CSSE Technical Report UWA-CSSE-09-002

Arbitrary Gaussian Filtering with 25 Additions and 5 Multiplications per Pixel

Peter Kovesi
School of Computer Science & Software Engineering
The University of Western Australia
pk@csse.uwa.edu.au

September 2009

Abstract

Summed area tables, also known as integral images, allow image averaging to be performed very efficiently at a small fixed cost per pixel, independent of averaging filter size. Repeated filtering with averaging filters can be used to approximate Gaussian filtering. Thus a good approximation to Gaussian filtering can be achieved at a fixed cost per pixel independent of filter size. This note describes how to determine the averaging filters that one needs to approximate a Gaussian with a specified standard deviation. The use of difference of Gaussians to construct bandpass filters is also analysed.

Introduction

Gaussian filtering is a fundamental process that is used in almost every computer vision application. Various techniques can be used to implement Gaussian filtering efficiently, these include exploiting the separability of the Gaussian and exploiting its symmetry as done by Canny [4], or approximating the Gaussian using recursive infinite impulse response filters, also done by Canny [4]. Image pyramids can also be used to efficiently generate multiple smoothings of an image. The use of Gaussian pyramids and the differences between layers of Gaussian pyramids to produce Laplacian pyramids was introduced by Burt and Adelson [7]. Gaussian and Laplacian pyramids have since been widely used for many applications in computer vision. This note describes how summed area tables, or integral images, can be exploited to achieve a good approximation to Gaussian filtering at a small fixed cost per pixel, independent of filter size.

Summed area tables were devised by Crow in 1984 [1] but only recently introduced to the computer vision community by Viola and Jones [2]. A summed area table, or integral image, can be generated by computing the cumulative sums along the rows of an image

and then computing the cumulative sums down the columns. Thus the value at any point (x, y) in the integral image is the sum of all the image pixels above and to the left of (x, y), inclusive.

Having computed an integral image, *int*, the sum of all the image pixels within an arbitrary rectangle, with vertices *a, b, c* and *d* as shown in Figure 1, can be computed as follows

$$\Sigma_{abcd} = int(x_c, y_c) - int(x_b, y_b) - int(x_d, y_d) + int(x_x, y_a) \ .$$

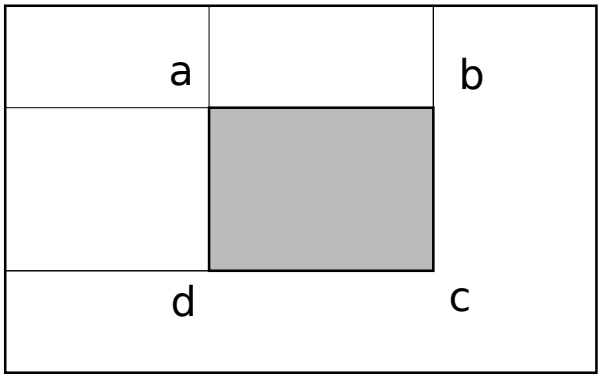

Figure 1: Computing the sum of pixel values within a rectangular area using an integral image.

If we then divide by the number of pixels in the rectangle we have an averaging filter. Two additions per pixel are required to compute the integral image, and then three additions/subtractions and a division are needed per pixel to achieve the averaging. The computational cost of this process is likely to be dominated by the cost of the memory accesses rather than the arithmetic operations.

One can use combination of these averaging/box filters to construct Haar like filters and to form crude approximations of first and second derivative Gaussian filters. This was exploited by Bay et al. with their SURF feature detector/descriptor [12]. However, one need not constrain one's thinking to the use of crude box filters. The purpose of this note is to show that high quality approximations to ideal filter shapes can be obtained via integral images at very little extra computational cost.

Repeated filtering with averaging filters can be used to approximate Gaussian filtering. Three repeated averagings achieve a passable approximation to a Gaussian and beyond four repeated averagings the approximation becomes very good. Note that if the result is to be differentiated to obtain first or second derivatives at least five filterings should be used, and perhaps even six. This is because the act of differentiation has the effect of 'rolling back' the smoothing induced by the averaging. If five filterings are used the total computational cost of the approximated Gaussian smoothing is 25 additions and 5 division operations per pixel. In principal the 5 division operations could be consolidated to a single division at the very end, however there is a risk of numerical overflow on the intermediate integral images if this is done.

Consider the crude approximation to a Hessian filter as used by the SURF feature detector shown in Figure 2. Simple approximations of the second order partial derivatives are obtained using ten box filterings. This requires 2 additions/pixel to compute the

integral image followed by 30 additions and 10 divisions/pixel to evaluate the ten box filters. However, at almost the same computational cost, one can compute a high quality approximation to a Hessian filter by performing 5 recursive averagings to obtain a Gaussian smoothing of the image and then performing discrete differentiations in the x and y directions to obtain the partial derivatives.

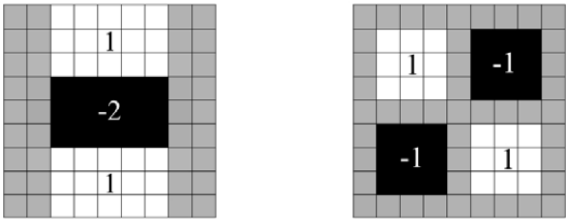

Figure 2: Box filter approximations of Gaussian second order partial derivatives used by Bay et al [12].

Determining the averaging filters needed to approximate a specific Gaussian

In order to exploit the efficiency of achieving approximate Gaussian filtering via multiple averagings one needs a way of determining the specific averaging filters required to approximate a Gaussian of a desired standard deviation.

The standard deviation of an averaging filter of width w is

$$\sigma_{av} = \sqrt{\frac{w^2 - 1}{12}}.$$

If we perform n averagings with the same filter the variances of the filters add to produce an overall filtering effect equivalent to a standard deviation of

$$\sigma_{nav} = \sqrt{\frac{nw^2 - n}{12}}.$$

From this we can compute the ideal width of the averaging filter that one should use to achieve filtering that is equivalent to that obtained with a Gaussian of standard deviation σ

$$w_{ideal} = \sqrt{\frac{12\sigma^2}{n} + 1}. \tag{1}$$

In general this will be some real valued quantity. However we can only use averaging filters of integer width and, in addition, we want to use filters of odd width so that there is always a centre pixel that the filtering result can be assigned to.

The solution adopted is to use two sizes of filter. One having width equal to the first odd integer that is less than w_{ideal}, call this w_l, and the other being being the first odd integer greater than w_{ideal}, call this w_u. Of course $w_u = w_l + 2$.

We then assume we will achieve our filtering by using m passes with a filter of width w_l followed by $(n - m)$ passes with a filter of width w_u, where $0 \leq m \leq n$. Again, noting

that variances of the filters add, the standard deviation of the equivalent filter obtained from this process will be

$$\begin{aligned}\sigma &= \sqrt{\frac{mw_l^2 + (n-m)w_u^2 - n}{12}} \\ &= \sqrt{\frac{mw_l^2 + (n-m)(w_l+2)^2 - n}{12}} \\ &= \sqrt{\frac{-m(4w_l+4) + nw_l^2 + 4nw_l + 3n}{12}}\end{aligned}$$

Thus, given σ, n and w_l we can solve for the value of m as

$$m = \frac{12\sigma^2 - nw_l^2 - 4nw_l - 3n}{-4w_l - 4} \tag{2}$$

Note that m has to be an integer so the value from the expression above has to be rounded to the nearest integer.

In summary, given σ and n, the overall process is:

- Use equation 1 to determine w_{ideal} and hence w_l and w_u.
- Use equation 2 to determine m.
- Apply an averaging filter of width w_l m times.
- Apply an averaging filter of width w_u $(n-m)$ times.

The rounding of m to an integer will introduce a small error in the effective standard deviation achieved. The larger the value of n the smaller this error will be. For example for $n = 5$, $w_l = 3$ and $w_u = 5$ if $m = 5$ the standard deviation will be

$$\sigma_{m=5} = \sqrt{\frac{5 \times 3^2 + 0 \times 5^2 - 5}{12}} = 1.8257$$

The next increment in sigma will be obtained with $m = 4$

$$\sigma_{m=4} = \sqrt{\frac{4 \times 3^2 + 1 \times 5^2 - 5}{12}} = 2.1602$$

The difference between these two values is 0.3345. For smaller values of m, and for larger values of w_l, this difference reduces gradually, hence one can expect the accuracy of the achieved σ to be at least as good as ± 0.1673 for $n = 5$.

While the accuracy of the standard deviation achieved can be improved by increasing n it is worth keeping n as small as is practical to reduce the edge effects in the final filtered result. With each averaging filter pass the edge effects propagate further into the image. If we define the 'radius' of an averaging filter as being (width-1)/2, then on the first filtering pass all pixels less than or equal to this distance from the boundary will be affected by the edge. On the second pass all pixels within this distance from the previously edge affected pixels will also be affected, and so on. Thus the total edge affected boundary will be of width $n \times$ radius.

For example consider the filtering achieved by using 5 passes of an averaging filter 7 pixels wide. This filtering process is equivalent to applying a Gaussian filter of standard deviation

$$\sigma = \sqrt{\frac{5 \times 7^2 - 5}{12}} = 4.47$$

The radius of an ideal Gaussian filter constructed to this size will be $3\sigma \simeq 13$ pixels.

In comparison the radius of the averaging filter of width 7 is 3, so the edge affected boundary obtained by repeated averagings will be of width $5 \times 3 = 15$ pixels. This is larger than that of an ideal Gaussian filter and will grow with an increased number of passes at a rate faster then the equivalent Gaussian standard deviation would also grow. So, overall it is worth keeping the number of passes small, certainly no more than 6.

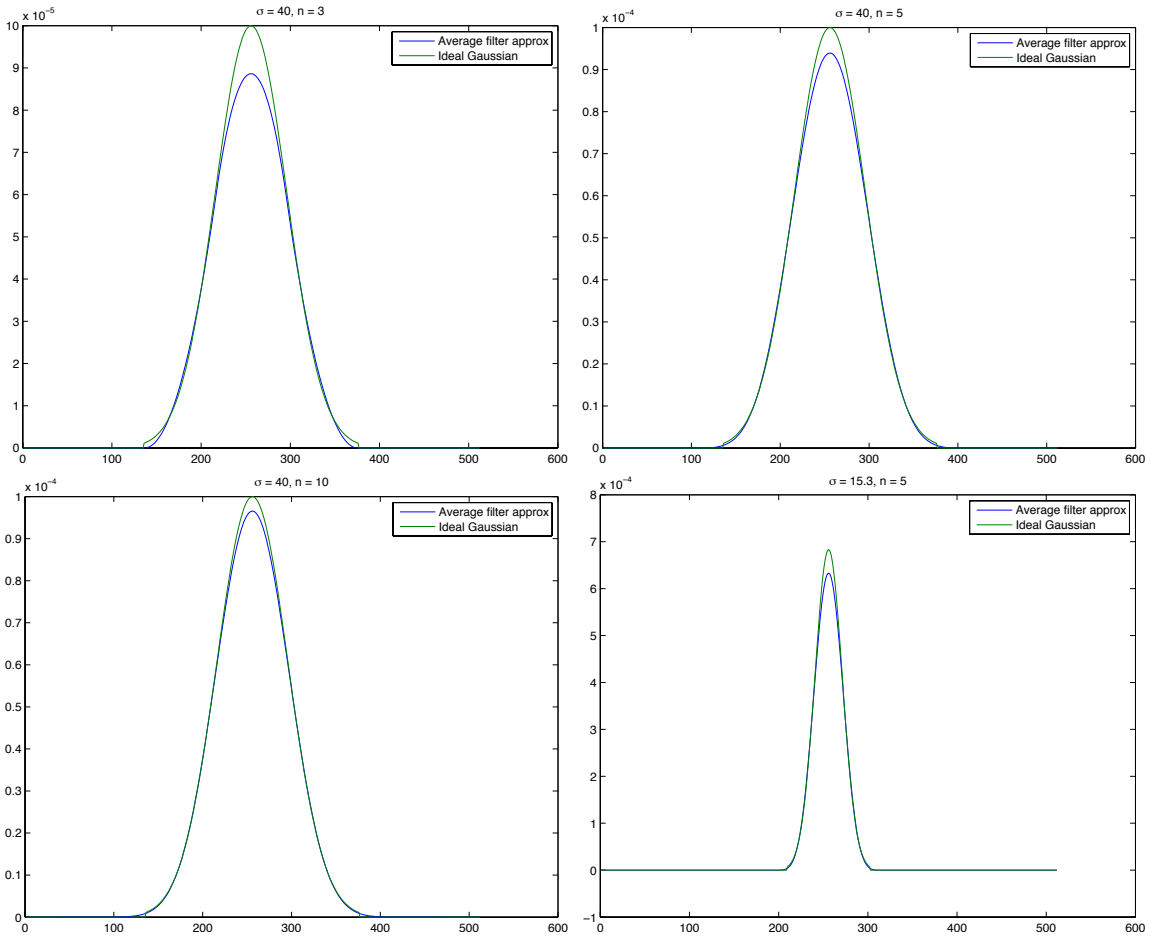

Figure 3: The first 3 plots show approximating a Gaussian with standard deviation of 40 using 3, 5 and 10 averaging passes. Actual standard deviation achieved with 5 passes was 39.983. Note that as the number of passes increases the peak of the Gaussian is represented with greater fidelity. The fourth plot shows approximation of a standard deviation of 15.3 using 5 averaging passes. Actual standard deviation achieved was 15.362.

Designing Gaussian Lowpass, Highpass and Bandpass Filters

Having the ability to generate equivalent Gaussian filters at arbitrary standard deviations allows one to design filters with specific properties in the frequency domain.

The Fourier transform of a normalized Gaussian is a unit height Gaussian.

$$\mathcal{F}_x\left[\frac{1}{\sqrt{2\pi\sigma^2}}\, e^{\frac{-x^2}{2\sigma^2}}\right](\omega) = e^{-2\pi^2\sigma^2 w^2} \tag{3}$$

Thus the standard deviations of the Gaussians in the spatial and frequency domains are related as follows:

$$\sigma_x = \frac{1}{2\pi\sigma_\omega}\ . \tag{4}$$

Filtering an image with a Gaussian in the spatial domain corresponds to lowpass filtering. Subtracting a lowpassed image from the original image corresponds to highpass filtering. For symmetry I prefer to define the cutoff frequency of the filter as being the point of half maximum amplitude rather than half maximum power. Using this definition one would use the same Gaussian filter to achieve either lowpass filtering or to generate a highpass filtering for a given cutoff frequency. In contrast, depending on whether one was performing highpass or lowpass filtering, different Gaussians would be needed if one was to use the traditional factor of $\sqrt{\frac{1}{2}}$ of the maximum to define the cutoff.

At half amplitude the cutoff frequency corresponds to

$$\omega_c = \sigma_\omega\sqrt{2\ln(2)}\ .$$

Thus, given a desired ω_c, one can compute the appropriate value of σ_x to use when performing the Gaussian smoothing in the spatial domain

$$\sigma_x = \frac{\sqrt{2\ln(2)}}{2\pi\omega_c}\ .$$

Difference of Gaussians Bandpass Filters

The difference of two Gaussians can be used to form a bandpass filter. Traditionally differences of Gaussians have been used to approximate the Laplacian of Gaussian. A ratio of standard deviations of 1.6 produces a very good approximation to the Laplacian [6, 5]. The use of Gaussian pyramids and the differences between layers of Gaussian pyramids to produce Laplacian pyramids was introduced by Burt and Adelson [7]. Gaussian and Laplacian pyramids have since been widely used for many applications in computer vision. Lindeberg [11]. For example Lowe used differences between levels of a Gaussian pyramid to generate a scale space for the detection of features [3].

In many applications 'Laplacian' pyramids are formed from the differences of Gaussians that differ considerably from the ideal standard deviation ratio of 1.6. In general this does not matter because usually all that is required is some kind of bandpass filter, not the Laplacian of Gaussian.

Thus, rather than just thinking of differences of Gaussians as being a way to approximate the 2nd derivative of a Gaussian it is probably more useful think of them as forming a family of bandpass filters.

The filters can be designed in the frequency domain as being the difference of two unit height Gaussians. The spatial equivalent of these unit height Gaussians defined in the frequency domain can then be determined using equation 4. In many applications we are

interested in forming geometrically scaled filters where the bandwidth is proportional to the centre frequency. This can be achieved if we specify, in the frequency domain, the standard deviation of the broader Gaussian to be some constant, k, times the smaller one, giving

$$H_{bp} = e^{\frac{-w^2}{2k^2\sigma_\omega^2}} - e^{\frac{-w^2}{2\sigma_\omega^2}} .$$

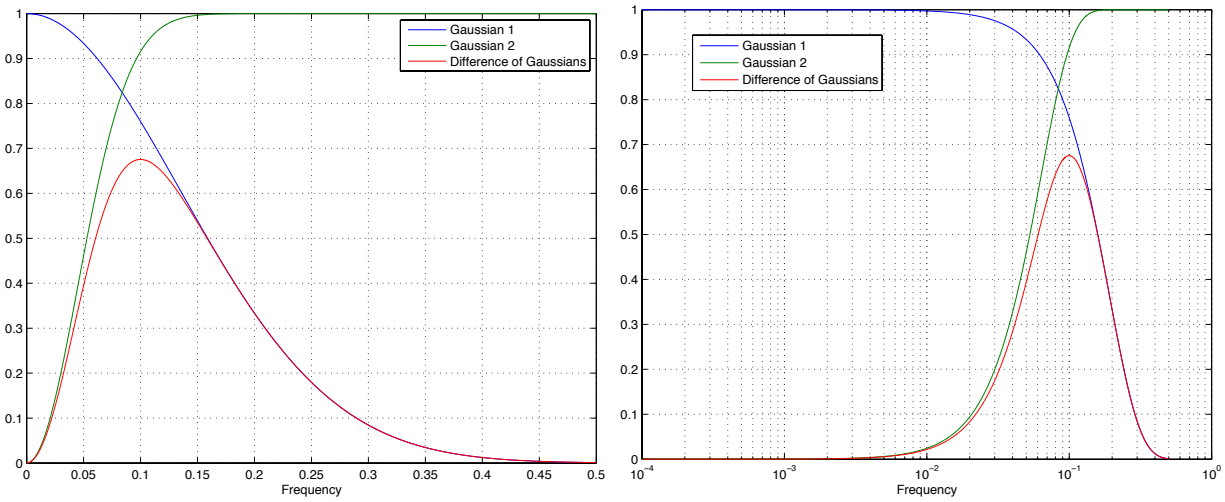

Figure 4: Using the difference of two unit height Gaussians to construct a bandpass filter in the frequency domain. The transfer function of the filter is shown on linear and logarithmic frequency scales. The center frequency is 0.1 and the ratio of standard deviations is 3.

As seen in Figure 4 the transfer function of the bandpass filter produced by a difference of Gaussians has a long tail towards the high frequency end. The extent of the tail depends on the value of k. On a logarithmic frequency scale the shape is more symmetric. In this case, where $k = 3$, the shape is not too dissimilar to the transfer function of a log-Gabor filter [9, 8].

If one defines the centre frequency of this filter as being the point where it peaks this can be solved for as being the point where H_{bp} has zero slope

$$\omega_o = 2\sigma_\omega \sqrt{\frac{\ln(k)}{1 - \frac{1}{k^2}}}$$

Alternatively, given a desired centre frequency and a value of k one can solve for the necessary standard deviations as follows

$$\sigma_l = \frac{\omega_o}{2} \sqrt{\frac{1 - \frac{1}{k^2}}{\ln(k)}} \quad (5)$$

$$\sigma_u = k\sigma_l \quad (6)$$

The value of the transfer function at the peak will be

$$M = e^{\frac{-\omega_o^2}{2k^2\sigma_l^2}} - e^{\frac{-\omega_o^2}{2\sigma_l^2}} .$$

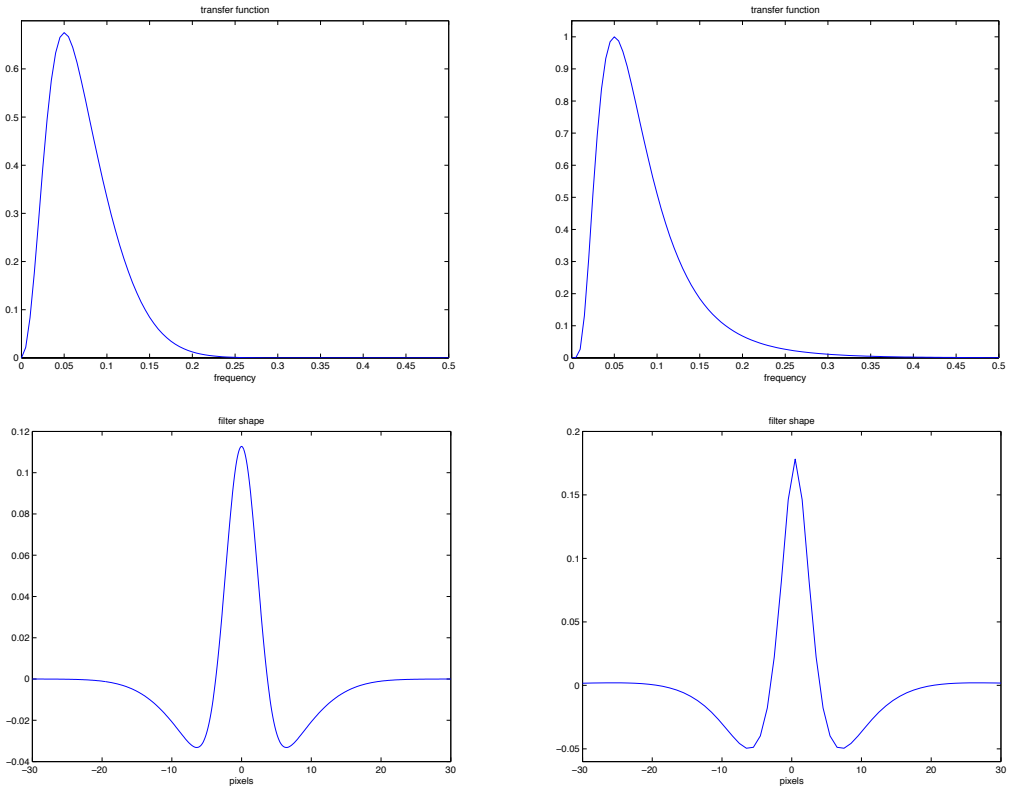

Figure 5: Transfer function and filter shape of difference of Gaussian filter (left) and log Gabor filter (right). Both have a centre frequency of 0.05 (wavelength 20 pixels) and a bandwidth of approximately 1.5 octaves.

The value of k controls the bandwidth of the filter. When measured in terms of the ratio of the upper cutoff frequency to the lower cutoff the bandwidth will be fixed for a given value of k for any value of centre frequency.

The relationship between k and the bandwidth cutoff ratio is non-linear and does not lend itself to analytic solution. Figure 6 plots a numerically evaluated curve showing the relationship between k and the filter bandwidth. There is a lower limit of the bandwidth of a difference of Gaussians filter. As the value of k tends towards 1, and the difference of Gaussians collapse towards 0, the bandwidth cutoff ratio tends to about 2.3.

The relationship between k and bandwidth ratio, B can be approximated by a fitted polynomial

$$k = c_1 B^2 + c_2 B + c_3$$

where $c_1 = -0.6775$, $c_2 = 6.8417$ and $c_3 = -10.9043$.

Given a desired centre frequency ω_o and bandwidth ratio, and hence k, one can compute the standard deviations of the constructing Gaussians in the frequency domain using Equations 5 and 6. These can then be converted to standard deviations of the Gaussian filters to apply in the spatial domain using Equation 4.

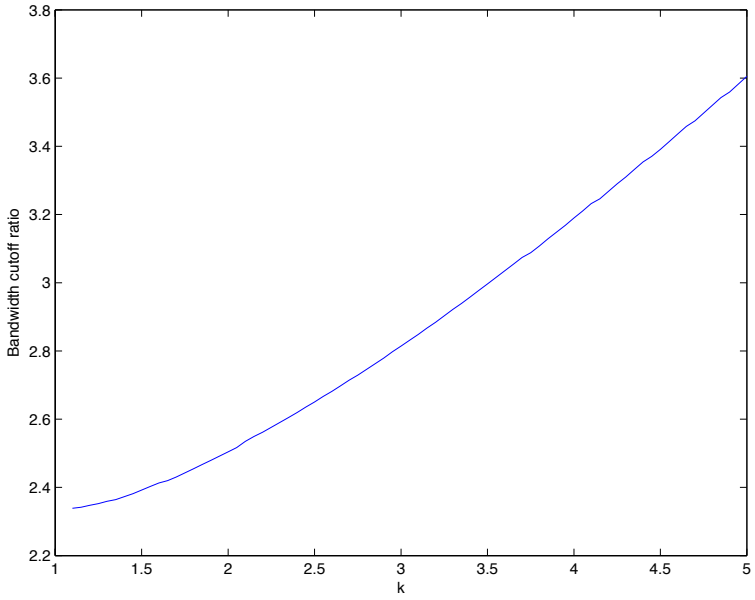

Figure 6: Plot of filter bandwidth cutoff ratio vs ratio of Gaussian standard deviations k.

References

[1] Franklin Crow, "Summed-area tables for texture mapping". *SIGGRAPH '84: Proceedings of the 11th annual conference on Computer graphics and interactive techniques*. pp. 207–212. 1984

[2] Paul Viola and Michael Jones, "Robust Real-time Object Detection". *International Journal of Computer Vision* 2002.

[3] David G. Lowe, "Distinctive image features from scale-invariant keypoints", *International Journal of Computer Vision*, 60, 2, pp. 91–110. 2004.

[4] J. F. Canny, "Finding Edges and Lines in Images". Masters Thesis MIT. AI Lab. TR-720, 1983.

[5] D. Marr, *Vision*. Freeman: San Francisco. 1982.

[6] D. Marr and E. C. Hildreth, "Theory of edge detection". *Proceedings of the Royal Society, London B* Vol. 207, pp. 187–217. 1980.

[7] Peter Burt and Ted Adelson, "The Laplacian Pyramid as a Compact Image Code", *IEEE Trans. Communications*, 9:4, pp. 532–540, 1983.

[8] D. J. Field, "What the Statistics of Natural Images Tell us About Visual Coding". *SPIE 1077*, Los Angeles, California, pp. 269–276, 1989.

[9] D. J. Field, "Relations Between the Statistics of Natural Images and the Response Properties of Cortical Cells". *Journal of The Optical Society of America A*, Vol. 4, No 12, pp. 2379–2394, 1987.

[10] A. Witkin, D. Terzopoulos and M. Kass, "Signal Matching Through Scale Space". *International Journal of Computer Vision*, pp. 133–144, 1987.

[11] Tony Lindeberg, "Scale-space for discrete signals", *PAMI* (12), No. 3, pp. 234–254, 1990.

[12] Herbert Bay, Andreas Ess, Tinne Tuytelaars and Luc Van Gool, "SURF: Speeded Up Robust Features", *Computer Vision and Image Understanding (CVIU)*, Vol. 110, No. 3, pp. 346–359, 2008.