# Fast Computation of Characteristic Scale Using a Half-Octave Pyramid

James L. Crowley, Olivier Riff, and Justus H. Piater
Projet PRIMA, Lab. GRAVIR-IMAG
INRIA Rhône-Alpes, France

## Abstract

The characteristic (or intrinsic) scale of a local image pattern is the scale parameter at which the Laplacian provides a local maximum. Nearly every position in an image will exhibit a small number of such characteristic scales. Computing a Gaussian jet at a characteristic scale provides a scale invariant feature vector for tracking, matching, indexing and recognition. However, the computational cost of directly searching the scale axis for the characteristic scale at each image position can be prohibitively expensive.

In this paper, we describe a fast method for computing the characteristic scale by interpolating values from a scale-invariant Laplacian pyramid. We present an experimental evaluation of the scale invariance of the impulse response for pyramids computed with three forms of Gaussian filters. We show that interpolation between pixels across scales can be used to provide an accurate estimate of the characteristic scale at each image point.

**Keywords:** Pyramid, Gaussian, multi-scaling, Laplacian profile, DoG, scale invariance, characteristic scale, natural interest points.

**Applications:** Characteristic scale computation, natural interest points detection, pattern recognition, object tracking.

## 1 Introduction

The visual appearance of a neighborhood can be described by a local Taylor series [7]. The coefficients of this series constitute a feature vector that compactly represents the neighborhood appearance for indexing and matching. The set of possible local image neighborhoods that project to the same feature vector are referred to as the "Local Jet". A key problem in computing the local jet is determining the scale at which to evaluate the image derivatives.

Lindeberg [8] has described scale invariant features based on profiles of Gaussian derivatives across scales. In particular, the profile of the Laplacian, evaluated over a range of scales at an image point, provides a local description that is "equivariant" to changes in scale. Equivariance means that the feature vector translates exactly with scale and can thus be used to track, index and recognize structures in the presence of changes in scale.

The problem with this approach is that a direct computation of the characteristic scale at each image position appears to make real-time implementation unfeasible. This paper presents a method to obtain the characteristic scale by interpolating the samples of a half-octave Laplacian Pyramid along both the image and the scale axes. The Laplacian for any image position is obtained by bi-linear interpolation between adjacent sample pixels. Local maxima over scale are determined by a fitting a parabolic function to samples in the scale direction at a pixel. However, not just any multi-resolution pyramid can be used for such calculations. Scale-invariant image description requires that the sampled impulse response be the same at every level of the pyramid.

## 2 Pyramid Methods for Multi-resolution Image Analysis

Multiresolution methods have been used in computer vision since the 1970's. Early work in multi-resolution image description was primarily motivated by a desire to reduce the computational cost of methods for image description and image matching. One of the earliest uses was a technique referred to as "planning" [6], in which image resolution was reduced by summing pixels in non-overlapping $8 \times 8$ blocks. The results of edge detection at low resolution were used to select regions for edge detection at high resolution.

Multi-resolution processing was soon generalized to computing multiple copies of an image by repeatedly summing non-overlapping blocks of pixels and re-sampling until the image reduced to a small number of pixels. Such a structure became known as a multi-resolution pyramid [12]. In a typical early pyramid algorithm, non-overlapping blocks of $4 \times 4$ pixels were summed at each level to produce the next reduced resolution level. Such pyramid structures were used to construct fast algorithms for image segmentation, edge detection, and to accelerate correlation for stereo matching [9, 10]. Unfortunately, computing a pyramid by averaging non-overlapping windows resulted in substantial aliasing. Such aliasing is most noticeable as a large component of additive random noise generated by image translation. Such noise can render most image analysis algorithms unreliable.

The problem of segmentation and classification of textures led a number of researchers to look for general-purpose multi-resolution representations. Burt [2, 3] proposed a multi-resolution pyramid algorithm using smoothing with overlapping windows. Weights for the smoothing filters were obtained by postulating a set of four principles. These principles resulted in the use of a mask that serves as a smoothing filter for repeated re-sampling. While smoothing with these masks did reduce noise, significant aliasing effects still remained. Moreover, Burt's pyramid was not scale invariant.

During this period, a half-octave scale-invariant pyramid algorithm was proposed based on considerations from signal processing [4, 11]. This algorithm was explicitly designed to maintain the same sampled impulse response at each level. Images were smoothed by a Gaussian filter designed to avoid aliasing effects. Unfortunately, the use of large FIR Gaussian filters led to computing times on the order of an hour for a single image.

By the mid-1980's, the multi-resolution pyramid had become a standard structure for use in stereo matching and motion analysis [1]. The use of techniques from digital signal processing provided mathematical tools to understand the effects of repeated smoothing and sampling. By the late 1980's, pyramids were generally computed using Gaussian filters of sufficiently large size so as to minimize the random noise dues to aliasing. However, generally little attention was paid to the scale-invariant properties.

## 3 A scale-invariant half-octave pyramid

### 3.1 Sampling a scale-space representation

A scale-equivariant space can be constructed using any kernel function. Let $x(t)$ be a signal defined over a continuous variable $t$. A kernel function, $k(t)$, can be scaled to any scale factor, $s$, by dividing the $t$. Thus for continuous variables, a scale-equivariant "scale-space" representation of a signal is easily defined, as

$$p(t, s) = x(t) * k(\frac{t}{s}).$$

Computing a sampled digital representation of such a space requires choosing appropriate sample rates for $t$ and $s$. The sample rate, $T_0$, for the $t$ variable is determined by the Nyquist frequency of the signal. For a scale-invariant representation, the $s$ variable should be sampled using an exponential series

$$s_k = s_o^k.$$

This is easily shown by taking the logarithm of $\frac{t}{s}$. The logarithm converts the $\frac{1}{s}$ term into translation along the scale axis. The set of possible scales range from 1 to the number of samples. The logarithmic sample rate in scale depends on the smoothness of the kernel. The cost of brute force sampling of such a space is the number of signal samples, $N$, times the number of scale samples, $\log N$. Thus such a space is, in principle, $O(N \log N)$. Unless the bandwidth of $x(t)$ is limited and the kernel is properly chosen, the actual constants required for such a space are computationaly prohibitive.

A multi-resolution pyramid algorithm produces a sampled scale-space representation of a signal, $p(t, s)$, with a computational complexity of $O(N)$. The reduction in complexity is achieved by re-using each scale-sampled representation of the signal as an intermediate result for producing the next. Strict scale equivariance requires that convolution of a kernel filter with itself produce a scaled copy of the kernel filter:

$$k(\frac{t}{s_o}) = k(t) * k(t)$$

The Gaussian function

$$g(t, \sigma) = \mathrm{e}^{-\frac{t^2}{2\sigma^2}} \tag{1}$$

obeys this property, with a scale factor of $s_o = \sqrt{2}$. More generally, the Gaussian functions are closed under convolution. That is, the convolution with two Gaussians of variance $\sigma_1^2$ and $\sigma_2^2$ results in a Gaussian of variance $\sigma_3^2 = \sigma_1^2 + \sigma_2^2$. As a result, a scale-invariant pyramid can be defined by cascaded convolution with a Gaussian kernel.

The Gaussian function has a number of other properties that make it ideally suited for use as a kernel filter for computing a scale-invariant pyramid. Among these is the fact that a circularly symmetric Gaussian is separable into a product of 1-D components. This property allows us to compute the convolution of an $N \times N$ Gaussian by a series of two 1-D convolutions. Thus the convolution with a Gaussian remains $O(N)$, even when applied to a 2-D $N \times N$ signal.

## 3.2 The $O(N)$ scale-invariant pyramid

A multi-resolution pyramid is an $O(N)$ method for computing a sampled scale space. The reduction in computation is achieved by reusing each level as an intermediate result to compute the next level. This pyramid algorithm is not simply scale equivariant. Each level is resampled at a step size that exactly equals the increase in scale. Thus the ratio of scale to sample rate is constant and the pyramid provides a scale invariant impulse response.

The scale-invariant pyramid algorithm (Figure 1) is composed of an initial convolution with the kernel filter followed by a series of processing stages, $k = 0$ to $K$. For each stage, $k$, the pyramid is composed of three signals $p_0(n, k)$, $p_1(n, k)$ and $p_2(n, k)$. The output of each stage is resampled to produce the input for the next stage. Because of resampling, each stage is composed of $N_k = \frac{N}{2^k}$ samples (in the case of a 1-D signal).

The signal $p_0(n, k)$ serves as the input to the $k$th stage. This signal is convolved with the kernel filter, $g(n, \sigma)$, to provide $p_1(n, k)$:

$$p_1(n, k) = p_0(n, k) * g(n, \sigma)$$

The second stage is computed by convolution with a scaled copy of the kernel filter:

$$p_2(n, k) = p_1(n, k) * g(n, \sqrt{2}\sigma)$$

This scaled copy can be obtained by cascaded convolution with the kernel filter:

$$p_2(n, k) = p_1(n, k) * g(n, \sigma) * g(n, \sigma)$$

To demonstrate the scale invariance, consider the impulse response for a scale-invariant pyramid with a Gaussian kernel $g(n, 0)$ using a typical value of $\sigma = 1.0$. Thus the kernel filter is:

$$g(n, 1) = \mathrm{e}^{-\frac{n^2}{2}}$$

To have an impulse as input, assume an $N$-sample input signal $x(n) = \delta(n - \frac{N}{2})$ composed of zero values, except at position $\frac{N}{2}$ where the value is set to 1. The initialization step convolves the impulse with the kernel filter:

$$p_0(n, 0) = g(n, 1).$$

Thus variance and $\sigma$ at $p_0(n, 0)$ are both 1.0. The next step is

$$p_1(n, 0) = p_0(n, 0) * g(n, 1)$$

Thus the variance at $p_1(n, 0)$ is $\sigma_{01}^2 = 2$ and the scale factor is $\sigma_{01} = \sqrt{2}$. Continuing,

$$p_2(n, 0) = p_1(n, 0) * g(n, 1) * g(n, 1).$$

The variance of $p_2(n, 0)$ is $\sigma_{02}^2 = 4$, and thus $\sigma_{02} = 2$. The result is resampled at $T_1 = 2$ to provide Stage $k = 1$. To show the effects of sampling, consider a change in variables, $m = 2n$, to obtain

$$p_0(m, 1) = p_2(2m, 0).$$

Expressed in the original variable, $n$, resampling does not effect the variance or $\sigma$ of the signal. Thus $\sigma_{10}^2 = 4$, and $\sigma_{10} = 2$. However, convolution with a resampled signal is the same as scaling the kernel filter. Thus,

$$p_1(m, 1) = p_0(m, 1) * g(m, 1) = p_2(n, 0) * g(2n, 1).$$

By virtue of resampling, the Gaussian kernel has effectively been rescaled by a factor of $\sigma = 2$. This is equivalent to rescaling the variance of the Gaussian by 4. Thus $\sigma_{11}^2 = 8$, and $\sigma_{11} = 4$. Continuing the stage,

$$p_2(m, 1) = p_1(m, 1) * g(m) * g(m)$$

which gives $\sigma_{12}^2 = 16$, and $\sigma_{11} = 8$. The result is resampled to provide the input to the next stage and the process is repeated:

$$p_0(m, 3) = p_2(2m, 2)$$

The result is a sequence of signals in which both the sample rate and the scale factor grow in powers of 2. At each stage, an intermediate result for $p_1(n, k)$ provides a $\sqrt{2}$ scaling of the impulse response.

The 1-D algorithm defined above is easily generalized to 2-D by replacing the variable $n$ with $i, j$. This input signal is changed from $x(n)$ of size $N$ sample to $x(i, j)$ of size $N^2$. However, the Gaussian kernel is separable:

$$g(i, j, \sigma) = \mathrm{e}^{-\frac{i^2+j^2}{2\sigma^2}} = \mathrm{e}^{-\frac{i^2}{2\sigma^2}} * \mathrm{e}^{-\frac{j^2}{2\sigma^2}}$$

Thus, convolution with the kernel with an $N \times N$ image can be computed as a series of two $O(N)$ 1-D convolutions. Thus the cost of convolution with a Gaussian remains $O(N)$ and the resulting pyramid remains an $O(N)$ algorithm.

## 4 An experimental comparison of fast Gaussian filters

### 4.1 Fast implementation of Gaussian Filters

Digital filters can be designed using either a direct (FIR) or recursive (IIR) form. The direct form is obtained as a finite number of samples of the desired impulse response. The recursive form is designed as a ratio of polynomials in the $z$ domain. Closure under convolution provides a third method for designing Gaussian filters by cascade convolution. The following section compares these three implementation methods for a 1-D Gaussian filter.

#### 4.1.1 The FIR implementation of a Gaussian

The simplest means to implement a digital Gaussian filter is to sample the Gaussian function (Eqn. 1) at integer multiples of $T_0$. For $\sigma = 1$ pixel, a reasonably good approximation is obtained using a kernel width of 9 pixels.

#### 4.1.2 Binomial filters

Binomial filters are obtained with cascaded convolution of a kernel filter composed of $[1, 1]$. The coefficients for the $n$th filter in the series, $b_n(m)$, are defined by:

$$b_n(m) = [1, 1]^{*n}$$

where the exponent $*n$ denotes $n$ autoconvolutions. The set of filter coefficients is well known as the binomial series, often computed using Pascal's triangle. This series provide best approximation to a Gaussian a finite size. The properties of the binomial filters are particularly easy to compute. For example, the sum of coefficients for the binomial $b_n(m)$ is simply $2^n$, while the variance is $\sigma^2 = \frac{n}{4}$.

The binomial filters $b_2(m)$ (with coefficients $1, 2, 1$) and $b_4(m)$ $(1, 4, 6, 4, 1)$ are of special interest. The Fourier transform of $b_2(m)$ is a single period of a cosine on platform and thus is a monotonic low-pass filter with no ripples in the stop band:

$$B_2(w) = 2 + 2\cos w$$

Since the even-order binomials are auto-convolutions, they are powers of this filter and thus also have no ripples in the stop band. The filters $b_2(m)$ and $b_4(m)$ have variances of $\frac{1}{2}$ and 1, respectively. The filter $b_4(m)$ is equivalent to $b_2(m) * b_2(m)$. Thus, a $\sigma = 1$ kernel filter can be computed by two convolutions with the kernel $[1, 2, 1]$ at a cost of two multiplications and 4 additions per pixel.

#### 4.1.3 Recursive filters

Different recursive implementations of Gaussian filters have been proposed by Deriche [5] and by Vliet, Young and Verbeek [13]. To maintain shift invariance (or zero phase), the filter is implemented as a cascade of forward and backward difference equations with real-valued coefficients $b$.

$$\begin{aligned} \text{Backward:} \quad v[n] &= \alpha x[n] - \sum_{i=1}^{N} b_i v[n-i] \\ \text{Forward:} \quad y[n] &= \alpha v[n] - \sum_{i=1}^{N} b_i y[n+i] \end{aligned}$$

with $\alpha = 1 + \sum_{i=1}^{N} b_i$. An interesting property of recursive filters is that the number of operations is independent of the variance of the filter. In the following we consider recursive filters of size $N = 3$ and $N = 5$.

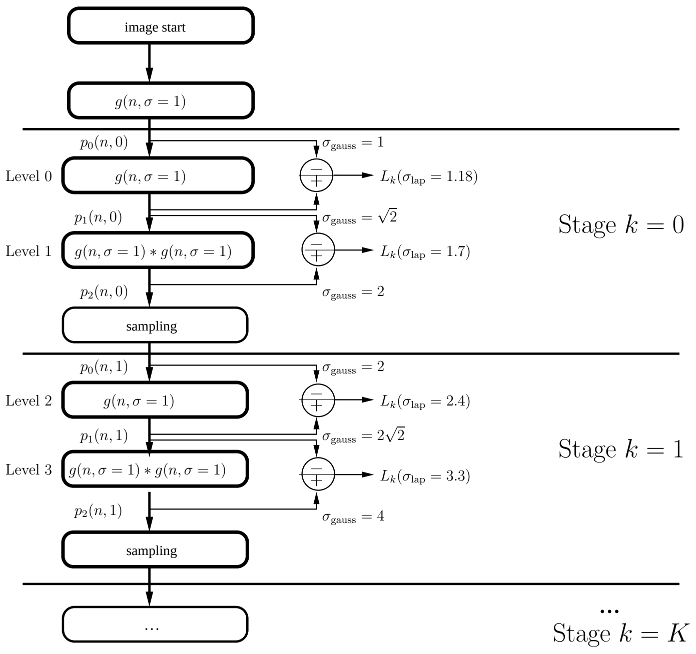
image start
$g(n,\sigma=1)$
$p_0(n,0)$
$\sigma_{\text{gauss}}=1$
Level 0
$g(n,\sigma=1)$
$L_k(\sigma_{\text{lap}}=1.18)$
$p_1(n,0)$
$\sigma_{\text{gauss}}=\sqrt{2}$
Stage $k=0$
Level 1
$g(n,\sigma=1)*g(n,\sigma=1)$
$L_k(\sigma_{\text{lap}}=1.7)$
$p_2(n,0)$
$\sigma_{\text{gauss}}=2$
sampling
$p_0(n,1)$
$\sigma_{\text{gauss}}=2$
Level 2
$g(n,\sigma=1)$
$L_k(\sigma_{\text{lap}}=2.4)$
$p_1(n,1)$
$\sigma_{\text{gauss}}=2\sqrt{2}$
Stage $k=1$
Level 3
$g(n,\sigma=1)*g(n,\sigma=1)$
$L_k(\sigma_{\text{lap}}=3.3)$
$p_2(n,1)$
$\sigma_{\text{gauss}}=4$
sampling
...
...
Stage $k=K$

Figure 1: Scheme of the pyramid

## 4.2 Approximating a Laplacian as a difference of Gaussians

A difference of Gaussians (DoG) is widely used as an approximation for the Laplacian of a Gaussian. A Gaussian low-pass pyramid is thus easily used to compute a Laplacian pyramid. However, the precision of this approximation is rarely studied. In radial form, the normalized Laplacian is a second derivative, given by:

$$\nabla^2 G(r,\sigma_{\text{lap}}) = \frac{r^2-\sigma_{\text{lap}}^2}{\sigma_{\text{lap}}^5\sqrt{2\pi}}\mathrm{e}^{-\frac{1}{2}\frac{r^2}{\sigma_{\text{lap}}^2}}$$

The difference of Gaussians is:

$$\mathrm{DOG}(r,\sigma_{\text{dog}}) = \frac{1}{\sigma_1\sqrt{2\pi}}\mathrm{e}^{-\frac{1}{2}\frac{r^2}{\sigma_1^2}} - \frac{1}{\sigma_{\text{dog}}\sqrt{2\pi}}\mathrm{e}^{-\frac{1}{2}\frac{r^2}{\sigma_{\text{dog}}^2}}$$

Approximating the Laplacian with a difference of Gaussians requires the specification of the two parameters $\sigma_1$ and $\sigma_{\text{dog}}$. Our Gaussian pyramid provides Gaussians in scale step sizes of $\sqrt{2}$ so that $\sigma_1 = \sqrt{2}\sigma_{\text{dog}}$. To determine the $\sigma$ of the corresponding Laplacian, we wrote a simple script search for the value of $\sigma$ for which the sum of squares of the difference is minimized. The minimum error energy was obtained when $\sigma_{\text{lap}} = 1.18\sigma_{\text{dog}}$. Figure 2 shows the difference between a Laplacian in radial form and a difference of Gaussians.

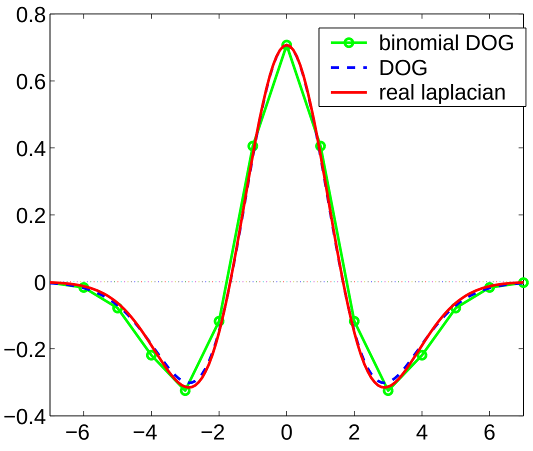

Figure 2: Comparisons of real Laplacian versus real DoG and binomial DoG for $\sigma_{\text{dog}} = \sqrt{2}$ and $\sigma_1 = \sqrt{2}\sigma_{\text{dog}} = 2$ and $\sigma_{\text{lap}} = 1.7$

In Figure 3, both the DoG computed with binomial coefficients and a DoG computed using FIR Gaus-

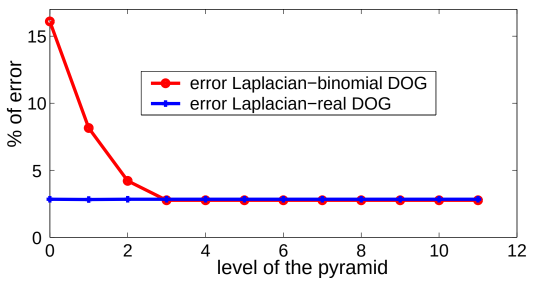

Figure 3: Evaluation of algorithm accuracy

sian's ($N = 9$) are compared to a true Laplacian. The FIR DoG demonstrates a constant error of approximately 3.6% at all scales. The Binomial DoG starts with an error of 16% but rapidly descends to match the 3.6% error of the FIR implementation by the third image of the pyramid.

## 5 Comparison of Scale Invariance

### 5.1 Comparison of Gaussian Filters

The scale invariance of the impulse response for a pyramid with $\sigma_0 = 1$ was evaluated on an image where the central pixel has a value of 100 and all others pixels are set to zero. Gaussian Pyramids with $\sigma_0 = 1$ were computed using the three filter methods: FIR ($N = 9$), Recursive ($N = 5$) and Binomial. Two DoG images were computed at each level:

$$\begin{aligned} d_{01}(i,j,k) &= p_1(i,j,k) - p_0(i,j,k) \\ d_{12}(i,j,k) &= p_2(i,j,k) - p_1(i,j,k) \end{aligned}$$

All three filters exhibited rapid convergence to a scale-invariant impulse reponse. For example, the percentage of change for the center pixel at levels $k = 1, 2, 3, 4$ are shown for $d_{01}(i,j,k)$ and $d_{12}(i,j,k)$ in Figure 4. These are representative of the errors observed at other pixel positions. One can note that the invariance error for $d_{01}$ is within 3%. Both binomial, recursive and FIR filter implementations rapidly converged to extremely small errors (less than 0.0001%).

The percentage error for $d_{12}(i,j,k)$ are within 1% with the same rapid convergence. The improvement in error rates is primarily due to the extra smoothing provided by a larger ratio of $\sigma$ to sample rate,

Table 1: Computational costs of filters.

| Filter | FIR | binomial | Rec. | Rec. |
|---|---|---|---|---|
| | $N = 9$ | [1,2,1] | $N = 3$ | $N = 5$ |
| $g_0$ | 36 | 16 | 28 | 44 |
| $g_1$ | 72 | 32 | 28 | 44 |
| $g_0 * g_1$ | **108** | **48** | **56** | **88** |

which results in less error due to sampling. The experiments also validate our choice of $\sigma_0 = 1.0$ for our pyramid by showing that such pyramid provides reasonably accurate scale invariance.

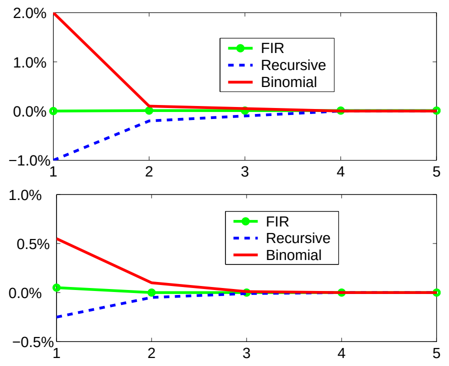

Figure 4: 1.(top): Scale invariance of $d_{01}(i,j,k)$ 2.(bottom): Scale invariance of $d_{12}(i,j,k)$

### 5.2 Computational Cost

Table 1 recapitulates the previous results in operations per pixel for filters $g_0$ and $g_1$ with the FIR ($N = 9$), the binomial $[1, 2, 1]$ and 2 recursive filters ($N = 3$ and $N = 5$). This shows that a pyramid computed using the binomial filter has a lower cost than either the recursive filter or the direct FIR filter.

## 6 Characteristic Scale

Determining characteristic scale requires comparison of Laplacian values along the scale axis. However, because the pyramid is computed on resampled images, Laplacian values are not directly available at most pixels. These samples were eliminated without loss of information due to smoothing. Thus they can be easily be recovered through bi-linear interpolation.

Suppose that we seek the value at pixel $i, j$ at level $k$, and that this pixel falls between pixels $(i_0, j_0)$ and

$(i_1, j_1)$. Note that $T_k = 2^k$ is sample rate at level $k$. Given

$$\begin{aligned} a &= \frac{p(i_1, j_0, k) - p(i_0, j_0, k)}{T_k} \\ b &= \frac{p(i_0, j_1, k) - p(i_0, j_0, k)}{T_k} \\ c &= a + \frac{p(i_0, j_1, k) - p(i_1, j_1, k)}{T_k} \end{aligned}$$

the interpolated value at pixel $i, j$ is

$$\begin{aligned} p(i, j, k) &= a(i - i_0) + b(j - j_0) \\ &\quad + c(i - i_0)(j - j_0) + p(i_1, j_1, k). \end{aligned}$$

## 6.1 Computing Characteristic scale

Let us refer to the difference of Gaussian images at each level $k$ as $l = 0$ for $d_{01}$ and $l = 1$ for $d_{12}$. In that case, we can define an integer scale index $n = 2k + l$. For a typical 6-level pyramid, $n$ runs from 0 to 11. Using this index as a free variable, the Laplacian profile, at pixel $(i, j)$ is the series of interpolated Laplacian values, the $d(n)$ determined for each pixel $i, j$. The charactistic scale at a pixel is the peak in this profile.

The precision of the characteristic scale can be improved by interpolation using a parabola for the three samples closest to the peak. Let $d(n_0)$ be a local peak in $d(n)$. The interpolated extremum is

$$\sigma_{\max} = n_0 + 1 + \frac{d(n_0 - 1) - d(n_0 + 1)}{2(d(n_0 - 1) + d(n_0 + 1) - 2d(n_0))}.$$

Multiple characteristic scales correspond to concentric patterns in an image. The half-octave pyramid limits discrimination of such patterns to concentric scale changes of powers of 2. This is a fundamental limitation due to sampling scale at multiples of $\sqrt{2}$. Fortunately denser concentric scales tend to be rare in real images.

The following graph (Figure 5) shows an example of Laplacian values as a function of the characteristic scale on a 12-level pyramid (i.e., 6 stages). The extremum of the curve in Figure 5 is located around a characteristic scale of 10 pixels. The interpolated curve is shown as a dashed line on this figure.

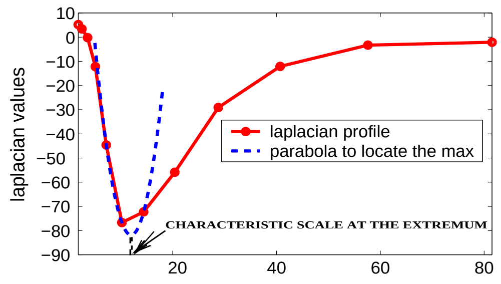

Figure 5: example of Laplacian profile

## 6.2 Estimating size from characteristic scale

To evaluate the ability of intrinsic scale to recover size, we constructed an image set containing uniform disks of radius from 1 to 100 pixels. Each image was processed with a binomial pyramid, and the profile of Laplacian values was computed at the center of the circle. This profile was interpolated using parabolic interpolation. Figure 6 compares the results to the ideal straight line.

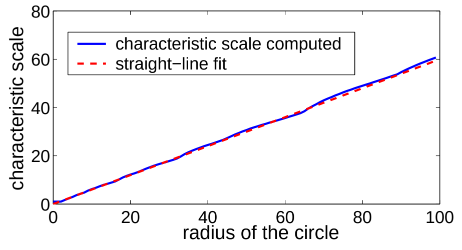

Figure 6: Scale invariance: The characteristic scale was estimated at the center pixel for 100 images containing each containing disks of radius from 1 to 100 pixels.

The real and interpolated values of the Laplacian at each extremum are compared in Figure 7. The constancy of these curves further confirms the scale invariance of the pyramid.

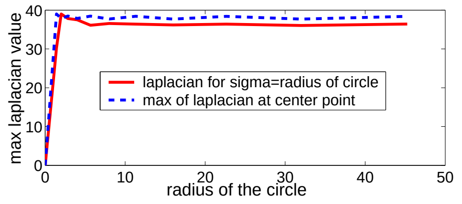

Figure 7: Scale invariance of the Laplacian values

### 6.3 Invariance to rotation

Figure 8 demonstrates the invariance to rotation of the characteristic scales. In this experiment, the characteristic scale was computed at every pixel of an image containing a Dirac impulse. The resulting image of characteristic scales, encoded as gray levels, is displayed together with a set of level curves. Note the slight deviations from perfect radial symmetry.

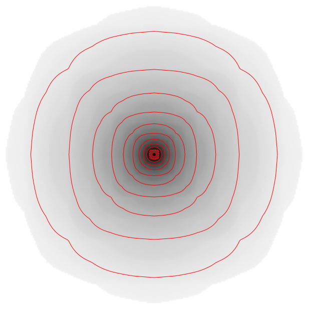

Figure 8: Characteristic scales of a Dirac image and level curves

## 7 Conclusions

We have presented a simple and fast algorithm to evaluate characteristic scales at any pixel in an image. This method is based on the computation of differences of Gaussians obtained by binomial filtering in a pyramid.

The experiments described above demonstrate that a scale-invariant half-octave pyramid computed with a binomial kernel can provide an efficient and precise means to compute characteristic scales. At first glance, it may seem surprising that a relatively crude Gaussian approximation such as a 1-2-1 binomial filter yields reliable estimates of characteristic scale.

## References

[1] P. Anandan. *Measuring Visual Motion from Image Sequences.* PhD thesis, Computer Science Department, University of Massachusetts, 1987.

[2] P. J. Burt. Fast filter transforms for image processing. *Computer Graphics and Image Processing*, 16:20–51, 1981.

[3] P. J. Burt and E. H. Adelson. The Laplacian pyramid as a compact image code. *IEEE Transactions on Communications*, 31:532–540, 1983.

[4] J. L. Crowley. *A Representation for Visual Information.* PhD thesis, Carnegie-Mellon University, 1981.

[5] R. Deriche. Recursively implementing the Gaussian and its derivatives. Rapport de Recherche 1893, INRIA, Sophia Antipolis, France, Apr. 1993.

[6] M. D. Kelly. Edge detection by computer in pictures using planning. *Machine Intelligence*, 6:379–409, 1971.

[7] J. J. Koenderink and A. J. van Doorn. Representation of local geometry in the visual system. *Biological Cybernetics*, 55:367–375, 1987.

[8] T. Lindeberg. Feature detection with automatic scale selection. *International Journal of Computer Vision*, 30(2):77–116, 1998.

[9] H. P. Moravec. Towards automatic visual obstacle avoidance. In *Proceedings of the 5th International Joint Conference on Artificial Intelligence*, 1977.

[10] A. Rosenfeld and G. J. Vanderbrug. Coarse-fine template matching. *IEEE Transactions on Systems, Man and Cybernetics*, 7:104–107, 1977.

[11] R. Stern and J. Crowley. Fast computation of the difference of low-pass transform. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 6:212–222, 1984.

[12] S. L. Tanimoto and T. Pavlidis. A hierarchical data structure for picture processing. *Computer Graphics and Image Processing*, 4:104–119, 1975.

[13] L. J. van Vliet, I. T. Young, and P. W. Verbeek. Recursive Gaussian derivative filters. In *Proc. 14th International Conference on Pattern Recognition (ICPR'98)*, volume 1, pages 509–514. IEEE Computer Society Press, Aug. 1998.