Fast Almost-Gaussian Filtering

Peter Kovesi
Centre for Exploration Targeting
School of Earth and Environment
The University of Western Australia
35 Stirling Highway
Crawley WA 6009
Email: peter.kovesi@uwa.edu.au

***Abstract*—Image averaging can be performed very efficiently using either separable moving average filters or by using summed area tables, also known as integral images. Both these methods allow averaging to be performed at a small fixed cost per pixel, independent of the 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 paper describes how to determine the averaging filters that one needs to approximate a Gaussian with a specified standard deviation. The design of bandpass filters from the difference of Gaussians is also analysed. It is shown that difference of Gaussian bandpass filters share some of the attributes of log-Gabor filters in that they have a relatively symmetric transfer function when viewed on a logarithmic frequency scale and can be constructed with large bandwidths.**

***Index Terms*—Gaussian smoothing; Difference of Gaussian filtering;**

I. Introduction

Gaussian smoothing is a fundamental process that is used in almost every computer vision application. It forms the cornerstone of scale-space theory [1], [2], [3], [4], [5]. With the increasing resolution of images there is an interest in minimising Gaussian filtering costs. In addition, with higher resolution images one is also often wanting to use Gaussian filters with correspondingly larger standard deviations. Various techniques can be used to implement Gaussian filtering efficiently, these include exploiting the separability and symmetry of the Gaussian as done by Canny [6], or approximating the Gaussian using recursive infinite impulse response filters, as investigated by by Canny [6], Deriche [7], and Young and van Vliet [8]. Haddad and Akansu [9] employed Binomial filters to achieve fast near–Gaussian filtering. 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 [10]. Gaussian and Laplacian pyramids have since been widely used for many applications in computer vision. More recently attention has been directed at the efficient implementation of anisotropic Gaussian filtering [11].

Of all the techniques mentioned above only the approaches based on recursive IIR filters are capable of achieving filtering at a fixed cost per pixel independent of the Gaussian size. This paper describes how the fixed low cost of averaging achieved through separable moving average filters, or via summed area tables, can be exploited to achieve a good approximation to Gaussian filtering also at a small fixed cost per pixel, independent of filter size.

Summed area tables were devised by Crow in 1984 [12] but only recently introduced to the computer vision community by Viola and Jones [13]. 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, S, 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

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

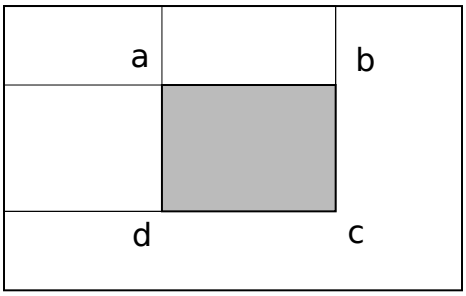

Fig. 1. Computing the sum of pixel values within a rectangular area using an integral image.

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 combinations 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 [14].

However, one need not constrain one's thinking to the use of crude box filters. The purpose of this paper 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.

An alternative to the use of integral images is to exploit the separability of the averaging filter and perform repeated moving average filtering on the rows and columns of the image. If some of the division operations on the repeated averaging passes are consolidated the computational cost can be made similar to that of using integral images.

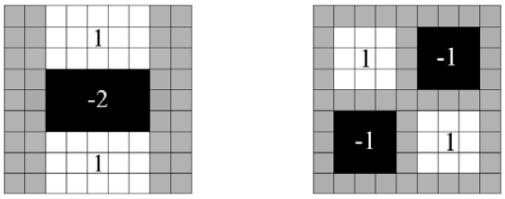

Fig. 2. Box filter approximations of Gaussian second order partial derivatives used by Bay *et al.* [14].

II. 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

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

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

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

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

$$w_{ideal} = \sqrt{\frac{12\sigma^2}{n} + 1}. \tag{3}$$

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

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

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

$$m = \frac{12\sigma^2 - nw_l^2 - 4nw_l - 3n}{-4w_l - 4}. \tag{5}$$

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

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

- Use equation 3 to determine w_{ideal} and hence w_l and w_u.
- Use equation 5 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, the spacing between adjacent filter standard deviations 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, the width of the edge affected boundary will be $n \times \text{radius}$. For $n = 5$ this boundary width will be slightly greater than the 3σ that is typically allowed for in Gaussian smoothing. So, overall it is worth keeping the number of passes small, certainly no more than 6.

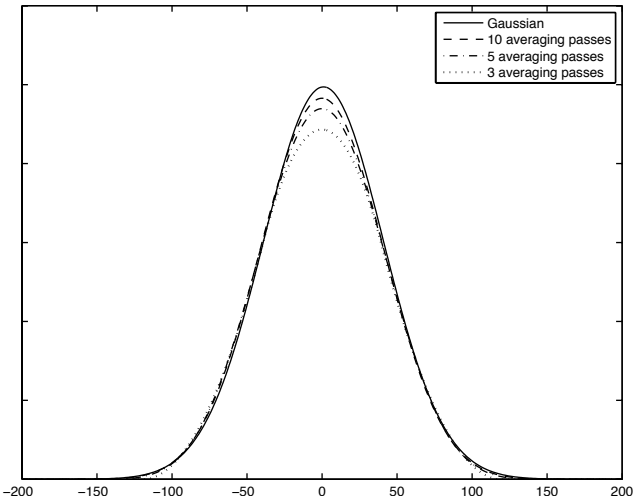

Fig. 3. Approximating a Gaussian with standard deviation of 40 using 3, 5 and 10 averaging passes. Actual standard deviation achieved with 5 passes was 39.983.

III. 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.

$$\mathcal{F}_x \left[\frac{1}{\sqrt{2\pi\sigma^2}} \, e^{\frac{-x^2}{2\sigma^2}} \right] (\omega) = e^{-2\pi^2\sigma^2 w^2} \tag{6}$$

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

$$\sigma_x = \frac{1}{2\pi\sigma_\omega} \ . \tag{7}$$

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 one may 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 generate either a lowpass or 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 $\sqrt{1/2}$ of the maximum to define the cutoff.

At half amplitude the cutoff frequency corresponds to

$$\omega_c = \sigma_\omega \sqrt{2\ln(2)} \ . \tag{8}$$

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

$$\sigma_x = \frac{\sqrt{2\ln(2)}}{2\pi\omega_c} \ . \tag{9}$$

A. Difference of Gaussians Bandpass Filters

The difference of two Gaussians can be used to form a bandpass filter. Traditionally differences of Gaussians have been thought of in terms of approximating the Laplacian of Gaussian. A ratio of standard deviations of 1.6 produces a good approximation to the Laplacian [15], [16]. The use of differences between layers of Gaussian pyramids to produce Laplacian pyramids, as introduced by Burt and Adelson [10], have become widely used for many applications in computer vision. A recent example is Lowe's use of differences between levels of a Gaussian pyramid to generate a scale space for the detection of keypoint features [17].

However, there are many examples in the literature of 'Laplacian' pyramids being formed from the differences of Gaussians that differ considerably from the ideal standard deviation ratio of 1.6. Indeed, Burt and Adelson acknowledge this in their paper and only describe their Gaussian difference images as being similar to the Laplacian operator. Lowe uses Gaussians that differ in scale by $2^{1/3} \simeq 1.26$ [17]. The use of these 'non-Laplacian' pyramids generally 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 second derivative of a Gaussian it is probably more useful think of them as forming a family of bandpass filters. They form a useful alternative to the use of Gabor filters in that they are guaranteed to have 0 DC component and can be constructed with much larger bandwidths.

Difference of Gaussian bandpass 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 7. 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

$$H_{bp} = e^{\frac{-w^2}{2k^2\sigma_\omega^2}} - e^{\frac{-w^2}{2\sigma_\omega^2}} \ . \tag{10}$$

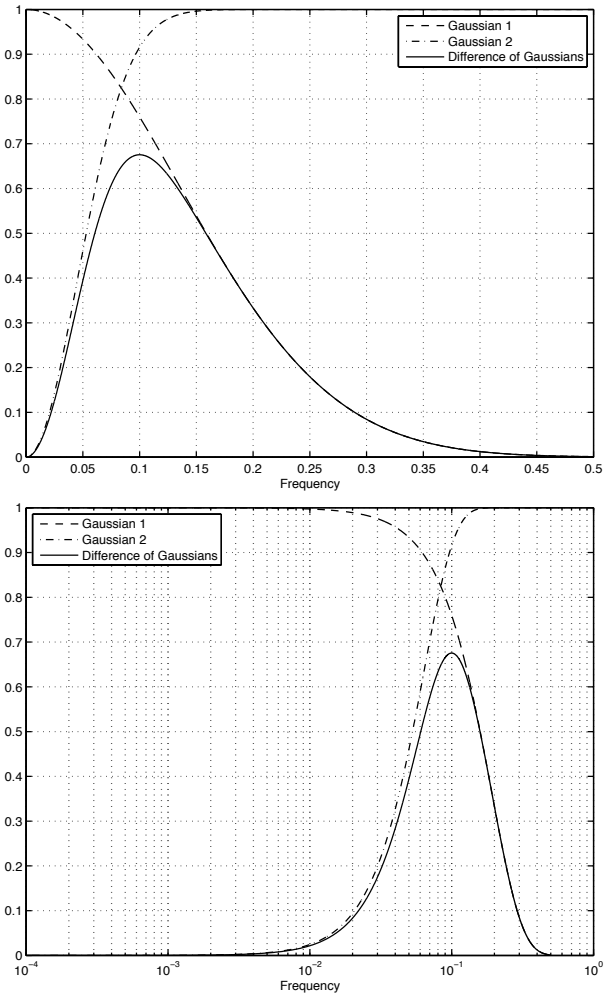

Fig. 4. Using the difference of two unit height Gaussians to construct a bandpass filter in the frequency domain. The transfer function of the filter is shown on linear and logarithmic frequency scales. The center frequency is 0.1 and the ratio of standard deviations is 3.

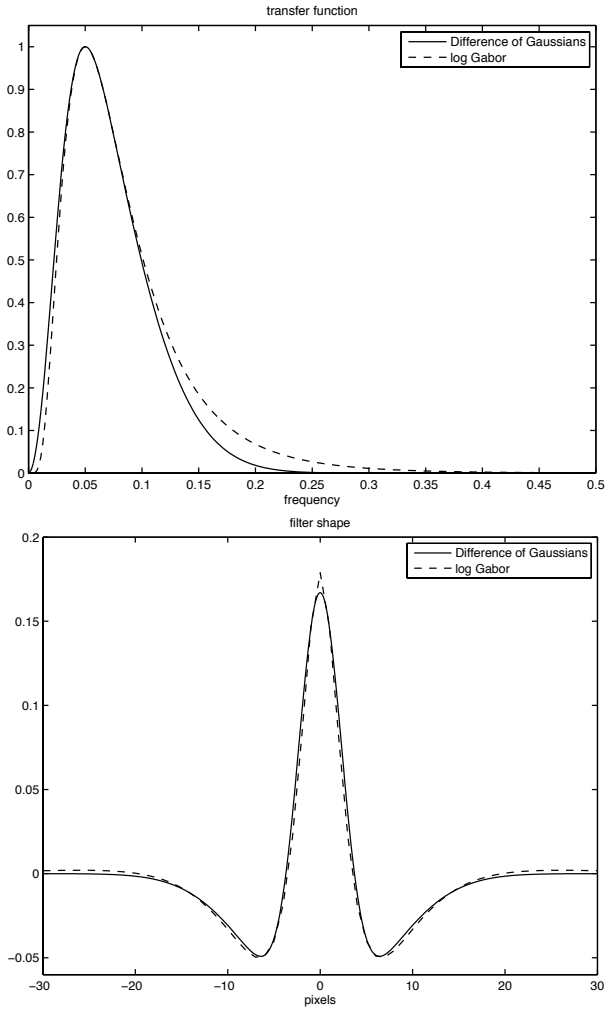

Fig. 5. Transfer function and filter shape of difference of Gaussian filter and log-Gabor filter. The difference of Gaussian filter was constructed with a ratio of standard deviations of 3. The log-Gabor filter was constructed to have a similar bandwidth to allow comparison. Both have a centre frequency of 0.05 (wavelength 20 pixels) and a bandwidth of approximately 1.5 octaves.

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 [18], [19], which has a Gaussian transfer function when viewed on a logarithmic frequency scale.

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

$$\omega_o = 2\sigma_\omega \sqrt{\frac{\ln(k)}{1 - \frac{1}{k^2}}} \tag{11}$$

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

$$\sigma_l = \frac{\omega_o}{2} \sqrt{\frac{1 - \frac{1}{k^2}}{\ln(k)}} \tag{12}$$

$$\sigma_u = k\sigma_l \tag{13}$$

The value of the transfer function at the peak will be

$$M = e^{\frac{-\omega_o^2}{2k^2\sigma_l^2}} - e^{\frac{-\omega_o^2}{2\sigma_l^2}} \quad . \tag{14}$$

If required, the transfer function of the bandpass filter can then be renormalised back to unit height by scaling the Gaussians by $1/M$.

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 readily 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. Note that, unlike the choice made in Section III for defining lowpass and highpass filters, the cutoff frequencies are defined here by the points on the transfer function that are at half maximum power.

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

$$k = c_1 B^2 + c_2 B + c_3 \tag{15}$$

where $c_1 = -0.8855$, $c_2 = 8.1259$ and $c_3 = -12.8646$.

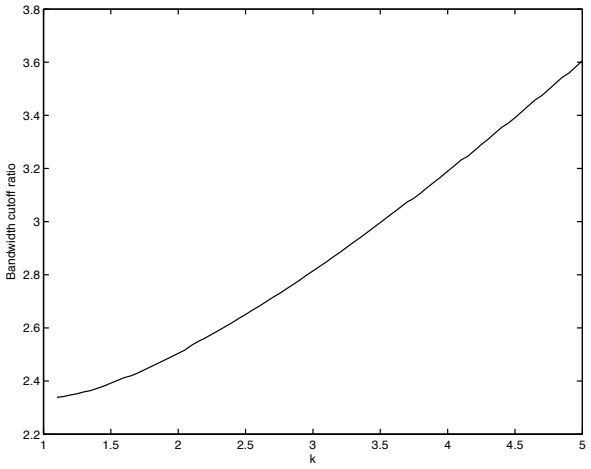

Fig. 6. Plot of filter bandwidth cutoff ratio vs ratio of Gaussian standard deviations k.

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 12 and 13. These can then be converted to standard deviations of the Gaussian filters to apply in the spatial domain using Equation 7. Equation 14 can be used to determine the scaling of the Gaussians needed to ensure a unit height of the filter's transfer function.

It is interesting to note that the bandwidth of a difference of Gaussian filter is limited to frequency ratios of about 2.4 and upwards. Thus they are not appropriate for applications requiring narrow bandpass filters. This is in contrast to the Gabor filter which is limited to a *maximum* bandwidth ratio of about 2. Larger bandwidth Gabor filters cannot be constructed because the DC value can no longer be held at 0 [20]. On the other hand log-Gabor filters [18], [19] can be constructed with arbitrary bandwidth. It is for this reason the comparison with a log-Gabor filter shown in Figure 5 is presented. For applications requiring large bandwidth filters difference of Gaussian filters appear to share many of the attributes of log-Gabor filters. The transfer function, when viewed on a logarithmic frequency scale, is relatively symmetric and arbitrarily large bandwidths can be constructed. Difference of Gaussians have the convenience of being applied to the image in the spatial domain whereas log-Gabor filters can only be readily applied if the image has first been transformed to the frequency domain.

IV. Conclusion

This paper shows that there is no computational justification for using crude box filters to approximate Gaussians and their derivatives in image processing. High quality approximations can be obtained at negligible cost by repeated averagings. The appropriate averaging filters required to closely achieve a desired Gaussian standard deviation can be determined readily.

It has also been argued that is is more useful to think of differences of Gaussians as bandpass filters rather than as approximations of the Laplacian. They are useful for applications requiring large bandwidth filters and thus provide a useful complementary alternative to Gabor filters. Difference of Gaussian bandpass filters can be implemented efficiently and have the advantage over log-Gabor filters in that they can be applied directly in the spatial domain.

References

[1] J. J. Koenderink, "The structure of images," *Biological Cybernetics*, vol. 50, no. 5, pp. 363–370, August 1984.

[2] T. Lindeberg, "Scale-space for discrete signals," *IEEE Transactions PAMI*, vol. 12, no. 3, pp. 234–254, 1990.

[3] ——, *Scale-Space Theory in Computer Vision*. Kluwer, 1994.

[4] ——, "Feature detection with automatic scale selection," *International Journal of Computer Vision*, vol. 30, no. 2, pp. 79–116, 1998.

[5] A. Witkin, D. Terzopoulos, and M. Kass, "Signal matching through scale space," *International Journal of Computer Vision*, pp. 133–144, 1987.

[6] J. F. Canny, "Finding edges and lines in images," MIT. AI Lab., Tech. Rep. TR-720, 1983, masters Thesis.

[7] R. Deriche, "Fast algorithms for low-level vision," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, pp. 78–87, 1990.

[8] I. T. Young and L. J. van Vliet, "Recursive implementation of the gaussian filter," *Signal Processing*, vol. 44, pp. 139–151, 1995.

[9] R. A. Haddad and A. N.Akansu, "A class of fast gaussian binomial filters for speech and image processing," *IEEE Transactions on Signal Processing*, vol. 39, no. 3, pp. 723–727, 1991.

[10] P. Burt and T. Adelson, "The Laplacian pyramid as a compact image code," *IEEE Trans. Communications*, vol. 9, no. 4, pp. 532–540, 1983.

[11] J.-M. Geusebroek, A. W. M. Smeulders, and J. van de Weijer, "Fast anisotropic gauss filtering," *IEEE Transactions on Image Processing*, vol. 12, no. 8, pp. 938–943, 2003.

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

[13] P. Viola and M. Jones, "Robust real-time object detection," in *Second International Workshop on Statistical and Computational Theories of Vision Modeling, Learning, Computing, and Sampling*, 2001, vancouver.

[14] H. Bay, A. Ess, T. Tuytelaars, and L. V. Gool, "SURF: Speeded Up Robust Features," *Computer Vision and Image Understanding*, vol. 110, no. 3, pp. 346–359, 2008.

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

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

[17] D. G. Lowe, "Distinctive image features from scale-invariant keypoints," *International Journal of Computer Vision*, vol. 60, no. 2, pp. 91–110, 2004.

[18] 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.

[19] ——, "What the statistics of natural images tell us about visual coding," in *SPIE Vol 1077: Human Vision, Visual Procsssing and Digital Display*, 1989, pp. 269–276.

[20] P. Kovesi, "Image features from phase congruency," *Videre*, vol. 1, no. 3, pp. 1–26, 1999.