A Fast Image Sensor System with an Efficient Multi-Scale Gaussian Filtering Circuit

Yuki Yamaji
Graduate School of Information Science and Technology, Osaka Institute of Technology,
1-79-1 Kitayama Hirakata, Osaka 573-0196, Japan

Akito Morita
Graduate School of Information Science and Technology, Osaka Institute of Technology,
1-79-1 Kitayama, Hirakata, Osaka 573-0196, Japan

Hirotsugu Okuno
Faculty of Information Science and Technology, Osaka Institute of Technology,
1-79-1 Kitayama, Hirakata, Osaka 573-0196, Japan
E-mail: hirotsugu.okuno@oit.ac.jp
www.oit.ac.jp

Abstract

We designed an efficient multi-scale Gaussian filtering circuit whose coefficients of the standard deviation are selectable from any multiple of the square root of two. We also developed an image sensor system composed of a CMOS image sensor and a field-programmable gate array that contains the proposed filtering circuit. The system provided eight images whose resolution is 160 × 120 filtered by different scales of Gaussian filters at 156 frames / second.

Keywords: multi-scale, Gaussian filter, FPGA, image sensor, bio-inspired

1. Introduction

Efficient extraction of multi-scale visual features is required for a wide range of image processing systems. In particular, many useful bio-inspired visual processing algorithms, such as saliency-based visual attention[1] and the scale-invariant feature transform[2], rely on multi-scale Gaussian and/or Gabor filters because the early stages of visual nervous system are modeled as such filters[3,4] based on physiological studies (Refs. 5 and 6 for examples).

In the visual nervous system, the spatial property modeled as the Gaussian and the difference of Gaussians (DoG) with multiple scales is implemented in the retina[3], and all the visual functions are achieved using the output signal of the retina. Therefore, multi-scale Gaussian filtering is an essential component for implementing a wide range of vision-based tasks. Although the computation itself of spatial filtering is simple, the computational cost required for spatial filters with a large kernel is very high because the computation contains a large number of multiply-accumulate (MAC) operations. One of the most widely used solutions for this problem is separating the two-dimensional (2D) filter into two one-dimensional (1D) filters when the filter kernel is separable. Several studies on hardware implementation of 2D filters also use the technique to achieve efficient filtering architecture (Refs. 7 and 8 for examples).

The separable filter technique is applicable to multi-scale Gaussian filtering. In addition to the technique, the recursive algorithm proposed by Burt can further reduce the computational cost of filtering.[9] In the case of hardware implementation, the algorithm is an effective way to save the hardware resources required.

In the present study, we designed a multi-scale Gaussian filtering circuit based on the recursive filtering algorithm[9]. In order to save hardware resources, we used fixed-point computation, and investigated the adequate bit width with which errors are negligible even after repetitive recursive filtering. We also developed an image sensor system composed of a CMOS image sensor and a field-programmable gate array (FPGA) that contains the proposed filtering circuit.

2. Filtering algorithm

2.1. *Fast recursive filtering algorithm*

The fast recursive filtering algorithm implemented in this study is based on the algorithm described in Ref. 9. The algorithm applies separated 1D filters whose kernel is very simple to the input image. A set of kernels that differ in width but not in weights are convolved recursively. Fig. 1(a) shows how the kernel is convolved with the input data for the first two scales. The basic size of the kernel is five, and therefore, only five MAC operations are needed for 1D filtering in each scale. The kernel with the doubled spacing between two weights (d in Fig. 1(a)) is applied to the output image of the previous filter to increase the scale by one. One scale increment doubles the standard deviation of the Gaussian filter.

2.2. *Algorithm implemented in the circuit*

In order to enable configuration of the standard deviation of the Gaussian with a smaller step, we modified the recursive filtering algorithm described in section 2.1. Fig. 1(b) shows the kernels of the first three scales of the modified algorithm. Different from the original algorithm, the basic size of the kernel of the modified algorithm is three, and one-scale increment in the modified algorithm corresponds to multiplying the standard deviation by the square root of two. The spacing (d in Fig. 1(b)) is expressed as:

$$d = 2^m \tag{1}$$

$$m = floor\left(\frac{n-1}{2}\right) \tag{2}$$

where n represents the scale. The following two sets of weight values are employed depending on the scale:

$$(w_0, w_1) = \begin{cases} (a_1, b_1) (n \text{ is odd}) \\ (a_2, b_2) (n \text{ is even}) \end{cases} \tag{3}$$

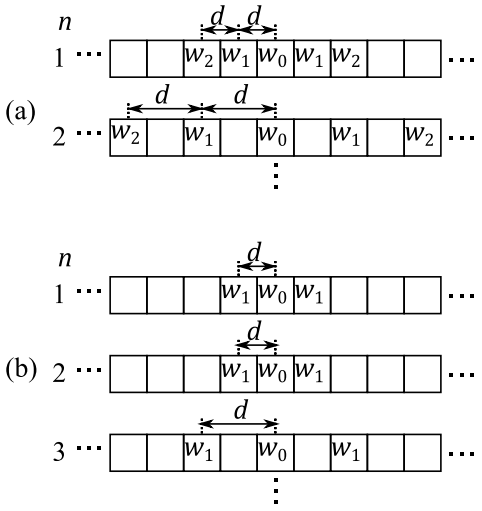

Fig. 1. Relationship between scale $\boldsymbol{n}$ and a set of kernels. (a) Original algorithm. (b) Modified algorithm.

The following constraints should be met for normalization:

$$a_1 + 2b_1 = 1 \tag{4}$$

$$a_2 + 2b_2 = 1 \tag{5}$$

The following set of parameters was used in this study:

$$(a_1, b_1) = (0.625, 0.1875) \tag{6}$$

$$(a_2, b_2) = (0.5, 0.25) \tag{7}$$

In the circuit implemented in this study, the filters are applied using fixed point computation to save hardware resources. An appropriate bit width was examined using the simulation described in section 3.

3. Simulation

3.1. *Filter kennel*

We examined the impulse response of the modified Gaussian filtering algorithm by using Python. Fig. 2 shows the impulse response of the filter whose scales are 1, 2, 3, and 4. The dashed lines plot fitted Gaussian functions. Gaussian functions are well-approximated by the weights achieved by the algorithm.

Table 1 shows the standard deviations of the fitted Gaussian functions. The ratio of the standard deviation of the adjacent scales is approximated by the square root of two, for a larger scale in particular.

Table. 1 Standard deviations of the fitted Gaussian functions.

scale n	1	2	3	4	5
σ_n	0.639	1.057	1.482	2.201	3.128
σ_n/σ_{n-1}	-	1.654	1.402	1.485	1.421

scale n	6	7	8	9	10
σ_n	4.467	6.322	8.971	12.694	17.964
σ_n/σ_{n-1}	1.428	1.415	1.419	1.415	1.415

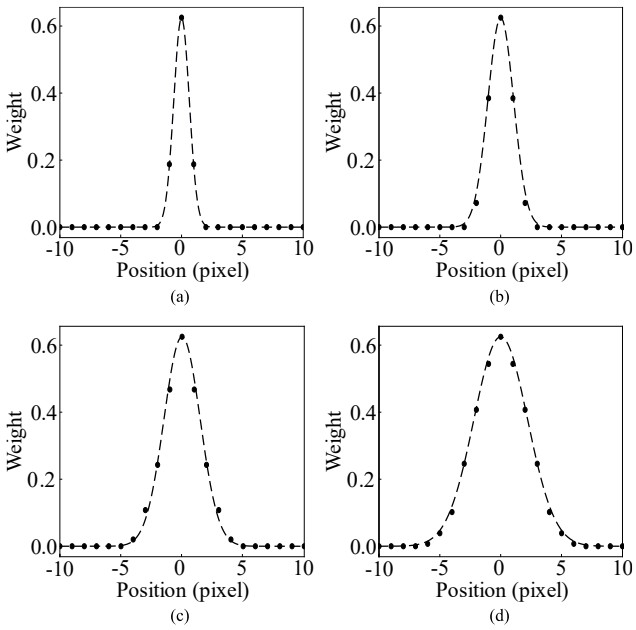

Fig. 2. Impulse responses of the Gaussian filters. Solid black circles represent the weight and the dashed line plots the fitted curve of the Gaussian function. (a)(b)(c) and (d) shows the weights of the filters whose scales are 1, 2, 3, and 4, respectively.

3.2. *Bit width*

We examined an appropriate bit width for implementing the modified Gaussian filtering algorithm by comparing the error between the filtered image computed with double-precision floating point and that computed with a fixed point. In this experiment, the bit width of the input and output images was eight, and the bit width with which the accumulation of the error was negligible was investigated. Here, the round-to-even method was employed. We chose several images from SIDBA[10] as an input image in this experiment. Fig. 3(a) shows an example of the relationship between the scale and the maximum error in the filtered image. Although Fig. 3(a)

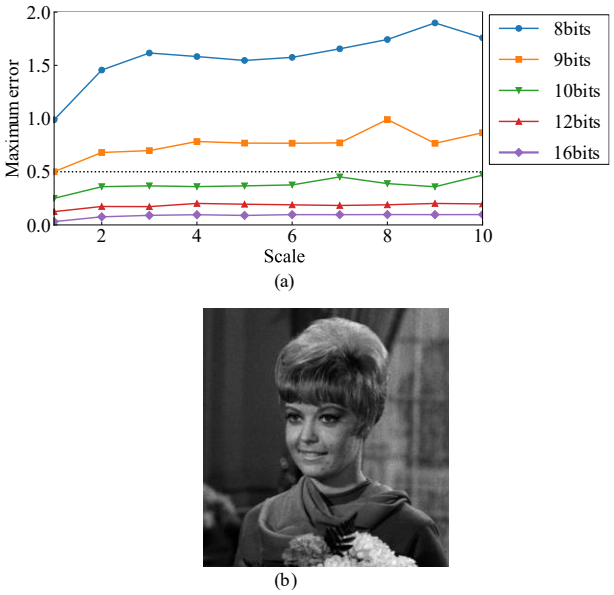

Fig. 3. (a) Relationship between the filter scale and the maximum error in the filtered image. (b) Input image[10] used in this experiment.

shows the relationship for a single image shown in Fig. 3(b), the results for other input images were essentially the same. When eight and nine bits were used, the error exceeded 0.5; this magnitude of the error can affect the least significant bit (LSB). On the other hand, the error never exceeded 0.5 when ten or more bits were used.

Taking these results into account, we adopted ten-bit computing for hardware implementation.

4. Circuit and system implementation

4.1. *System structure*

We implemented the multi-scale Gaussian filtering algorithm described in section 2.2 into an FPGA (Xilinx Artix-7 XC7A100T), and developed an image sensor system composed of a CMOS image sensor (Omni Vision OV5642), the FPGA, and a USB interface. Fig. 4 and 5 show the structure and the appearance of the image sensor system, respectively.

The CMOS image sensor in the system acquires image with 160 ×120 pixels at a maximum rate of 156 frames / second. The multi-scale Gaussian filtering circuit in the FPGA provides up to eight Gaussian filtered images with different standard deviations; the filtered images are sent to PC via the USB interface.

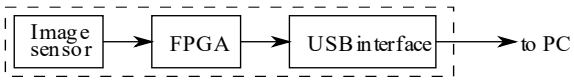

Fig. 4. System structure of the image sensor system with the multi-scale Gaussian filtering circuit

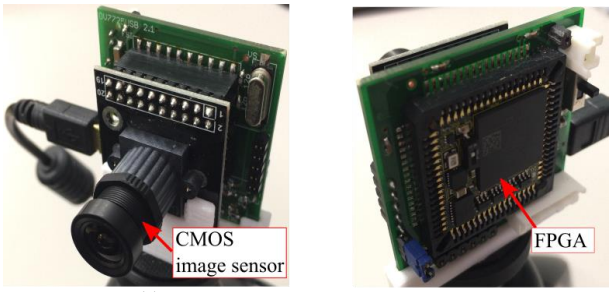

Fig. 5. Appearance of the image sensor system. (a) Front side. (b) Back side.

4.2. *Circuit implementation*

Fig. 6(a) shows the structure of the multi-scale Gaussian filtering circuit implemented in the FPGA.

First, the address counter circuit generates a read-out address signal, and an image is read out from the raw image buffer (not shown in the figure). Here, the image is read out along the X axis and is sent to the component labeled “cascade of 1D filters” through the selector, which is the trapezoid labeled “S”. The 1D filter circuit applies the filter along the X axis, and the filtered image is recorded in the temporal RAM.

Next, the address counter circuit generates a read-out address signal again, and the image is read out from the temporal RAM. Here, the image is read out along the Y axis and is sent to the cascade of 1D filters through the selector. The 1D filter circuit applies the filter along the Y axis, and the filtered image is recorded in the temporal RAM. The repetition of filtering along the X and Y axes provides Gaussian filtered images with a large scale.

The scale counter outputs n, which is the scale of the current filter applied to the image. The RAM selector compares the current scale n with the target scales, and records the filtered image if the current scale equals one of the target scales. Components in the later stage (not shown) can read the filtered images by sending signals to the address ports (ADDR1 and 2 in the figure).

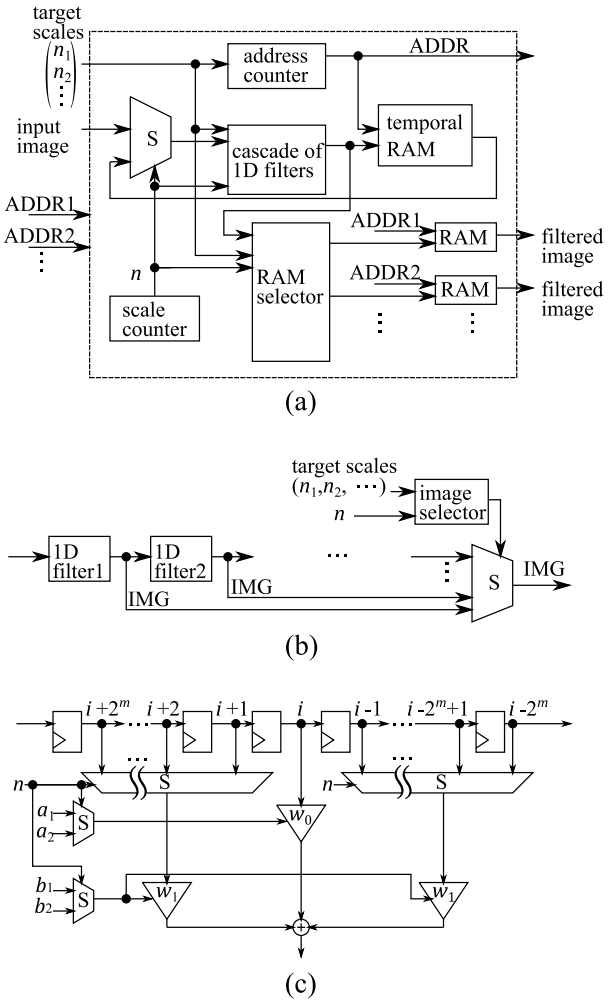

Fig. 6. (a) Structure of the entire multi-scale Gaussian filtering circuit. (b) Structure of the component labeled “cascaded 1D filters”. (c) Structure of the 1D filter circuit. The weights and the width of the filter are determined by the current scale $\boldsymbol{n}$.

Fig. 6(b) shows the structure of the component labeled “cascade of 1D filters”. This circuit is composed of a cascade of 1D filter circuits, each of which performs a single scale of Gaussian filtering. The selector compares the current scale n with the target scales, and outputs the filtered image with the target scale. The number of the cascaded filter implemented in this study was two.

Fig. 7 shows the time chart of a cascade of two 1D filter circuits. This circuit performs two scales of

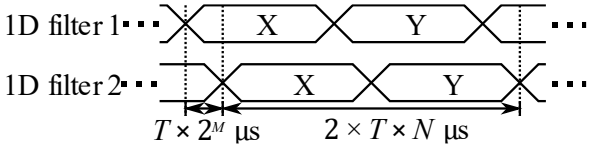

Fig. 7. Time chart of a cascade of 1D filters. ***T*** and ***N*** represent the clock cycle and the total number of pixels, respectively. ***M*** represents the maximum number of ***m*** in equation (2).

Gaussian filtering at $T \times 2^M + 2 \times T \times N$ μs, where T represents the clock cycle, M represents the maximum number of m in equation (2), and N represents the total number of pixels. In the present study, $= 0.02$ μs, $M = 4$, and $N = 19200$ pixels, and the total time required is about 768 μs. One additional cascaded 1D filter performs one more scale of filtering in 0.02×2^M μs.

Fig. 6(c) shows the structure of the 1D filtering circuit. This circuit in Fig. 6(c) consists of the following five components: a shift register that stores data of $2^{M+1} + 1$ pixels, two selectors that select the pixel to be multiplied by the weights at the current scale n, two selectors that select the value of the weights for w_0 and w_1, three multipliers, and an adder that sums the three multiplied values. The values described at the nodes of the shift register $(i + 2^m, \ldots, i - 2^m)$ denote the indices of the stored pixel data, and i represents the center of the stored data.

The circuit in Fig. 6(c) operates as follows. First, image data read from the RAM are stored in the shift register. Next, three multipliers multiply values of the pixels with the following indices by weights chosen by the selectors; the indices are i, $i - 2^m$, and $i + 2^m$. Finally, the adder at the bottom sums three multiplied values and rounds the result to a ten-bit value.

4.3. *Evaluation*

We evaluated the multi-scale Gaussian filtering circuit alone and the entire image sensor system with the circuit using the methods described below.

We evaluated the multi-scale Gaussian circuit by comparing the output of the circuit in response to a step input shown in Fig. 8(a) with the result computed with double-precision floating point. In this evaluation, we implemented a circuit that generates a step image shown in Fig. 8(a) in the FPGA. Fig. 8(b)(c) show the differences between the circuit output and the floating-point operation for Gaussian scales 5 and 6, respectively.

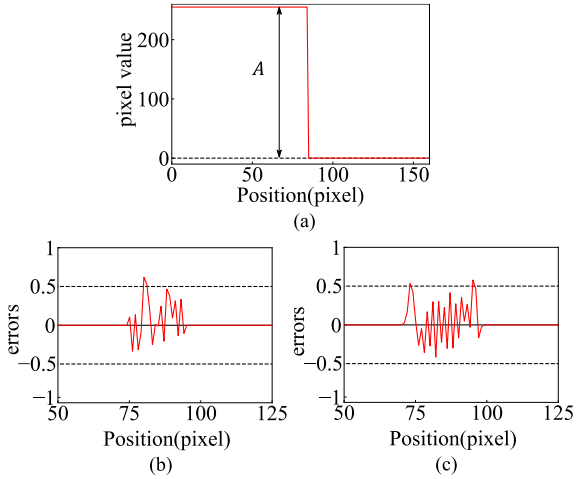

Fig. 8. Difference between the output of the circuit in response to a step input and the result computed with double-precision floating point. (a) Input pattern. The value of ***A*** is 255. (b)(c) The difference for Gaussian scales 5 and 6.

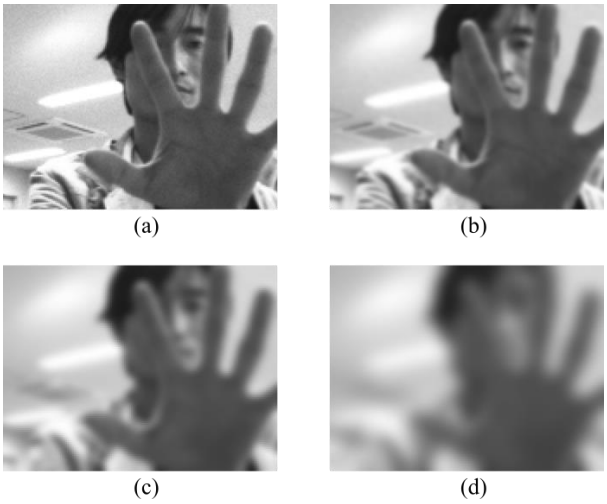

Fig. 9. Output images of the sensor system. (a) Raw image. (b)(c)(d) Gaussian-filtered images whose scales are 2, 4, and 6.

Here, the bit width of the image sent from the circuit is eight, and the input step amplitude is the maximum in eight bits, i.e. A = 255, and therefore, the error was larger than that shown in Fig. 3(a). However, the error was still smaller than the LSB.

We evaluated the entire image sensor system by presenting objects (a human in this experiment) to the image sensor. Fig. 9(a) shows the raw image, and Fig. 9(b), (c), and (d), show the Gaussian-filtered images

whose scales are 2, 4, and 6. Gaussian-filtered images with a larger scale are smoothed more widely. These output results were obtained at 156 frames / second.

5. Conclusion

In the present study, we developed a multi-scale Gaussian filtering circuit and developed an image sensor system with the filtering circuit. The filtering circuit alone and the entire system were evaluated by providing a step input and by presenting real-world scenes. The results showed that the filtering circuit provided filtered images whose error is smaller than the LSB in response to a step input and that the image sensor system provided eight-scale Gaussian-filtered images at 156 frames / second.

Acknowledgements

This work was supported by JSPS KAKENHI Grant Number 19K12916.

References

1. L. Itti, C. Koch, E. Niebur, "A Model of Saliency-Based Visual Attention for Rapid Scene Analysis," IEEE Trans. Pattern Analysis and Machine Intelligence ,Vol.20, pp.1254-1259, 1998.
2. G. Lowe, "Distinctive Image Features from Scale-Invariant Keypoints," International Journal of Computer Vision, Vol. 60, pp. 91-110, 2004.
3. R. W. Rodieck, "Quantitative analysis of cat retinal ganglion cell response to visual stimuli," Vision Research, Vol. 5, pp. 583-601, 1965.
4. S. Marcelja, "Mathematical description of the responses of simple cortical cells," Journal of the Optical Society of America, Vol. 70, No. 11, pp. 1927-1300, 1980.
5. S. W. Kuffler, "Discharge Patterns and Functional Organization of Mammalian Retina," Journal of Neurophysiology, Vol. 16, pp. 37-68, 1953.
6. D. H. Hubel, T. N. Wiesel, "Receptive fields of single neurones in the cat's striate cortex," The Journal of Physiology, Vol. 148, pp. 574-591, 1959.
7. D. Llamocca, C. Carranza, M. Pattichis, "Separable FIR Filtering in FPGA and GPU Implementations: Energy, Performance, and Accuracy Considerations," Proceedings of the 21st International Conference on Field Programmable Logic and Applications, pp. 363-368, 2011.
8. A. K. Joginipelly, D. Charalampidis, "Efficient separable convolution using field programmable gate arrays," Microprocessors and Microsystems, Vol. 71, 102852, 2019.
9. P. J. Burt, "Fast filter transform for image processing," Computer Graphics and Image Processing, Vol. 16, No. 1, pp. 20-51, 1981.
10. M. Sakauchi, Y. Ohsawa, M. Sone, M. Onoe, "Management of the standard image database for image processing researchs (SIDBA)", ITEJ Technical Report, Vol. 8, No. 38, pp. 7-12, 1984. (in Japanese)

Authors Introduction

Mr. Yuki Yamaji

He received his B.S. degree from the Department of Information Science and Technology, Osaka Institute of Technology, Japan in 2021. He is currently a Master's course student in Osaka Institute of Technology, Japan.

Mr. Akito Morita

He received his B.S. degree from the Department of Information Science and Technology, Osaka Institute of Technology, Japan in 2020. He is currently a Master's course student in Osaka Institute of Technology, Japan.

Dr. Hirotsugu Okuno

He received the Ph.D degree in electrical, electronic and information engineering from Osaka University, in 2008. He is currently an Associate Professor at the Faculty of Information Science and Technology, Osaka Institute of Technology. His research interests include visual information processing in the nervous system and their applications to robotics.