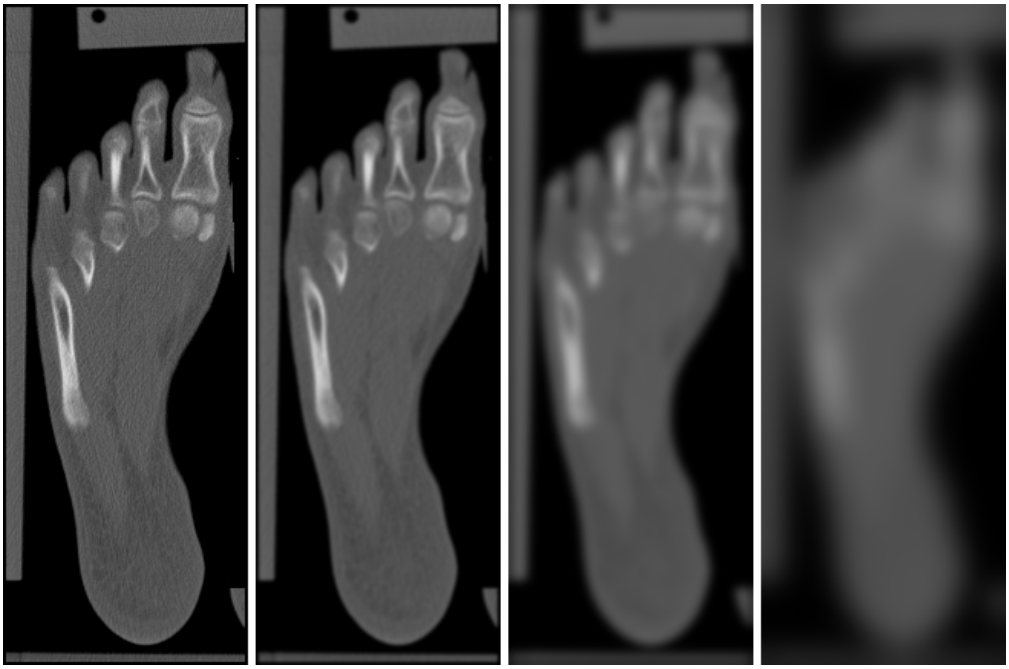

Figure 4: Recursive Gaussian Scale-space

### 3.2.1 Recursion Initialization

As we mentioned above, it is not clear at all how to start the recursion for Deriche's method. It seems to be common to choose $y^+[-1] = y^+[-2] = y^+[-3] = y^+[-4] = 0$, but, unless a renormalization is done at the end of the process, this has the effect of making the whole image darker (closer to 0) at every convolution performed.

Nevertheless, there is an alternative for the definition of $y^+$ if we assume that the finite signal $x$ is extended periodically. Indeed, just note that the answers $y^+$ and $y^-$ must also be periodic with the same period of $x$. So, if the period is assumed to be $T$, we would like to have

$$\begin{cases} y^+[T] = y^+[0] \\ y^+[T+1] = y^+[1] \\ y^+[T+2] = y^+[2] \\ y^+[T+3] = y^+[3] \end{cases}$$

On the other hand, successive applications of the formula 1 lead to a relation that looks like

$$\begin{cases} y^+[T] = a_{00}y^+[0] + a_{01}y^+[1] + a_{02}y^+[2] + a_{03}y^+[3] + A_0 \\ y^+[T+1] = a_{10}y^+[0] + a_{11}y^+[1] + a_{12}y^+[2] + a_{13}y^+[3] + A_1 \\ y^+[T+2] = a_{20}y^+[0] + a_{21}y^+[1] + a_{22}y^+[2] + a_{23}y^+[3] + A_2 \\ y^+[T+3] = a_{30}y^+[0] + a_{31}y^+[1] + a_{32}y^+[2] + a_{33}y^+[3] + A_3 \end{cases} \tag{4}$$

where the coefficients $A_0 \cdots A_3$ depend on the input signal $x$ and on the constants $n_{00}^+ \cdots n_{33}^+, d_{11}^+ \cdots d_{44}^+$, while the values of $a_{ij}$ depend *only* on $d_{11}^+ \cdots d_{44}^+$ and $T$, that is, depend only on $\sigma$ and the size of $x$.

How to quickly find such coefficients? Well, for $A_0$, $A_1$, $A_2$ and $A_3$ it is enough to "simulate" the process with $y^+[0] = y^+[1] = y^+[2] = y^+[3] = 0$ (using the periodic extension for $x$) and apply formula 1 repeated times. According to system 4, for this "simulation" we must end up with $A_0 = y^+[T]$, $A_1 = y^+[T+1]$, $A_2 = y^+[T+2]$ and $A_3 = y^+[T+3]$.

On the other hand, system 4 also says that, if we take $x[n] = 0, \forall n$, $y^+[0] = 1$ and $y^+[1] = y^+[2] = y^+[3] = 0$ and apply recursion law 1 repeated times we end up finding $a_{00} = y^+[T]$, $a_{10} = y^+[T+1]$, $a_{20} = y^+[T+2]$ and $a_{30} = y^+[T+3]$. All other coefficients can be computed similarly; for example, take $x \equiv 0$, $y^+[0] = y^+[1] = y^+[2] = 0$ and $y^+[3] = 1$, apply recursion 1 repeatedly until you find $a_{i3} = y^+[T+i]$.

Once we have all coefficients on the linear system 4, we now take into account the periodicity of $y^+$ and solve a simple linear system to find the correct values of $y^+[0]$, $y^+[1]$, $y^+[2]$ and $y^+[3]$, namely,

$$\begin{cases} (a_{00}-1)y^+[0] + a_{01}y^+[1] + a_{02}y^+[2] + a_{03}y^+[3] = -A_0 \\ a_{10}y^+[0] + (a_{11}-1)y^+[1] + a_{12}y^+[2] + a_{13}y^+[3] = -A_1 \\ a_{20}y^+[0] + a_{21}y^+[1] + (a_{22}-1)y^+[2] + a_{23}y^+[3] = -A_2 \\ a_{30}y^+[0] + a_{31}y^+[1] + a_{32}y^+[2] + (a_{33}-1)y^+[3] = -A_3 \end{cases}$$

The initial values $y^-[N]$, $y^-[N-1]$, $y^-[N-2]$ and $y^-[N-3]$ for recursion 2 are calculated similarly. This guarantees that the answers $y^+$ and $y^-$ are both periodic of period $T$, as we wanted.

Finally, if the signal $x$ is extended with zeroes (therefore, a non-periodic extension), there is no way out, and we do take indeed $y^+[-1] = y^+[-2] = y^+[-3] = y^+[-4] = 0$ for the calculation of $y^+[i]$ and $y^-[N+1] = y^-[N+2] = y^-[N+3] = y^-[N+4] = 0$ for the calculation of $y^-[i]$. Note that these are not guaranteed to make $y$ a signal with zero extension (there would be a "leak" and some values of $y[n]$ would be non-zero for values of $n$ out of $x$ support).

### 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].

**Definition 4** *A* B-Spline of order $n$ *is defined by*

$$\beta^n(x) = \underbrace{\beta^0(x) * \beta^0(x) * \cdots * \beta^0(x)}_{n \text{ convolutions}}$$

*where* $\beta^0(x)$ *is a unitary square pulse function:*

$$\beta^0(x) = \begin{cases} 1, & \text{if } -\frac{1}{2} \leq x < \frac{1}{2} \\ 0, & \text{otherwise} \end{cases}$$

Since B-splines are very similar in shape to Gaussian functions[2] (figure 5 show the similarity already for order 3), one could use them to construct a scale-space. Besides, B-splines have the advantage of having finite support.

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

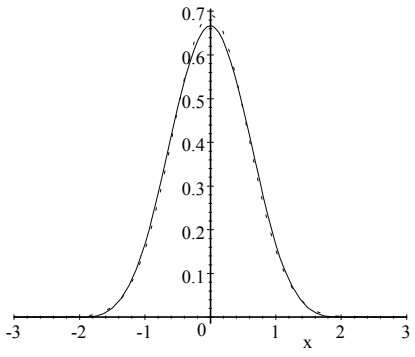


Figure 5: A Gaussian function (dotted line) and the cubic B-spline.

**Definition 5** *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 $n_1$:

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

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

$$L_t[k] = m_2\left(b^{n_1+n_2+1} * B_{m_2}^{n_1} * B_{m_1}^{n_2} * c_{\uparrow m_2}\right)_{\downarrow m_2}[k] \tag{5}$$

where the arrows represent upsampling and downsampling of a signal, $b^n$ is the integer sampling of the B-spline of order $n$ and $B_m^n$ 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

$$L_t[k] = m_2\left(b^7 * B_{m_2}^3 * B_{m_1}^3 * c_{\uparrow m_2}\right)_{\downarrow m_2}[k] \tag{6}$$

where:

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

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

- $B_m^3$ is the discrete spline of order 3 dilated by a factor $m$, that is,

$$B_m^3 = B_m^0 * B_m^0 * B_m^0 * B_m^0$$
$$B_m^0 = \frac{1}{m}\underbrace{[1, 1, ..., 1]}_{m}$$

Figure 6 gives a block diagram for the computation of $L_t[k]$ offered by formula 6

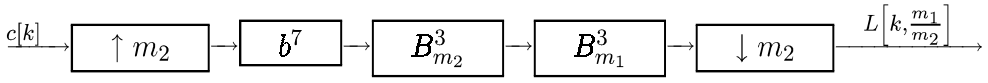


Figure 6: Block diagram for Spline Scale-spaces

If the scale $t$ is an integer ($m_2 = 1$), the spline scale-space can be computed efficiently since formula 6 is simplified to

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

Once more, the multidimensional case is obtained using tensor products (that is, applying the convolution separately in each coordinate direction).

Figure 7 show the spline scale-space of Lenna's image on scales $t = 0, \frac{1}{4}, \frac{1}{2}, 1, 2, 4, 8, 16$ and 32. Note that the cubic B-spline $\beta_t^3$ has variance $\frac{t^2}{3}$; therefore, the corresponding variances in figure 7 are $0, \frac{1}{48}, \frac{1}{12}, \frac{1}{3}, \frac{4}{3}, \frac{16}{3}, \frac{64}{3}, \frac{16^2}{3}$ and $\frac{32^2}{3}$.

# 4 Discretization via Heat Equation

As we have seen in the previous section, the Gaussian scale-space can be defined as the solution of the heat equation $\frac{\partial L(\mathbf{x},t)}{\partial t} = \nabla^2 L(\mathbf{x}, t)$. 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 6** *The discrete heat equation*

$$\frac{\partial L_t[\mathbf{n}]}{\partial t} = \mathcal{A} * L_t[\mathbf{n}] \tag{7}$$

*(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.*

**Proof.** See chapter 10 of [2] or chapter 4 of [3]. Here we present just the general idea of the proof. First, we note that the semigroup property is basically equivalent to ask the scale-space to be given by some differential equation of the form

$$\frac{\partial L_t[\mathbf{n}]}{\partial t} = \mathcal{F}\{L[\cdot], \mathbf{n}\}$$

The invariance by translations practically eliminates the dependence on $\mathbf{n}$ on the right side; together with the linearity, we can already see that the equation must have a convolution form

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

Figure 7: Spline scale-space

The maximum principle forces the operator $\mathcal{A}$ to have several properties; in particular, it has to be local (for example, in $\mathbb{Z}^2$, using 8-connectivity, $\mathcal{A}$ must be a $3x3$ kernel), its center coefficient is negative and all others are positive, and the sum of its coefficients is 0. Finally, the symmetry condition forces $\mathcal{A}$ to be symmetric as well. Summarizing, $\mathcal{A}$ must have the form

$$\mathcal{A} = K\,[1, -2, 1]$$

for one-dimensional scale-spaces, while

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

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

$$\mathcal{A}_0 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

that has the advantage of satisfying the maximum principle even using 4-connectivity. However, for a more "isotropic" Laplacian, we prefer $c = \frac{1}{6}$:

$$\mathcal{A}_{\frac{1}{6}} = \frac{1}{6}\begin{bmatrix} 1 & 4 & 1 \\ 4 & -20 & 4 \\ 1 & 4 & 1 \end{bmatrix}$$

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 7; for example, in the 2D case, we take a small step $\Delta t$ and write

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

where $\delta$ is the discrete unitary impulse

$$\delta = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

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 $c = \frac{1}{6}$ (see chapter 4 of [3]).

Figure 8 show samples of this scale-space (using $c = \frac{1}{6}$) for $t = 0$, 0.5, 5 and 50.

### 4.2 Crossed Convolutions

While all the scale-spaces presented so far approximate the original Gaussian scale-space, none of them solve exactly equation 7, 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 7: one can solve it explicitly.

For example, in the 1D case,

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

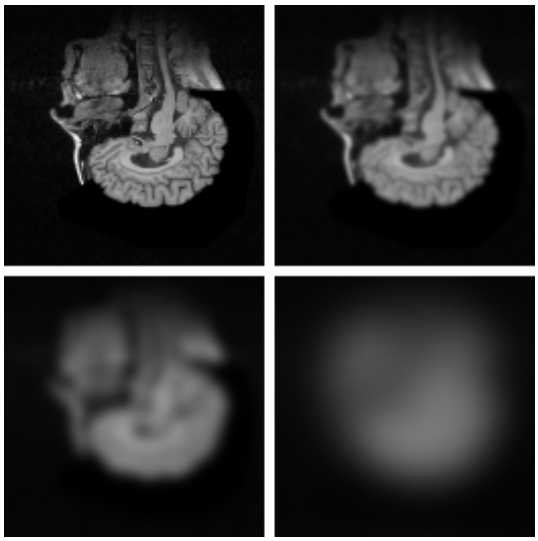

Figure 8: First-order generator scale-space

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

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

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; then, we would have a scale-space defined by

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

that is indeed the solution of

$$\frac{\partial L[m,n]}{\partial t} = \mathcal{A}_0 * L[m,n]$$

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:

[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 $e^{\frac{\alpha}{2}(z+z^{-1})}$ in a Laurent power series in $z$, that is

$$e^{\frac{\alpha}{2}(z+\frac{1}{z})} = \sum_{k=-\infty}^{\infty} I_n(\alpha) z^n$$

A subroutine for the calculation of modified Bessel functions can be found in [7].

**Theorem 7** *The solution of the discrete 2D heat equation*

$$\begin{aligned} \frac{\partial L_t[m,n]}{\partial t} &= \mathcal{A}_c * L_t[m,n] \\ L_0[m,n] &= f[m,n] \end{aligned} \tag{8}$$

*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, $P^x_\alpha$ is the symmetric Poisson kernel of variance $2\alpha$ spread only in the "x direction", while $P^{x+y}_\alpha$ 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 8

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

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). ■

Therefore, we can compute $L_t[m,n]$ in 4 steps:

- Convolve $f[m,n]$ with $P^x_{(1-2c)t}$; this is the same as convolving each line of $f$ with the 1D Poisson kernel $P_{(1-2c)t}$;
- Convolve each column of the result with $P_{(1-2c)t}$;
- Convolve each $45^0$ diagonal of the result with $P_{ct}$;
- Convolve each $-45^0$ diagonal of the result with $P_{ct}$.

In particular, the case $c = 0$ gives the old tensor product $L_t[m,n] = f[m,n] * P^x_t[m] * P^y_t[n]$.

Figure 9 shows samples of the Crossed convolution Scale-space for $c = \frac{1}{6}$ in the variances $2t = 0, 1, 10$ and 100. Since the convolution has infinite support, we mirrored the image across its borders to compute it.

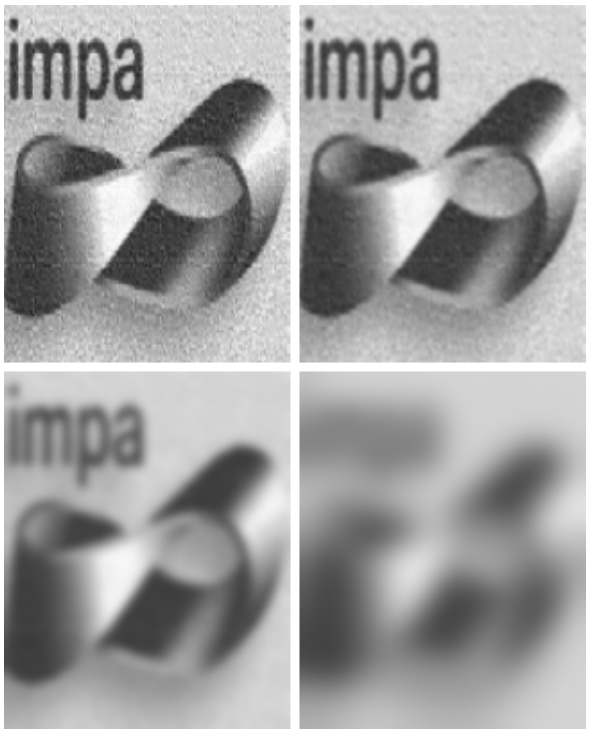


Figure 9: Crossed Convolution Scale-space

# 5 Comparative Analysis

## 5.1 Theoretical Analysis

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

| | semigroup | causality | scale parameter in |
|---|---|---|---|
| sampled Gaussian | | | $\mathbb{R}$ |
| recursive Gaussian | | | $\mathbb{R}$ |
| splines | | | $\mathbb{Q}$ |
| first-order generator[5] | ✓ | ✓ | $\Delta t\mathbb{Z}$ |
| crossed convolutions | ✓ | ✓ | $\mathbb{R}$ |

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[6] to avoid infinite supports; the first-order generator method trades this infinite support by the approximation $L_{t+\Delta t}[m,n] \approx (\delta + \mathcal{A}\Delta t) * L_t[m,n]$; the recursive gaussian method and the spline method approximate the Gaussian kernel by other kernels that can be computed quicker.

[6]It is interesting to note that the cut-off for the crossed convolution method can be avoided using an implementation in the frequency domain, since the Fourier transforms of the symmetric Poisson kernels can be explicitly written.

### 5.2 Runtime

The scale-space libraries are divided in 4 parts: *util*, *ss*, *deriche* and *splines*. The first library contains structure definitions for signals, filters (kernels), 1D and 2D scale-spaces; creation and destruction subroutines; input/output subroutines; signal extension subroutines; etc. The libraries *deriche.c* and *splines.c* contain the subroutines that perform the calculations of recursive and spline scale-spaces, respectively. Finally, *ss.c* has subroutines to compute the remaining scale-spaces previously discussed (sampled Gaussians, first-order generators and crossed convolutions). All of them plus a library on edge detection (described in [1]) can be found at `http://www.visgraf.impa.br/escala`.

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 lower-end Pentium PC, they are a good basis of comparison between different algorithms. All of them show that calculation of scale-spaces can be quite costly.

| | algorithmic | var=1.0 | var=3.0 | var=27.0 | var=243.0 |
|---|---|---|---|---|---|
| sampled Gaussian | $8\sigma N$ | 0.22 | 0.27 | 0.67 | 1.87 |
| recursive 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 |
| first-order generator | $10\sigma^2 N$ | 0.4 | 0.95 | 7.5 | 67.6 |
| crossed convolutions | $16\sigma N$ | $0.22 - 0.36$ | $0.27 - 0.4$ | $0.67 - 0.86$ | $1.87 - 2.37$ |

The first column indicates the number of multiplications for the calculation of each convolution, where $N$ is the number of pixels in the image, $\sigma$ 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 Output difference

We show the squared distance between images $f, g : \{0, 1, \cdots, M-1\} \times \{0, 1, \cdots, N-1\} \to \mathbb{R}$ given by

$$dist\,(f, g) = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} \left(f\,[m, n] - g\,[m, n]\right)^2$$

In the table below, we display the *dist* function 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.

| | | | *variance* | *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 | $3,0$ - extension by zeroes | $0,00005$ |
| 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. ($c = 0$) | First-Order Gen. ($c = 0$) | $27,0$ | $0,0044$ |
| 8 | Crossed Conv. ($c = 0$) | First-Order Gen. ($c = \frac{1}{6}$) | $27,0$ | $0,0029$ |
| 9 | First-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 $c = \frac{1}{6}$.

### 5.4 Final remarks

We note 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.