# Fast Computation of Scale Normalised Gaussian Receptive Fields

**James L. Crowley and Olivier Riff**

Laboratoire GRAVIR, INRIA Rhône Alpes,
655 Ave de l'Europe, F-38330 Montbonnot, France
`{Crowley, Riff}@inrialpes.fr`
`http://www-prima.imag.fr`

**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 vector of Gaussian derivatives (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. We describe a fast method for computing a vector of Gaussian derivatives that are normalised to the characteristic scale at each pixel. This method is based on a scale equivariant half-octave binomial pyramid. The characteristic scale for each pixel is determined by an interpolated maximum in the Difference of Gaussian as a function of scale. We show that interpolation between pixels across scales can be used to provide an accurate estimate of the intrinsic scale at each image point. We present an experimental evaluation that compares the scale invariance of this method to direct computation using FIR filters, and to an implementation using recursive filters. With this method we obtain a scale normalised Gaussian Jet at video rate for a 1/4 size PAL image on a standard 1.5 Ghz Pentium workstation.

## 1 Introduction

The visual appearance of a neighborhood can be described by a local Taylor series [1]. The coefficients of this series constitute a feature vector that compactly represents the neighborhood appearance for indexing[2] and matching[3]. 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 [4] 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 Fast Computation of Chromatic Receptive Fields

Multi-resolution 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", in which image resolution was reduced by summing pixels in non-overlapping 8x8 blocks [5]. 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 [6]. In a typical early pyramid algorithm, non-overlapping blocks of 4x4 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. 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 proposed a multi-resolution pyramid algorithm using smoothing with overlapping windows [7]. 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 [8]. 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 [9]. 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

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 t by s. 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 the appropriate sample rates for t and for s. The sample rate, $T_o$ , for the t variable is determined by the frequency content of the signal that should be preserved  in  the sampled representation. For a scale-invariant representation, the variable s should be sampled using an exponential series

$$S_k = s_0^k$$

This is easily shown by taking the logarithm of t/s. The logarithm converts the 1/s term into translation along the scale axis. Thus changes in scale are expressed as translation in a logarithmic scale space.

The set of possible scales range from 1 to the number of samples. The desired sample rate in scale will often depend 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, LogN. Thus computational cost of 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 computationally 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:

$$k(\frac{t}{s_0}) = k(t) * k(t)$$

The Gaussian function:

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

obeys this property, with a scale factor of $s_0 = \sqrt{2}$. More generally, the Gaussian functions are closed under convolution. That is, the convolution of 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 NxN 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 NxN signal.

## 3.1 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 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. Scaling a signal translates its response in the scale axis.

The scale-equivariant pyramid algorithm shown in figure 4 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 $N_k=N/2^k$ samples (in the case of a 1-D signal).
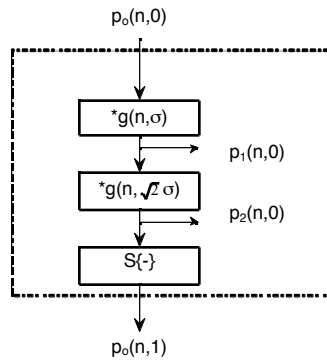


Figure4: First stage of the scale invariant pyramid algorithm.

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 equivariance, consider the impulse response for a scale-invariant pyramid with a Gaussian kernel $g(n,0)$ using a typical value of $\sigma=1$. Thus the kernel filter is:

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

To have an impulse as input, assume an N-sample input signal $s(n) = \delta(n - \frac{N}{2})$ composed of zero values, except at position $N/2$ 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 the variance and standard deviation at $p_0(n,k)$ are both 1.0. The next step is

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

Thus the variance at $p_1(n,k)$ 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 $p_2(n,k)$ 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

$$p_1(m,1) = p_2(2m,0)$$

Expressed in the original variable, n, resampling does not effect the variance or $\sigma$ of the signal. Thus $\sigma_{01}^2 = 2$ and $\sigma_{01} = 2$. 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

$$\sigma_{11}^2 = 8 \text{ , and } \sigma_{11} = 2\sqrt{2} \text{ .}$$

Continuing the stage,

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

which gives $\sigma_{12}^2 = 16$, and $\sigma_{12} = 4$ . 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 x, y. This input signal is changed from p(n) of size N sample to p(x,y) of size $N^2$. However, the Gaussian kernel is separable:

$$g(x,y,\sigma) = e^{-\frac{x^2+y^2}{2\sigma^2}} = e^{-\frac{x^2}{2\sigma^2}} * e^{-\frac{y^2}{2\sigma^2}}$$

Thus, convolution with the kernel with an NxN 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 is an O(N) algorithm.

# 4  Experimental comparison of fast Gaussian filters

## 4.1  Fast 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 at integer multiples of $T_o$. For $\sigma=1$, a reasonably good approximation is obtained using a kernel width of 9 pixels. This gives

$$G(x) = e^{-\frac{x^2}{2}}$$

for integer values of x in the range $x \in [-4, 4]$.

## 4.2  Binomial filters

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

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

where the exponent *n denotes n auto-convolutions. The set of filter coefficients is well known as the binomial series, often computed using Pascal's triangle. This series provide the best (least sum of squares error) approximation to a Gaussian function by an integer coefficient sequence of finite duration. The properties of the binomial filters are particularly easy to compute. For example, for the $n^{th}$ binomial $b_n(m)$, there are n coefficients, whose sum is $2^n$. The midpoint (or center of gravity) is the coefficients at $m = \frac{n}{2}$ and the variance is $\sigma^2 = \frac{n}{4}$ .

The binomial filters $b_2(m)$ (with coefficients [1, 2, 1]) and $b_4(m)$ (with coefficients [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:

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

Since the even-order binomials are auto-convolutions of this filter, their Fourier transforms are powers of $B_2(\omega)$ and thus have no ripples in the stop band. The filters $b_2(m)$ and $b_4(m)$ have variances of 0.5 and 1, respectively. The filter is $b_4(m)$ equivalent to $b_2(m) * b_2(m)$. Thus, a $\sigma=1$ Gaussian filter can be computed by two convolutions with the kernel $[1, 2, 1]$ at a cost of two multiplications and 4 additions per pixel.

### 4.2.1 Recursive filters

Recursive implementations of Gaussian filters have been proposed by Deriche [10] and by Vliet, Young and Verbeek [11]. 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.

*Backward:* $\qquad v[n] = \alpha x[n] - \displaystyle\sum_{i=1}^{N} b_i v[n-i]$

*Forward:* $\qquad y[n] = \alpha v[n] - \displaystyle\sum_{i=1}^{N} b_i y[n+i]$

with. $\alpha = 1 + \displaystyle\sum_{i=1}^{N} b_i$

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=5.

### 4.3 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:

$$\nabla^2 G(r,\sigma) = \frac{r^2 - \sigma^2}{\sigma^4 \sqrt{2\pi}} e^{-\frac{1}{2}\frac{r^2}{\sigma^2}}$$

The difference of Gaussians is:

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

Approximating the Laplacian with a difference of Gaussians requires the specification of the two parameters $\sigma_1$ and $\sigma_{dog}$. Our Gaussian pyramid provides

Gaussians in scale step sizes of $\sqrt{2}$ so that $\sigma_1 = \sqrt{2}\sigma_{dog}$. 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 $\sigma_{lap}=1.18\sigma_{dog}$. Figure 5 shows the difference between a Laplacian in radial form and a difference of Gaussians.
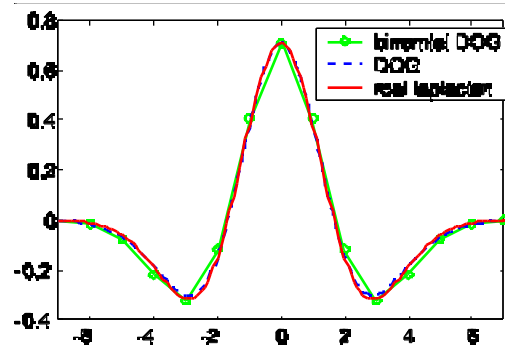


Figure 5: Comparisons of real Laplacian versus real DoG and binomial DoG for $\sigma_{dog} = \sqrt{2}$

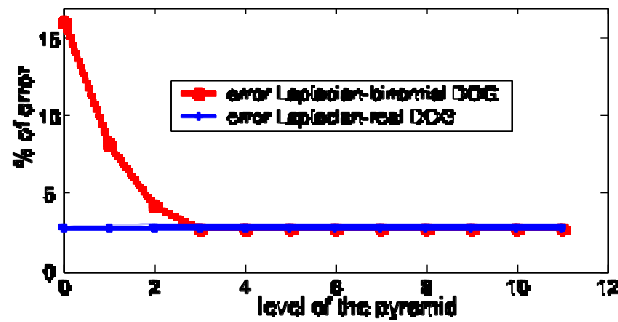and $\sigma_1 = \sqrt{2}\sigma_{dog}$ and $\sigma_{lap} = 1.7$



**Figure 6:** Evaluation of accuracy of approximation of a Laplacian with a binomial DoG.

In Figure 6, a DoG computed with binomial coefficients and a DoG computed using an FIR Gaussian 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.

The binomial pyramid based on the Kernel filter [1, 4, 6, 4, 1] provides the fastest implementation of the methods tested. The experiments indicate that this method provides sufficiently accurate approximation for a Laplacian.

# 5 Comparison of scale invariance

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:

$$d_{01}(i,j,k) = p_1(i,j,k) - p_0(i,j,k)$$

$$d_{12}(i,j,k) = p_2(i,j,k) - p_1(i,j,k)$$

All three filters exhibited rapid convergence to a scale-invariant impulse response. 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 7. These are representative of the errors observed at other pixel positions. One can note that the invariance error for $d_{01}$ is within 3%. The binomial, the recursive and the 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, 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.
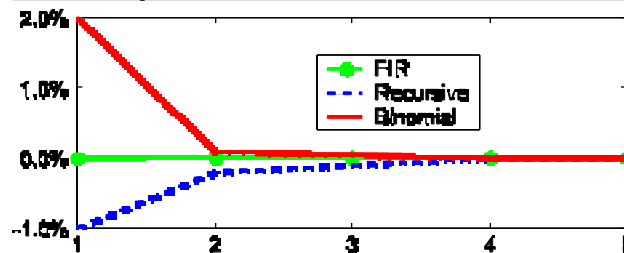


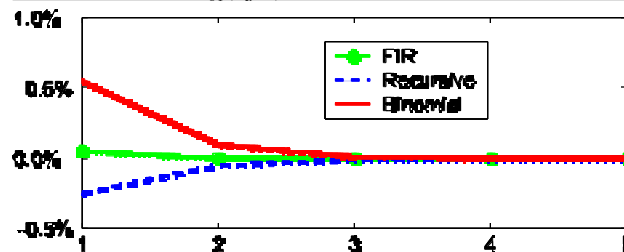Figure 7a Scale invariance of $d_{01}(i,j,k)$ for FIR, Recursive and Binomial Laplacians



Figure 7b Scale invariance of $d_{12}(i,j,k)$ for FIR, Recursive and Binomial Laplacians

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.

| Filter | FIR N=9 | Binomial | IIR N=3 | IIR N=5 |
|--------|---------|----------|---------|---------|

| $g_0(n)$ | 36 | 16 | 28 | 44 |
|---|---|---|---|---|
| $g_1(n)$ | 72 | 32 | 28 | 44 |
| $g_0(n)*g_1(n)$ | 108 | 48 | 56 | 88 |

Table 1. Computational cost (Standard Ops) per pixel for different filter types.

# 6 Determining Intrinsic 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 with minimal loss of information due to smoothing. Thus they can 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 $(i_o, j_0)$ and $(i_1, j_1)$. Note that $T_k = 2^k$ is the sample rate at level k. Given

$$a = \frac{p(i_1, j_0, k) - p(i_0, j_0, k)}{T_k}$$

$$b = \frac{p(i_0, j_1, k) - p(i_0, j_0, k)}{T_k}$$

$$c = a + \frac{p(i_0, j_1, k) - p(i_1, j_1, k)}{T_k}$$

the interpolated value at pixel i, j is

$$p(i, j, k) = a(i - i_0) + b(j - j_0) + c(i - i_0)(j - j_0) + p(i_1, j_1, k)$$

## 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 $d_{12}$. 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 peak in this profile is equivariant with scale. We refer to the scale of this peak as the characteristic scale of the signal at that image position.

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 8) 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 9 is located around a characteristic scale of 10 pixels. The interpolated curve is shown as a dashed line on this figure.
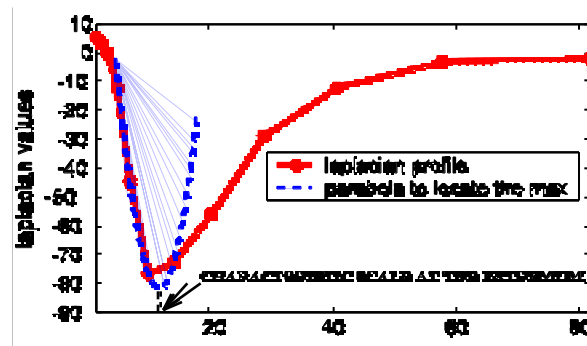


Figure 8. Interpolation of the Laplacian profile

## 6.2 Estimating size from intrinsic 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. The interpolated values of the Laplacian at each extremum are compared in Figure 9 to an ideal straight line. The constancy of these curves further confirms the scale invariance of the pyramid.
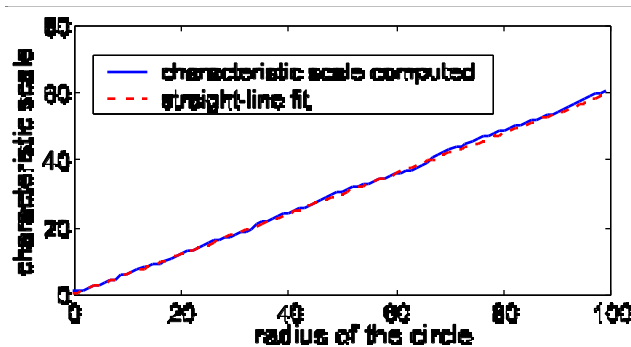
**Figure 9:** Scale invariance: The characteristic scale was estimated at the center pixel for 100 images containing each containing disks of radius from 1 to 100 pixels.

# 7 Invariance to rotation

Figure 10 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.
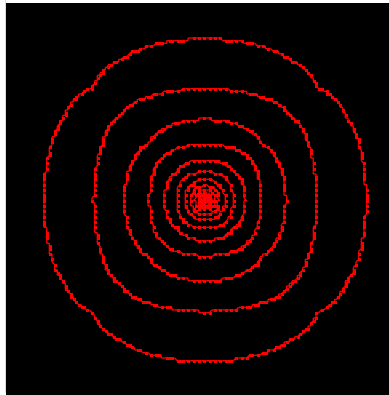


**Figure 10:** Rotational invariance of Intrinsic Scale

# 8 Synthesis of Normalized Receptive Fields

Receptive fields at canonical (row and column) directions can be directly computed from differences of adjacent pixels at level of the binomial pyramid. Such derivative filters are close approximations to Gaussian derivatives at the same scale. In this section we explain how to calculate such receptive fields from the binomial pyramid, how to steer the receptive fields to the intrinsic orientation, and demonstrate that the impulse response has a scale invariance that is similar to the value demonstrated above for the binomial filters.

For each image point in a logo or ROI, a local feature vector can be produced by using the binomial pyramid to compute a vector of 9 chromatic receptive fields [12]. This vector can be computed in a manner that is normalized to the intrinsic scale and orientation at that point. Such normalization provides a vector for robust matching invariant to transformations scale and orientation.

The receptive field vector is based on computing the product with a vector of image differences in the row and column directions of the luminance and chrominance images. The luminance (L) and chrominance ($C_1$, $C_2$) images can be obtained by

$$L = R+G+B, \quad C_1 = \frac{1}{2}(R-B) \qquad C_2 = \frac{1}{3}(R+B-G)$$

A first derivative along the row or column direction is obtained by convolution with the mask $[1, 0, -1]$ in that direction. For position (x, y) at sampled pyramid image k, this is equivalent to computing the differences of adjacent pixels

$$P_x(x, y, k) = P(x - 1, y, k) - P(x+1, y, k)$$

$$P_y(x, y, k) = P(x, y - 1, k) - P(x, y+1, k)$$

The second difference is computed by a convolution of the mask $[1, -2, 1]$. For the rows and columns at pyramid level k, this is equivalent to.

$$P_{xx}(x, y, k) = P(x - 1, y, k) + P(x + 1, y, k) - 2 \cdot P(x, y, k)$$

$$P_{yy}(x, y, k) = P(x, y - 1, k) + P(x, y + 1, k) - 2 \cdot P(x, y, k)$$

The mixed derivative is

$$P_{xy}(x,y,k) = P(x-1,yk) + P(x+1,y,k) + P(x,y-1,k) + P(x,y+1,k) - 4 \cdot P(x,y,k)$$

Image differences computed over the $L, C_1$, and $C_2$ images compose the vector.

$$\dot{P} = (P^L, P_x^L, P_y^L, P_{xx}^L, P_{xy}^L, P_{yy}^L, P^{C_1}, P_x^{C_1}, P_y^{C_1}, P^{C_2}, P_x^{C_2}, P_y^{C_2})$$

This vector gives an un-oriented feature vector at each point equivalent to

$$\dot{P} =< \dot{G}(\sigma), p(x, y) >$$

where

$$\dot{G} = (G^L, G_x^L, G_y^L, G_{xx}^L, G_{xy}^L, G_{yy}^L, G^{C_1}, G_x^{C_1}, G_y^{C_1}, G^{C_2}, G_x^{C_2}, G_y^{C_2})$$

is the vector of Gaussian derivatives at the intrinsic scale.

The differences in row and column directions can be steered to the intrinsic orientation, $\theta$, at pixel (x, y) using the steerable filter formulas of Freeman and Adelsen [13]:

$$V_1 = P_x \cos(\theta) + P_y \sin(\theta)$$

$$V_2 = P_{xx} \cos(\theta)^2 + P_{xy} \cos(\theta) \sin(\theta) + P_y \sin(\theta)^2$$

where the intrinsic orientation for each pixel is provided by :

$$\theta = \tan^{-1}(\frac{P_y}{P_x})$$

The steered local feature vector for luminance and chrominance at the intrinsic scale and orientation can be written as

$$\dot{V}(x, y) = (V^L, V_1^L, V_2^L, V^{C_1}, V_1^{C_1}, V_2^{C_1}, V^{C_2}, V_1^{C_2}, V_2^{C_2})$$

where the subscript 1 represents a first derivative and the subscript 2 represents a second derivative

Figure 11a and 11b show the impulse response obtained from such a calculation. White pixels represent positive values and purple negative values. A second derivative can be obtained from a convolution of [1, −2, 1]. Figure 11c and 11d show examples of the resulting impulse response. The mixed impulse responses can be obtained by convolving the row directions with [1, 0, −1] followed by convolving the column directions with this filter. Figure 11e shows an example of the resulting impulse response. Synthetic filters at any desired angle can be computed from these filters using a weighted sum of the derivatives.



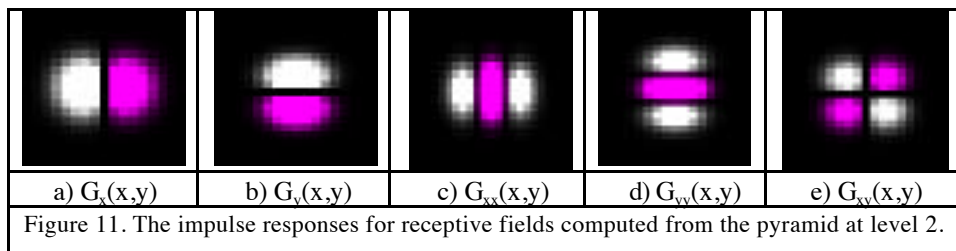| a) G$_x$(x,y) | b) G$_y$(x,y) | c) G$_{xx}$(x,y) | d) G$_{yy}$(x,y) | e) G$_{xy}$(x,y) |
| --- | --- | --- | --- | --- |

Figure 11. The impulse responses for receptive fields computed from the pyramid at level 2.

Because the sampled impulse response of the pyramid is the same at every level (beyond the first), the impulse responses of the derivatives are also equivalent at all levels. As with the Laplacian, these derivative impulse responses are "equivariant" with scale. As a demonstration of the invariance of the impulse response, Figure 12 shows the impulse responses from filter G$_x$(i, j, k) and G$_{xx}$(i, j, k) computed at the first image in each of the stages of the binomial pyramid. That is, these impulse responses are

$$G_x(i, j, k) = [-1,0,1] * p_o(i, j, k)$$

$$G_{xx}(i, j, k) = [1, -2, 1] * p_1(i, j, k)$$

for k = 0, 1, 2, 3, 4, 5. It can be seen from figure 12 that after the first level of the pyramid, the impulse response is invariant to scale.
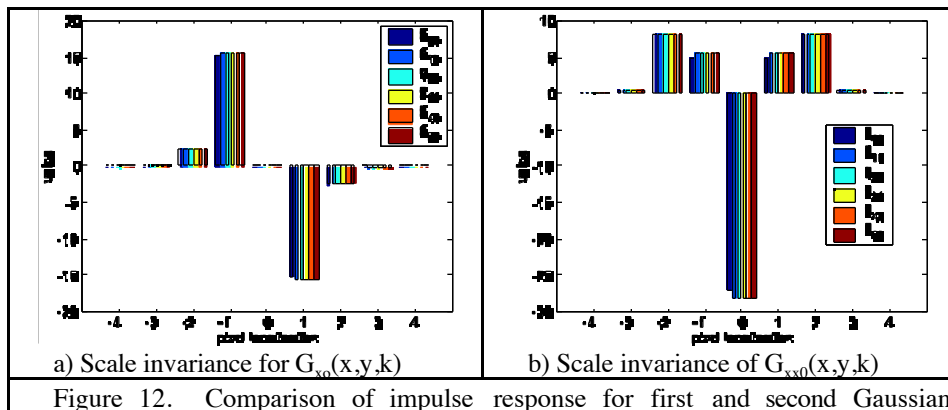


| a) Scale invariance for G$_{xo}$(x,y,k) | b) Scale invariance of G$_{xx0}$(x,y,k) |
| --- | --- |

Figure 12.    Comparison of impulse response for first and second Gaussian

derivatives for pyramid stages k=0,1,2,3,4,5.

## 9 Summary and conclusion

The binomial pyramid gives a simple and fast method 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. However, this kernel allows video rate calculation of intrinsic scale for 1/4 PAL images on a standard 1.5 GHz personal computer.

### Acknowledgment

### References

1. J. J. Koenderink and A. J. van Doorn, "Representation of local geometry in the visual system", Biological Cybernetics, 55:367-375, 1987.
2. D. G. Lowe, "Object Recognition from local scale-invariant features", in 1999 International Conference on Computer Vision (ICCV-99), Corfu Greece, pp 1150-1157, Sept. 1999.
3. C. Schmid and R. Mohr. "Local greyvalue invariants for image retrieval", IEEE Transactions on PAMI, PAMI Vol 19, No. 5, pages 530-534, 1997.
4. T. Lindeberg, "Feature detection with automatic scale selection", International Journal of Computer Vision, IJCV 30(2):77-116, 1998.
5. M. D. Kelly, "Edge detection by computer in pictures using planning", Machine Intelligence, 6:379-409, 1971.
6. S. L. Tanimoto and T. Pavlidis, "A hierarchical data structure for picture processing", Computer Graphics and Image Processing, 4:104-119, 1975.
7. P. J. Burt and E. H. Adelson, "The Laplacian pyramid as a compact image code", IEEE Transactions on Communications, 31:532-540, 1983.
8. J. L. Crowley, "A Representation for Visual Information", Doctoral Dissertation, Carnegie-Mellon University, 1981.
9. P. Anandan, "Measuring Visual Motion from Image Sequences", PhD thesis, Computer Science Department, Doctoral Thesis, University of Massachusetts, 1987.
10. R. Deriche. Recursively implementing the Gaussian and its derivatives. Rapport de Recherche 1893, INRIA, Sophia Antipolis, France, Apr. 1993.
11. 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.

12. D. Hall, V. Colin de Verdiere and J. L. Crowley, "Object Recognition using Coloured Receptive Field", 6$^{th}$ European Conference on Computer Vision, Springer Verlag, Dublin, pp 164-178, June 2000.

13. W.T. Freeman, E.H. Adelson, "The Design and Use of Steerable Filters", Transactions on Pattern Analysis and Machine Intelligence, (PAMI), Vol 13, No. 9, pp 891-906, September 1991.