PAPER *Special Section on Advanced Technologies in Digital LSIs and Memories*

A VGA 30-fps Realtime Optical-Flow Processor Core for Moving Picture Recognition

Yuichiro MURACHI[†a)], **Yuki FUKUYAMA**[†], **Ryo YAMAMOTO**[†], **Junichi MIYAKOSHI**[†], ***Members***, **Hiroshi KAWAGUCHI**[†], **Hajime ISHIHARA**[††], ***Nonmembers***, **Masayuki MIYAMA**[††], ***Member***, **Yoshio MATSUDA**[††], ***Nonmember***, ***and*** **Masahiko YOSHIMOTO**[†], ***Member***

SUMMARY This paper describes an optical-flow processor core for real-time video recognition. The processor is based on the Pyramidal Lucas and Kanade (PLK) algorithm. It features a smaller chip area, higher pixel rate, and higher accuracy than conventional optical-flow processors. Introduction of search range limitation and the Carman filter to the original PLK algorithm improve the optical-flow accuracy, and reduce the processor hardware cost. Furthermore, window interleaving and window overlap methods reduces the necessary clock frequency of the processor by 70%, allowing low-power characteristics. We first verified the PLK algorithm and architecture with a proto-typed FPGA implementation. Then, we designed a VLSI processor that can handle a VGA 30-fps image sequence at a clock frequency of 332 MHz. The core size and power consumption are estimated at $3.50 \times 3.00\,\mathrm{mm}^2$ and 600 mW, respectively, in a 90-nm process technology.

key words: *optical flow, real-time video recognition, VLSI processor*

1. Introduction

An optical flow is a motion vector of a pixel between two successive images; that flow is the basis of video recognition. Figure 1 depicts an image sequence, "*Table Tennis*" and its optical flows detected from the sequence. Using the optical flows, moving objects in an image sequence or movement of a camera itself can be extracted. An optical flow is useful for vehicle safety systems, robot systems, medical systems, and surveillance systems.

In optical-flow calculations, several equations must be solved at every pixel. The computational cost reaches a few tens of GOPS, even in a CIF 30-fps image sequence (352×288 pixels per frame and 30 frames per second). Consequently, software approaches using generally available processors have examined only a small part of an image; alternatively, such approaches have neglected accuracy. In higher-resolution real-time operation, dedicated hardware is required. Moreover, scalability in the pixel rate and accuracy is preferable for an optical flow processor because the required pixel rate and accuracy differ among application areas.

Several optical-flow processors have been developed [1]–[3]. Figure 2 depicts a comparison of our proposed processor to the conventional ones in terms of accuracy (= MAE: mean angle error) and pixel rate. The MAE is expressed in the following equation:

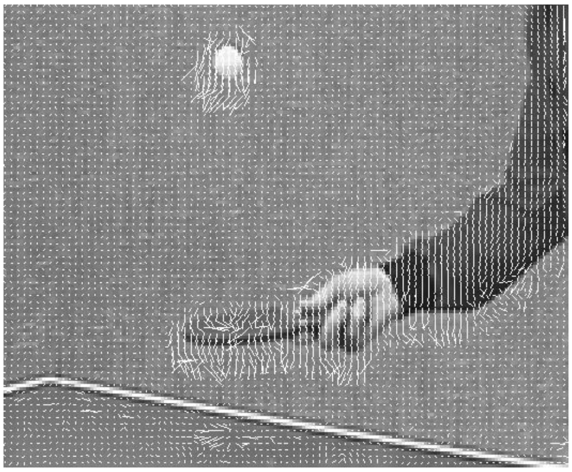


Fig. 1 Optical flows in an image sequence, "*Table tennis.*"

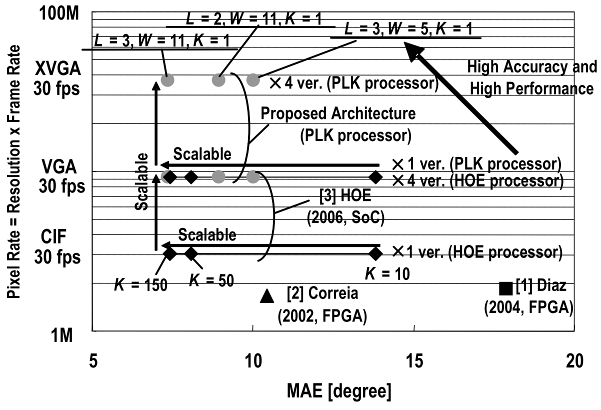


Fig. 2 Performance comparison.

$$\mathrm{MAE} = \frac{1}{N} \sum_{x} \sum_{y} \left(\arccos \frac{\boldsymbol{v}_c \cdot \boldsymbol{v}_e}{\|\boldsymbol{v}_c\| \cdot \|\boldsymbol{v}_e\|} \right), \tag{1}$$

where N is the total number of pixels in a sequence. $\boldsymbol{v}_c$ is a calculated optical flow and $\boldsymbol{v}_e$ is a correct optical flow at each pixel. The MAE is adopted in the references as the error index.

The conventional processors have several problems to handle a VGA 30-fps image. For instance, [1] is comprised of an FPGA, and can not process a CIF 30-fps im-

Manuscript received August 18, 2007.
Manuscript revised November 14, 2007.

[†]The authors are with Kobe University, Kobe-shi, 657-0013 Japan.
[††]The authors are with Kanazawa University, Kanazawa-shi, 920-1164 Japan.
a) E-mail: murachi@cs28.cs.kobe-u.ac.jp

DOI: 10.1093/ietele/e91–c.4.457

age because of insufficient memory resource. The MAE is 18.30 degree, and it is still large for many applications. [2] achieves higher accuracy, but it can not process a CIF 30-fps because its max frequency of the FPGA is 20 MHz. [3] is a full-custom SoC and deals with a CIF 30-fps image sequence. However, the algorithm is based on the hierarchical optical-flow estimation (HOE) algorithm, which requires about 1.2-M Bytes inside the chip. This is because the HOE algorithm assumes that optical flows are smoothly varied over a frame, and calculation has to be carried out on a frame-by-frame basis. The calculated luminance gradients and optical flows in a frame need to be preserved. The HOE algorithm can refine its accuracy by iterating asymptotic equations, although it results in larger workload and higher memory bandwidth for a higher resolution image sequence.

This paper describes an optical-flow processor core based on the Pyramidal Lucas and Kanade (PLK) algorithm [4]. In Fig. 2, L, W, and K respectively denote a hierarchical level, window size, and iteration count in our proposed processor. The PLK algorithm is released in the OpenCV library by Intel Corp.; it applies a hierarchical scheme to the Lucas and Kanade algorithm [5] to handle large movement of objects. The PLK algorithm assumes that optical flows are equal in a small area. An optical flow is separately calculated flow by flow in this algorithm. Therefore, the PLK processer requires only 319-K Byte memory (or possibly less; we mention this in detail later on) for a VGA image. In addition, the iteration steps in this algorithm rapidly converge since the considering area is small. Thus, the PLK algorithm has lower computational cost, less memory size, and higher accuracy than other optical-flow algorithms [5]–[8].

First, we implement the PLK algorithm to verify its function and performance on an FPGA board, as a proto type. Then, we design the optical-flow processor core, as a VLSI. Our VLSI processor based on the PLK algorithm can handle a VGA 30-fps image sequence with far less memory capacity than the HOE processor [3]. The MAE in our proposed processor is 7.36° for the "*Yosemite*" image sequence, which is equal accuracy to that of the HOE processor. Our PLK processor provides both the highest pixel rate and accuracy, and its architecture has wide scalability in terms of pixel rate and accuracy: it can handle an XVGA 30-fps image sequence by connecting four processors in parallel. In addition, it can save its power consumption in low-accuracy applications by appropriately choosing values of the algorithm parameters.

2. Pyramidal LUCAS and KANADE (PLK) Algorithm

An optical flow $\boldsymbol{u}$ in the PLK algorithm is defined as a vector to minimize the following residual function $E(\boldsymbol{u})$:

$$E(\boldsymbol{u}) = \sum (I(\boldsymbol{r}) - J(\boldsymbol{r} + \boldsymbol{u}))^2, \tag{2}$$

where I and J are luminance values of the first and second images of two successive ones, respectively and the summation is over a region centering at position $\boldsymbol{r}$. The region is referred as a window A on the first image and a window B on the second image. The window B has displacement $\boldsymbol{u}$ (the optical flow) to the window A. In a linear approximation, (2) leads to (3):

$$\boldsymbol{G}\boldsymbol{u} = \boldsymbol{b},$$
$$\boldsymbol{G} = \begin{bmatrix} \sum I_x^2 & \sum I_x I_y \\ \sum I_x I_y & \sum I_y^2 \end{bmatrix}, \boldsymbol{b} = \begin{bmatrix} \sum I_x I_t \\ \sum I_y I_t \end{bmatrix}, \tag{3}$$

where the spatial gradient matrix and a mismatch vector are respectively represented as $\boldsymbol{G}$ and $\boldsymbol{b}$. The luminance gradients of x, y, and t coordinates are respectively denoted as I_x, I_y, and I_t. Using (3), the optical flow is computed iteratively with the Newton-Raphson method.

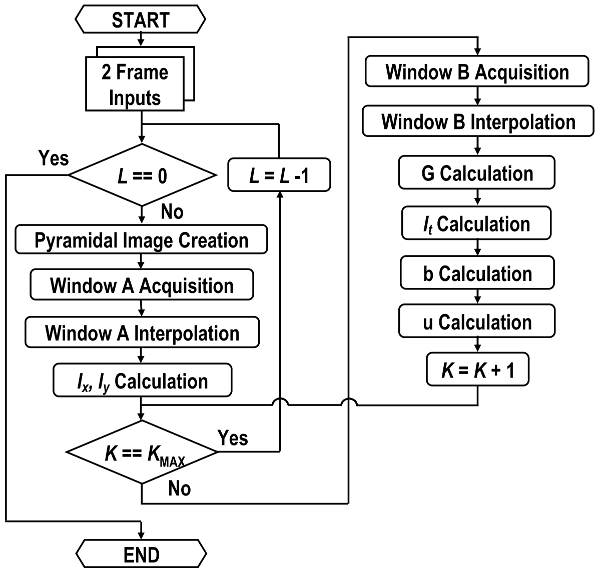


Fig. 3 Flowchart of PLK algorithm.

Figure 3 depicts a flowchart of the PLK algorithm, where L denotes a hierarchical level, and K is the iteration count. First, hierarchical images are generated in a recursive fashion. Then, I_x and I_y are computed from pixel data in the window A and I_t from ones in the windows A and B. Then, $\boldsymbol{G}$ and $\boldsymbol{b}$ are computed to produce an optical flow. These steps are repeated iteratively at a hierarchical level. The position of the window B varies at every iteration step depending on the previous optical flow. Furthermore, this procedure is repeated from the uppermost level to the first level (raw image). Finally, the optical flow is obtained.

3. PLK Algorithm Optimization for VLSI Implementation

In the PLK algorithm, a window B at the L-th level is determined using the (L+1)-th level optical flow. The search range of an optical flow will become large proportionately if the computed optical flow will be large. This increases the size of the memory on a chip. Figure 4 shows the proposed

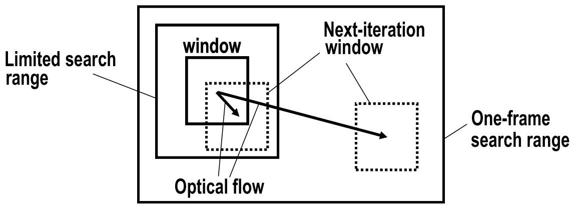


Fig. 4 Search range limitation method.

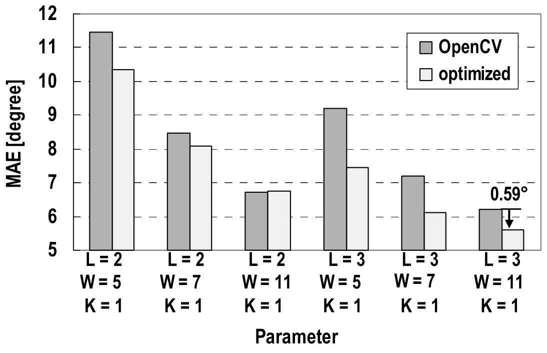


Fig. 5 Accuracy comparison of the PLK algorithm. This MAE is an average value of MAEs of four image sequences: "*Translating Tree*," "*Diverging Tree*," "*Yosemite*" and an original composite sequence.

search range limitation method, which configures the upper limit value of an optical-flow. The search range limitation reduces an internal memory size from a capacity of more than picture frame (319-k Byte) to that of a search range (18-k Byte).

In the search range limitation, an optical flow that has a value larger than the upper limit is rounded to the upper limit value. This is because the PLK algorithm assumes small movement of the flow in a small area, so a large value of the flow is likely to be false caused by a local noise. The method described here reduces false detections and enhances the flow accuracy.

In addition, the Carman filter [9] is adopted to further boost the accuracy. The Carman filter improves the quality of the first image I by using both I and the second image J; it is a simple two-frame weighted average.

$$I_{car}(\boldsymbol{r}) = \frac{3}{4}I(\boldsymbol{r}) + \frac{1}{4}J(\boldsymbol{r}+\boldsymbol{u}), \tag{4}$$

where I_{car} is the Carman-filtered first image. By the way, I_t is expressed as follows:

$$I_t = I(\boldsymbol{r}) - J(\boldsymbol{r}+\boldsymbol{u}), \tag{5}$$

which is rewritten by introducing the Carman filter ($I_{car} \rightarrow I$):

$$I_t = \frac{3}{4}(I(\boldsymbol{r}) - J(\boldsymbol{r}+\boldsymbol{u})). \tag{6}$$

MAE is improved by 0.1° with the Carman filter. Introduction of these methods to the original PLK algorithm both improves accuracy and reduces the memory size.

Figure 5 shows an accuracy comparison of the PLK algorithm according to parameters. The parameter set of L (hierarchical level) = 3, W (window size) = 11, K (iteration count) = 1 is adopted for our VLSI implementation. The algorithm optimization with the above parameter set improves MAE by 0.59° and reduces the memory size by 96% compared to the original PLK algorithm [4]. Here four test image sequences were employed for the accuracy evaluation (Fig. 6).

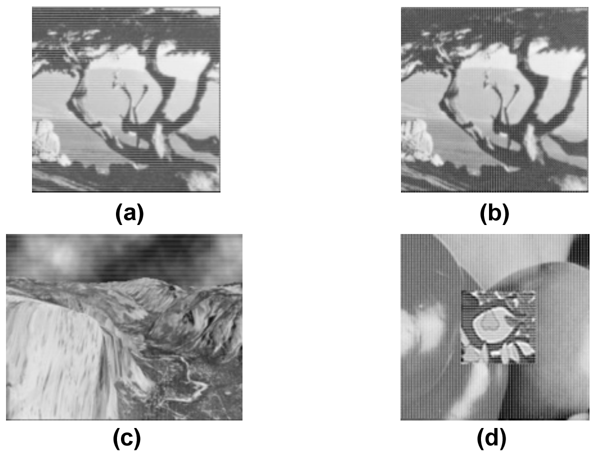


Fig. 6 Four image sequences: (a) "Translating Tree," (b) "Diverging Tree," (c) "Yosemite," and (d) an original composite sequence.

4. VLSI Architecture

4.1 PLK Optical-Flow Processor

Figure 7(a) shows a block diagram of the PLK optical-flow processor, which comprises a pyramidal image creation (PIC), a spatial gradient matrix (SGM), a mismatch vector (MMV), an optical flow (OPF), and so on. Each block is a pipeline stage and operates in parallel.

Because the window B at the L-th level is determined using the $(L + 1)$-th level optical flow, the MMV can not start computing the L-th level optical flow until the $(L + 1)$-th level optical flow is obtained. It causes pipeline stall, as shown in Fig. 8(a). The window interleaving method is proposed as shown in Fig. 8(b). Since an optical-flow calculation corresponding to a pixel is independent of a calculation corresponding to another pixel, the optical-flow calculation of the other pixel can be inserted into idle cycles in Fig. 8(a). Thanks to this method, pipeline stall does not occur and the clock frequency is reduced by 65%.

4.2 Pyramidal Image Creation (PIC)

Figure 7(b) shows a block diagram of the PIC that generates a hierarchical image by sub-sampling and 5 × 5 Gaussian filter. The filtering is made, according to the following equation:

$$I_{pyramidal}(x,y) = \begin{bmatrix}1\\4\\6\\4\\1\end{bmatrix}^{\mathrm{T}} \begin{bmatrix} I(x-2,y-2) & I(x-1,y-2) & I(x,y-2) & I(x+1,y-2) & I(x+2,y-2)\\ I(x-2,y-1) & I(x-1,y-1) & I(x,y-1) & I(x+1,y-1) & I(x+2,y-1)\\ I(x-2,y) & I(x-1,y) & I(x,y) & I(x+1,y) & I(x+2,y)\\ I(x-2,y+1) & I(x-1,y+1) & I(x,y+1) & I(x+1,y+1) & I(x+2,y+1)\\ I(x-2,y+2) & I(x-1,y+2) & I(x,y+2) & I(x+1,y+2) & I(x+2,y+2) \end{bmatrix} \begin{bmatrix}1\\4\\6\\4\\1\end{bmatrix} \tag{7}$$

where $I_{pyramidal}$ is a luminance values on the (L+1)-th level generated with I's on the L-th level. Note that the PIC generates integer pixels.

4.3 Spatial Gradient Matrix (SGM)

Figure 7(c) shows a block diagram of the SGM, which comprises interpolation A (IPAs), spatial gradient (SPGs), gradient matrix elements (GMEs), and a summation (SUM). First, pixel data of a window A are acquired from the Py.Img A memory (a first hierarchical image corresponding to the window A), as shown in Fig. 7(a). Next, the IPAs interpolate luminance values of integer pixels into decimal ones using bi-linear interpolation. The SPGs calculate I_x and I_y. The GMEs calculate elements of a spatial gradient matrix; they are added for each row at the SUM. By repeating these steps for iterations equal to the number of window rows, $\boldsymbol{G}$ is derived; it is the total of spatial gradient matrices.

Most pixels in the window corresponding to a calculating pixel and in the next window corresponding to a next pixel are identical. Most pixel data can be used in common in computing the optical flow at neighboring pixels. We name this reuse of the common pixels "window overlap method," which shortens clock cycles in reading pixels. The Window overlap method reduces the clock frequency by 15% from that using the window interleaving method alone.

4.4 Mismatch Vector (MMV)

Figure 7(d) shows a block diagram of the MMV, which comprises interpolation B (IPBs), mismatch vector (MVs) and a summation (SUM). First, pixel data of window B are acquired from the Py.Img B (a second-frame hierarchical image corresponding to the search range), as shown in Fig. 7(a). Next, IPBs interpolate luminance values at decimal pixels estimated using an upper level optical flow. The MVs calculate I_t from pixel data of both window A and window B with the Carman filter in (3). Mismatch vectors of each pixel are calculated from I_x, I_y, and I_t. Finally, as with the SGM, by adding mismatch vectors, $\boldsymbol{b}$ is derived, which is the total of mismatch vectors.

4.5 Optical Flow (OPF)

Figure 7(e) shows a block diagram of the OPF. This comprises a calculation of the denominator and numerator

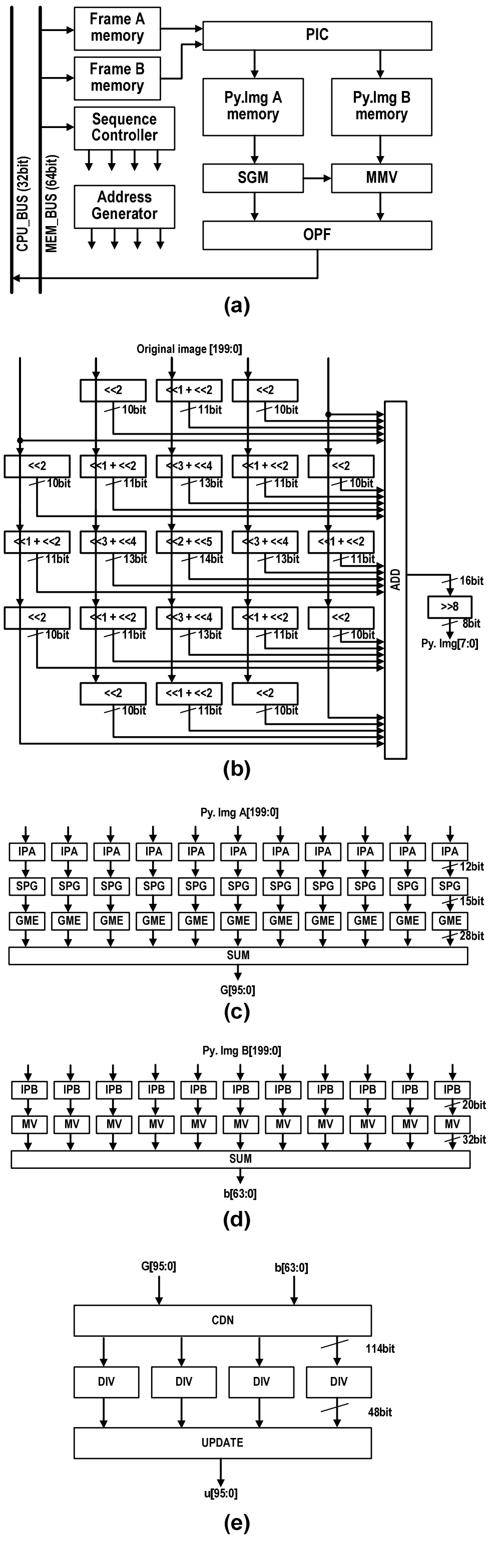


Fig. 7 (a) Block diagram of PLK optical-flow processor. Block diagrams of (b) pyramidal image creation (PIC), (c) spatial gradient matrix (SGM), (d) mismatch vector (MMV), and (e) optical flow (OPF).

(CDN), divider (DIVs), and an update (UPDATE). First, the CDN executes a 32-bit multiplication with four 16-bit multipliers in a four-stage pipelined multiplication. Next, DIVs execute 32-bit division using a subtraction shift recovery algorithm. The DIV can calculate a 1-bit quotient per clock cycle. Because the bit-length of an optical flow is 24, division with the DIV requires 24 clock cycles per optical flow. The DIV must finish calculation at every six clock cycles when $W = 5$ (the smallest window size in the proposed processor) because an optical flow is produced per six clock cycles. Four DIVs are placed in parallel to handle this occasion. Finally, at the UPDATE, an optical flow is updated by adding an upper level optical flow and this optical flow.

4.6 Scalability

We can scale the pixel rate if we increase/decrease the number of processors operating in parallel. One PLK processor just handles a VGA 30-fps images. However, if we utilize four processors in parallel, they can process an XVGA 30-fps image sequence. Figure 9 shows the parallel operation of four PLK processors and optical flow calculation with them. Optical flows of four pixels are processed at a time. Each processor is independent except for input from a memory bus. Each processor receives pixel data of the same region and calculates optical flows at different places. Therefore, optical flows can be computed with reduced clock frequency and without increasing the bus-bandwidth.

In addition, the processor operates at lower clock frequency in less accuracy applications by setting the values of the algorithm parameters to be smaller than those of the optimized ones; as a result, power consumption can be saved.

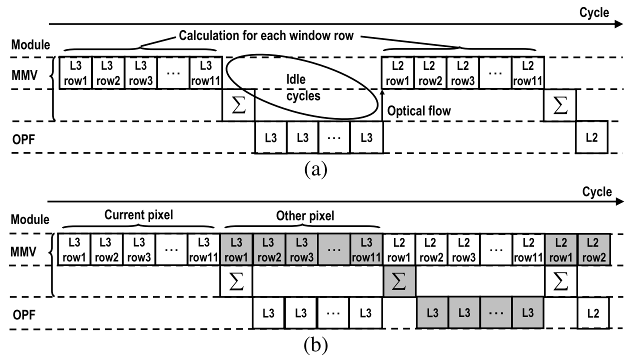


Fig. 8 Timing charts of (a) the conventional method and (b) the window interleaving method.

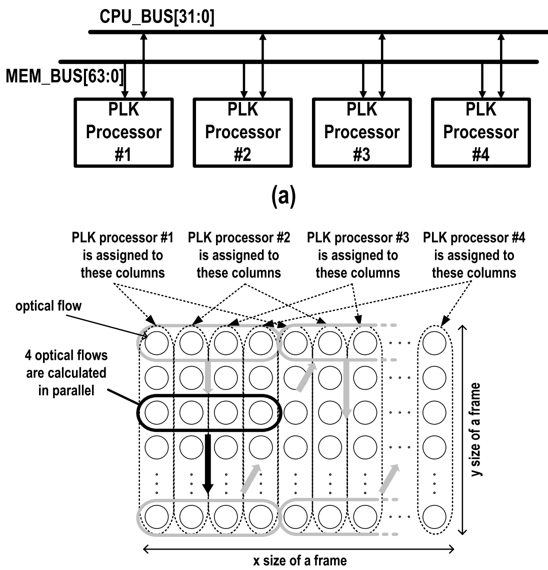

CPU_BUS[31:0]
MEM_BUS[63:0]
PLK Processor #1
PLK Processor #2
PLK Processor #3
PLK Processor #4
(a)
PLK processor #1 is assigned to these columns
PLK processor #2 is assigned to these columns
PLK processor #3 is assigned to these columns
PLK processor #4 is assigned to these columns
optical flow
4 optical flows are calculated in parallel
y size of a frame
x size of a frame
(b)

Fig. 9 (a) Parallel architecture of four PLK processors. (b) Optical flows calculation with four processors.

5. A Proto-Typed FPGA Implementation

To evaluate the PLK algorithm and our proposed architecture, we first implemented them on an FPGA. In this FPGA implementation, we downsized the pixel resolution to QCIF 30 fps, since the target FPGA board is limited to 20 MHz operation.

5.1 Target FPGA Board and Evaluation System

The target FPGA board is Celoxica RC2000. The FPGA board has a Xilinx Virtex-II XC2 V6000. As frame buffers, six external synchronous SRAMs (SSRAMs) are available. A camera (Silicon Video SV642M) and an image capture board (PIXCI SI Digital Frame Grabber) feed images to the FPGA board through a PC. The FPGA board calculates optical flow, and the optical flow is overlaid with its original image on the PC's display. The software framework operates on the PC to handle the image data and the optical flows. The FPGA board and the image capture board are controlled by APIs and drivers through Windows XP on the PC.

5.2 Verification

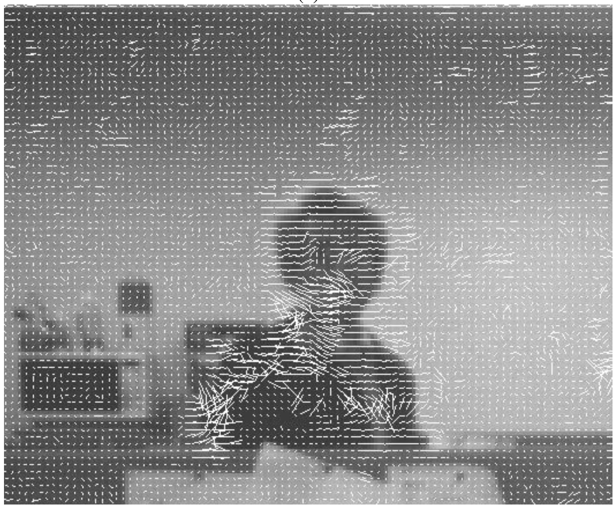
Considering the hardware resource of the FPGA, we set the parameters as $L = 2$, $W = 5$, and $K = 2$ in this implementation. Table 1 compares the average MAEs between the original OpenCV PLK algorithm (ANSI C: floating point) and the results from the FPGA implementation (fixed point). The four image sequences used in Fig. 6 are evaluated. The respective errors between them are about 0.3°, which demonstrates that our architecture is as accurate as the floating-point operation.

Figure 10 is an image taken by the camera, and its optical flows calculated with the FPGA board. A man is walking from the left side to the right side, bringing up his arm. We

Table 1 MAE comparison between floating-point operation and our proposed architecture.

	MAE (degree)			
	Translating Tree	*Diverging Tree*	*Yosemite*	*Original Composite sequence*
ANSI C	**2.537163**	**6.565456**	**13.702810**	**5.358324**
FPGA	**2.706657**	**6.844909**	**14.016331**	**5.600715**

(a)



(b)

Fig. 10 (a) Input image and (b) its optical flows.

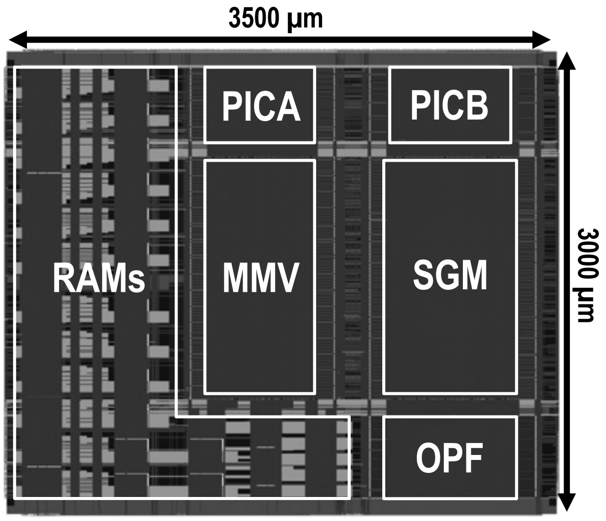


Fig. 11 Layout of PLK processor core.

Table 2 Comparison of performance.

		PLK Processor		HOE Processor
Prameter		L = 3, W = 11, K = 1		K = 150
Resolution		VGA 30 fps	CIF 30 fps	CIF 30 fps
Frequency		332 MHz	110 MHz	189 MHz
Power		600 mW	198 mW	500 mW
Size	Logic	590 kGate		311 kGate
	Memory	18 kByte		1200 kByte
	Area	10.5 mm^2		30 mm^2
MAE	Translating Tree	0.47°		0.65°
	Diverging Tree	2.87°		2.75°
	Yosemite	7.36°		7.44°

verified that the PLK algorithm and our proposed architecture properly work in a real system.

6. VLSI Implementation

Figure 11 shows a PLK processor core layout, which is designed in a 90-nm process technology. Then, the respective area sizes of the SGM, the MMV, and the OPF are $1.50 \times 0.84\,\text{mm}^2$, $1.50 \times 0.70\,\text{mm}^2$, and $0.50 \times 0.84\,\text{mm}^2$. The core size, which includes all blocks, is estimated within $3.50 \times 3.00\,\text{mm}^2$. Table 2 shows a performance comparison of the PLK and the HOE processors. The PLK processor achieves real-time processing of a VGA30-fps image sequence with smaller chip size than that of the HOE processor.

Total power consumption of the SGM, the MMV, and the OPF is estimated at 600 mW when the parameters are L = 3, W = 11, and K = 1. The accuracy of the PLK algorithm is equivalent to that of the HOE algorithm. The trade-off between MAE and power dissipation is indicated in Fig. 12.

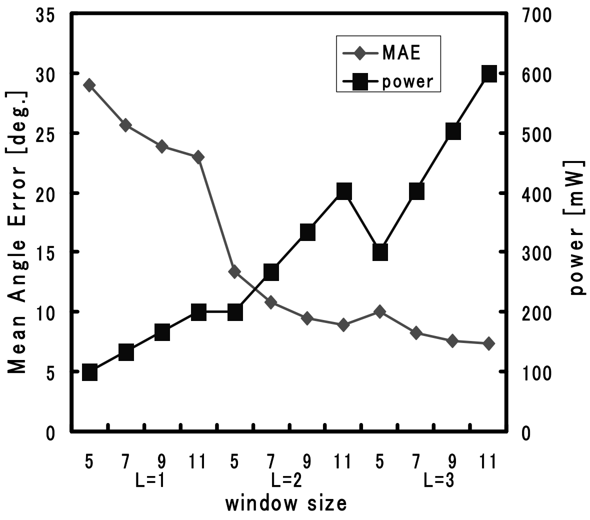


Fig. 12 Trade off between MAE and power.

A parameter set of L and W can be chosen, on condition of the accuracy or power.

7. Conclusion

We described an optical-flow processor core for real-time video recognition based on the PLK algorithm. For VLSI implementation, introducing the search range limitation and the Carman filter as computing the temporal luminance gradient optimizes the PLK algorithm. The optimized PLK algorithm provides accuracy which is equivalent to that of the HOE algorithm, with improved MAE by 0.59°, and memory size reduced by 96% for parameters of $L = 3$, $W = 11$ and $K = 1$ (see Table 2). Moreover, introduction of window overlap and window interleaving methods reduces the PLK processor clock frequency by 70%. The core size is estimated as $3.50 \times 3.00\,\mathrm{mm}^2$ in a 90-nm process technology; it can handle a VGA 30-fps image sequence at 332 MHz clock frequency and 600 mW power consumption. Therefore, the proposed optical-flow processor is applicable to several application fields of real-time video recognition tasks such as those for vehicle safety, robotics, medical care, and surveillance.

Acknowledgement

This work was supported by the Semiconductor Technology Academic Research Center (STARC), and VLSI Design and Education Center (VDEC), the University of Tokyo in collaboration with Cadence Design Systems, Inc., Synopsys, Inc., and Celoxica Ltd.

References

[1] J. Diaz, E. Ros, S. Mota, F. Pelayo, and E.M. Ortigosa, "Real-time optical flow computation using FPGAs," Proc. Early Cognitive Vision Workshop, 2004.

[2] M.V. Correia and A.C. Campilho, "Real-time implementation of an optical flow algorithm," ICPR, vol.4, pp.247–250, 2002.

[3] N. Minegishi, J. Miyakoshi, Y. Kuroda, T. Katagiri, Y. Fukuyama, R. Yamamoto, M. Miyama, K. Imamura, H. Hashimoto, and M. Yoshimoto, "VLSI architecture study of a real-time scalable optical flow processor for video segmentation," IEICE Trans. Electron., vol.E89-C, no.3, pp.230–242, March 2006.

[4] J.Y. Bouguet, "Pyramidal Implementation of the Lucas Kanade feature tracker description of the algorithm," Intel Corporation, Microprocessor Research Labs, OpenCV Documents, 1999.

[5] B.D. Lucas and T. Kanade, "An iterative image registration technique with an application to stereo vision," Proc. DARPA Image Understanding Workshop, pp.121–130, 1981.

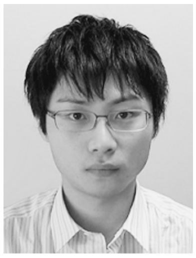
[6] T. Yamamoto, K. Imamura, and H. Hashimoto, "Improvement of optical flow by moving object detection using temporal correlation," Trans. IIITE, vol.55, no.6, pp.907–911, 2001.

[7] B.K.P. Horn and B.G. Schunck, "Determining optical flow," Artif. Intell., vol.17, pp.185–204, 1981.

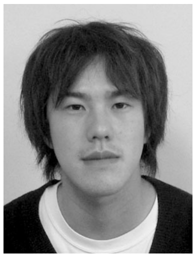
[8] J.L. Barron, D.J. Fleet, and S.S. Beauchemin, "Performance of optical flow techniques," Int. J. Comput. Vis., vol.12, no.1, pp.43–77, 1994.

[9] C. Neustaedter, "An evaluation of optical flow using Lucas and Kanade's algorithm," University of Calgary Department of Computer Science, Calgary, AB, T2N 1N4, Canada, 2002.

Yuichiro Murachi was born on November 1, 1980. He received a B.S. degree from Kanazawa University in 2003. He received an M.E. degree from Kanazawa University, Ishikawa, Japan, in 2005. He is currently enrolled in the Doctoral course in Kobe University. His research interests are VLSI systems and implementation of multimedia communication systems.



Yuki Fukuyama was born on March 15, 1982. He received the M.E. Degree in Computer Science and Systems Engineering from Graduate School of Science and Technology, Kobe University in 2007. He has been working at Central Research Laboratory, Hitachi, Ltd. His research interests include image processing algorithm and image recognition.



Ryo Yamamoto was born on April 26, 1982. He received a B.E. in Electrical and Information Engineering from Kanazawa Univer sity, Ishikawa, Japan, in 2005. He received an M.E. degree from Kobe University, Kobe, Japan, in 2007. He joined Mitsubishi Electric Corp. in 2007. His research interests are VLSI systems and implementation of multimedia communication systems.

Junichi Miyakoshi was born in Japan in 1980. He received the B.E. and M.E. degrees in electronics and computer science from Kanazawa University, Japan, in 2002 and 2004, respectively. He received the Ph.D. degree in infomatics and electoronics from Kobe University, Japan, in 2007. He joined the Central Research Labratory, Hitachi, Ltd., in 2007. His research interests include high-performance and low-power VLSI designs.

Hiroshi Kawaguchi received B.E. and M.E. degrees in Electronic Engineering from Chiba University, Chiba, Japan, respectively, in 1991 and 1993. He received a Ph.D. degree in Engineering from the University of Tokyo, Tokyo, Japan, in 2006. He joined Konami Corp., Kobe, Japan, in 1993, where he developed arcade entertainment systems. He moved to the Institute of Industrial Science, the University of Tokyo, as a Technical Associate in 1996, and was appointed as a Research Associate in 2003. In 2005, he moved to the Department of Computer and Systems Engineering, Kobe University, Kobe, Japan, as a Research Associate. Since 2007, he has been an Associate Professor at the Department of Computer Science and Systems Engineering, at Kobe University. He is also a Collaborative Researcher with the Institute of Industrial Science, the University of Tokyo. His current research interests include low-power VLSI design, hardware design for wireless sensor networks, and recognition processors. Dr. Kawaguchi received the IEEE ISSCC 2004 Takuo Sugano Outstanding Paper Award and the IEEE Kansai Section 2006 Gold Award. He has served as a Program Committee Member for IEEE Symposium on Low-Power and High-Speed Chips (COOL Chips), and as a Guest Associate Editor of IEICE Transactions on Fundamentals of Electronics, Communications and Computer Sciences. He is a member of IEEE and ACM.

Hajime Ishihara was born on November 5, 1983. He received a B.E. in Electrical and Electronic Engineering from Kanazawa University in 2006. He is currently a Master's student at Kanazawa University. His research interest is low-power VLSI techniques for image processing based on optical flow.

Masayuki Miyama was born on March 26, 1966. He received a B.S. degree in Computer Science from the University of Tsukuba in 1988. He joined PFU Ltd. in 1988. He received an M.S. degree in Computer Science from the Japan Advanced Institute of Science and Technology in 1995. He joined Innotech Co. in 1996. He received the Ph.D. degree in electrical engineering and computer science from Kanazawa University in 2004. He is a research assistant at the Department of Electrical and Electronic Engineering at Kanazawa University. His present research focus is low-power design techniques for multimedia VLSI.

Yoshio Matsuda was born in Ehime, Japan, on October 26, 1954. He received the B.S. degree in physics and the M.S. and Ph.D. degree in applied physics from Osaka University in 1977, 1979, and 1983, respectively. He joined the LSI Laboratory, Mitsubishi Electric Corporation, Itami, Japan, in 1985. He was engaged in development of DRAM, advance CMOS logic, and high frequency devices and circuits of compound semiconductors. Since 2005, he has been a professor of Graduate School of Natural Science and Technology at Kanazawa University, Japan. His research is in the fields of integrated circuits design where his interests have includes multimedia system, low power SoC, and image compression processors.

Masahiko Yoshimoto received a B.S. degree in Electronic Engineering from the Nagoya Institute of Technology, Nagoya, Japan, in 1975 and an M.S. degree in Electronic Engineering from Nagoya University, Nagoya, Japan, in 1977. He received a Ph.D. degree in Electrical Engineering from Nagoya University, Nagoya, Japan in 1998. He joined the LSI Laboratory, Mitsubishi Electric Power Products Inc., Itami, Japan, in April 1977. During 1978–1983 he was engaged in the design of NMOS and CMOS static RAM, including a 64 K full CMOS RAM with the world's first divided-word-line structure. From 1984, he was involved in research and development of multimedia ULSI systems for digital broadcasting and digital communication systems based on MPEG2 and MPEG4 Codec LSI core technology. Since 2000, he has been a Professor of the Dept. of Electrical and Electronic Systems Engineering at Kanazawa University, Japan. Since 2004, he has been a Professor of the Dept. of Computer and Systems Engineering at Kobe University, Japan. His current activities specifically emphasize research and development of multimedia and ubiquitous media VLSI systems including an ultra-low-power image compression processor and a low-power wireless interface circuit. He holds 70 registered patents. He served on the Program Committee of the IEEE International Solid State Circuit Conference during 1991–1993. In addition, he has served as a Guest Editor for special issues on Low-Power System LSI, IP, and Related Technologies of IEICE Transactions in 2004. He received R&D100 awards from R&D Magazine in 1990 and 1996, respectively, for development of the DISP and development of a real-time MPEG2 video encoder chipset.