The 2009 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 11-15, 2009 St. Louis, USA

# Development of High-speed and Real-time Vision Platform, H$^3$ Vision

Idaku Ishii, Taku Taniguchi, Ryo Sukenobe, and Kenkichi Yamamoto

*Abstract*— **In this paper, we introduce a high-speed vision platform, H$^3$ (Hiroshima Hyper Human) Vision, which can simultaneously process a 1024× 1024 pixel image at 1000 fps and a 256× 256 pixel image at 10000 fps by implementing image processing algorithms as hardware logic on a dedicated FPGA board. Various types of algorithms are actually implemented to show that H$^3$ Vision can work a high-speed image processing engine. Experimental results are shown for multi-target color tracking, feature point tracking, optical flow detection, and pattern recognition by using high-order local auto-correlation (HLAC) feature at a frame rate of 1000 fps or more.**

## I. Introduction

Many conventional robot visions use video signals (e.g., NTSC 30 fps) designed according to human eye characteristics and their processing speeds are limited to approximately the same level as the human eye. In robots and various other application fields, such as factory automation (FA), multimedia, and biomedical fields, there is a growing demand for high-speed image processing technology capable of recognizing high-speed phenomena in real time.

For high-speed visions at 1000 fps, vision chips consisting of sensors and parallel processing circuits integrated on single chips have been developed [1]-[4]. Because of the currently limited integration technologies, however, these vision chips have pixels not more than thousands or tens of thousands and are difficult to apply to image measurement with high spatial resolution. Meanwhile, many imagers can operate at high frame rates because of the region of interest (ROI) function that selects local image areas. For this ROI function, Ishii et al. adopted the idea of intellectual pixel selection, where only the pixels necessary for each frame are selected according to the result of processing the preceding frame. Thus, personal computer (PC) based multi target tracking that achieves both mega-pixel spatial resolution and 1000 fps-level high speed was realized [5]. Currently, the number of pixels that can be transmitted at 1000 fps from an image sensor and processed in a PC is not more than 10000. This limits the simultaneous capturing of measurement targets distributed in an image.

Recent attempts have been made to implement various kinds of high-speed image processing with circuits on a field-programmable gate array (FPGA) and other hardware after connecting a high-speed camera head and a dedicated processing board by high-speed serial communication. For example, parallel coprocessors were mounted for multi-point target tracking of 256×256 pixel images [6], and a high-speed Hough transform was implemented on a FPGA [7], and Hiroshima Hyper Human Vision (H$^3$ Vision) was realized for 1000 fps real-time image processing on 1024×1024 pixel images [8]. These implementations have made it possible to verify the performance of various algorithms on the same platform at high speed.

Therefore, this paper introduces a configuration of H$^3$ Vision that the authors have developed as a high-speed vision platform and verifies its effectiveness with examples of various hardware-implemented image processing algorithms.

I. Ishii, T. Taniguchi, R. Sukenobe, and K. Yamamoto are with Hiroshima University, Hiroshima 739-8527, Japan (corresponding author (Idaku Ishii) to provide phone: +81-82-424-7692; fax: +81-82-422-7158; e-mail: iishii@robotics.hiroshima-u.ac.jp).

## II. The Necessity of High-speed Vision

High-speed vision is a useful real-time sensor for visual feedback control at hundreds of Hz or more, which derives its maximum mechanical performance from a robot. The effectiveness of high-speed visual feedback control has been originally reported in 1ms visual feedback system [9], which realized 2 DOF high-speed target tracking. Several researches with high-speed visual feedback control have been also reported; high-speed grasping by a robot hand [10], high-speed batting manipulation [11], virtual stillness for beating heart surgery [12], and microscopic microbe tracking [13].

High-speed vision is also capable of capturing high-speed phenomena as distributed dynamics information, even though they are difficult to recognize by the human eye. Therefore, high-speed vision can work as a dynamic sensing tool significant for many applications such as human interface, biological behavior analysis, fluidic analysis, and structural vibration analysis. Several researches for dynamic sensing have been already reported; fingertip contact detection for computer interface [14], quantification of mice's scratching behavior for atopic dermatitis drug developments [15], and eye stiffness sensing for glaucoma diagnosis [16].

In order to have wider range of application for dynamic sensing as well as robot control, it is important to develop more intelligent high-speed vision systems for complex real world sensing, which are not limited with conventional binary image processing. As a result, we developed a high-speed vision platform that can implement more complicated image processing algorithms as hardware logics for high frame rate color and gray-level images with high spatial resolution.

## III. High-speed Vision Platform H$^3$ Vision

### A. Outline of System

H$^3$ Vision is a high-speed and real-time vision platform for image processing with high spatial resolution at a high frame rate. H$^3$ Vision was designed to implement and verify various image processing algorithms on the same platform. Fig. 1 shows the general configuration.

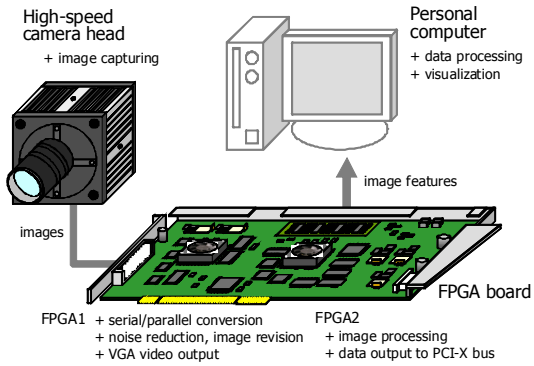



Fig. 1 Configuration of H³ Vision

This platform consists of a high-speed camera head, a dedicated FPGA image processing board, and a PC. Images captured by the high-speed camera head are transferred to the dedicated FPGA image processing board at high speed. The dedicated FPGA image processing board executes arbitrary image processing implemented by the user as hardware to greatly reduce the volume of transfer data to the PC. The final processing results are transferred at high speed to the PC through a PCI-X bus to realize processing with high spatial resolution at high frame rate.

### B. Components

For the high-speed camera head, the camera head for FASTCAM-1024PCI (Photron), a commercially available high-speed camera, was adopted [17]. Fig. 2 (left) shows the camera head and Table 1 gives the specifications. As we can see from the table, color 10-bit images can be transferred to a post-stage dedicated FPGA image processing board at 1000 fps for 1024×1024 pixels and 10000 fps for 256×256 pixels.

The FPGA image processing board dedicated to H³ Vision was designed for high-speed processing of 1024×1024 pixel images transferred as fast as 1000 fps and 256×256 pixel images transferred as fast as 10000 fps. Fig. 2 (right) shows the appearance of the FPGA board, Table 2 gives the specifications, and Fig. 3 is a function block diagram.

This board consists of the FPGA (Xilinx XC2VP100 – hereafter called FPGA1) for image correction and display


Fig. 2 H³ Vision

TABLE 1 SPECIFICATION OF H³ VISION CAMERA HEAD

| | |
|---|---|
| resolution | max. 1024 × 1024 pixel |
| bit depth | color 10 bit |
| frame rate | 1000 fps (1024 × 1024 pixel)<br>10000 fps (256 × 256 pixel) |
| imager size | 17.4 mm × 17.4 mm |
| pixel size | 17 µm × 17 µm |
| camera mount | C mount |
| interface | digital serial 12ch (LVDS) |
| size (W×D×H) | 120 mm × 120 mm × 120 mm |

TABLE 2 SPECIFICATION OF H³ VISION FPGA BOARD

| | |
|---|---|
| board size | 312 mm × 128 mm |
| camera I/F | digital serial 12ch (LVDS) |
| FPGA | Xilinx XC2VP100 × 2<br>approx. 11million gates, 1164 I/O pins |
| available memories | DDR-SDRAM : 640 MByte, DDR266<br>SDRAM : 512 MByte<br>SSRAM : 8 MByte<br>FPGA(internal) : 999 kByte |
| Bus I/F | PCI-X (64 bit, 66 MHz) |

processed images, the FPGA (Xilinx XC2VP100 – hereafter called FPGA2) for hardware implementation of the algorithms by the user, DDR-SDRAM and various other memories, and such interface circuits as a PCI-X bus and bridge. The following describes the path for image data to be processed in the FPGA image processing board.

(1) Converting image data from serial to parallel

Serial image data transferred from the high-speed camera head is converted into 160-bit parallel data (10 bits/pixel) with 16 pixels as a block.

(2) Correcting images

For the parallel data of (1), fixed pattern noise removal and shading correction are conducted by using corrected images recorded in flash memory.

(3) Writing corrected images into DDR-SDRAM

The corrected image data is transferred once from FPGA1 to FPGA2 through a 160-bit bus and stored in DDR-SDRAM, which functions as a buffer.

(4) Executing user-specified image processing

Stored images are read from DDR-SDRAM and processed in a processing block, where arbitrary user-implemented image processing in FPGA2 is executed.

(5) Transferring processing results to PC

By high-speed transfer using FIFO, the processing result of (4) is sent to the PCI-X bus through the PCI-I/F and PCI bridge for data acquisition from the PC.

For H³ Vision, any PC with PCI-X bus can be used if its OS is Windows XP. The CPU and memory specifications can be selected according to the purpose. Various API functions

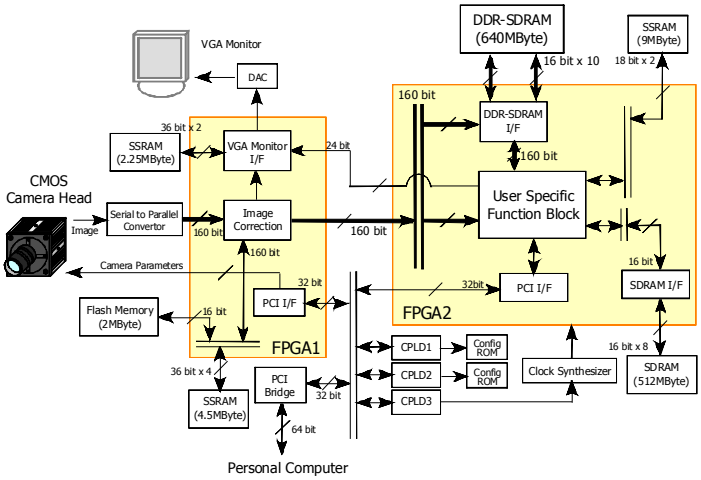



Fig. 3 Block diagram of H³ Vision FPGA board

associated with board control and data access are prepared as middleware for the development of various application programs using the C language and other general-purpose programming languages. In this study, we used a PC of the specifications given in Table 3.

This paper introduces examples of implementing various image processing algorithms on the high-speed vision platform $H^3$ Vision for high-speed image processing.

TABLE 3 SPECIFICATION OF PERSONAL COMPUTER

| | |
|---|---|
| main board | Intel Server Board SE7525GP2 |
| CPU | Xeon Dual Core 2.8GHz × 1 |
| memory | 2GB (DDR333 1GB × 2) |
| OS | Windows XP Professional SP2 |
| bus I/F | PCI-X bus |

## IV. MULTI-COLOR MARKER TRACKING

### A. Outline of Algorithm

Multi-color marker tracking in an image is realized by a) labeling target areas with color information and b) setting inter-frame correspondence to the tracking targets. By this tracking, time-series data containing the position of a tracked object can be acquired. When 10-bit images of 1024×1024 pixels are processed at 1000 fps, a bandwidth of at least 10 Gbps is necessary for processing requiring full-pixel access. This is difficult to realize by software that needs to transfer images to a PC.

In this study, the image size is assumed to be $N^2$ and the number of tracking objects is assumed to be $M$. Labeling processing is integrated as hardware logic for conversion from dimension $N^2$ to dimension $M$, and correspondence processing is PC software-implemented on $H^3$ Vision to set the correspondence from dimension $M$ to dimension $M$.

Fig. 4 shows the algorithm flow described as follows.

(1) Color labeling

a) Color extraction (Processing order: $O(N^2)$)

Based on the color hue information, only specified color areas are extracted to separate color marker areas from the background area.

b) Block labeling (Processing order: $O(N^2)$)

For high-speed processing, an area of 1024×1024 pixels is divided into 16 blocks and each block is labeled in parallel. The labeling algorithm completes labeling by a single scan that requires no image size storage areas and retains only the zeroth moment $m_0$ and the first moment $m_x, m_y$ of each labeled area as processing results.

From the zeroth and first moments, the center of gravity corresponding to a labeled area can be calculated as $(C_x, C_y) = (m_x/m_0, m_y/m_0)$ to extract each label position.

c) Merging (Processing order: $O(M)$)

Since labeling is executed on each block in parallel, a target across a block boundary is divided and different labels are affixed. To solve this problem, the distance between labels near a boundary is calculated. If the distance is within a given threshold, the divided targets are merged as the same target.

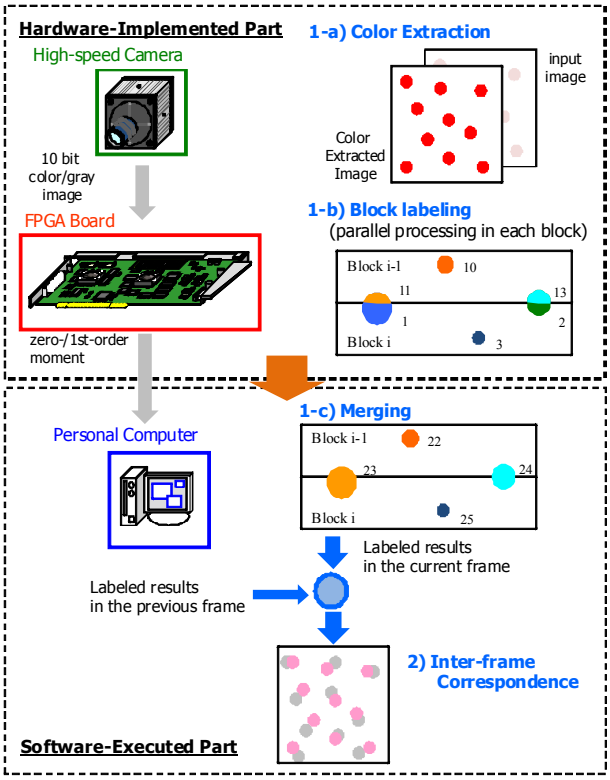



Fig. 4 Algorithm flow for multi-color marker tracking

(2) Inter-frame correspondence setting (Processing order: $O(M)$)

A label map for exploration is generated with a label number embedded at the center of gravity of every label in the one frame before. Then, for each label in the current frame, a square window around the center of gravity is scanned for the corresponding label number to set the label correspondence.

Processing of the processing order $O(N^2)$ is implemented by hardware and processing of the processing order $O(M)$ is software-executed.

### B. Implementation and Operation

Implementing the above algorithm on $H^3$ Vision realized marker extraction up to 2300 markers (1024×1024 pixels) at 1000 fps and 254 markers (256×256 pixels) at 10000 fps. The resource consumption of FPGA2 on the FPGA image processing board was Slice 66% (29233/44096), Slice Register 41% (36355/88912), LUT 45% (40077/88192), Block RAM 66% (297/444), and GCLK 50% (8/16).

A disc with 60 red markers was rotated at 26 rotations per second (rps) and all markers were measured at 1000 fps. Fig. 5 shows the $x$-$y$ trajectories, velocity distribution, and acceleration distribution of every marker. From the velocity distribution in the tangential direction proportional to the radius and from the acceleration distribution in the direction toward center proportional to the radius, we see that the high-speed circular motions of markers rotating are tracked correctly.

By marker tracking at 10000 fps on $H^3$ Vision, vibrating guitar strings were simultaneously measured from the 1st string to the 6th string. Fig. 6 shows the time-series changes of the y-coordinate in 0.1 second. The vibration frequencies

of the strings become 330, 246, 196, 147, 110, and 82 Hz, respectively. Here we set markers on the guitar strings to be tracked. They match the tuning frequencies when the strings are released. Thus, we see that H$^3$ Vision can measure the distribution of high velocity rotations and vibrations that cannot be caught by the human eye.

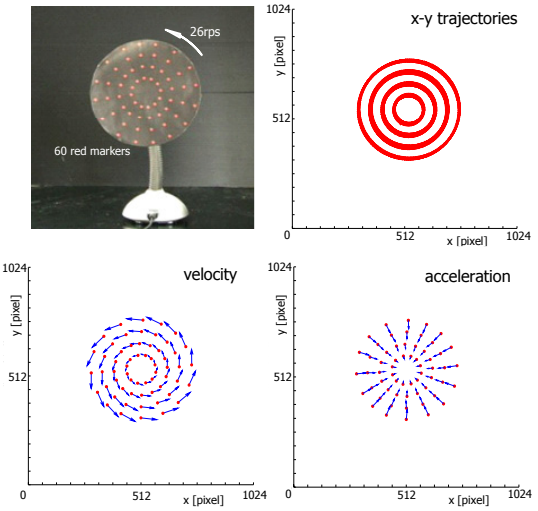



Fig. 5 Rotating disc with 60 red markers

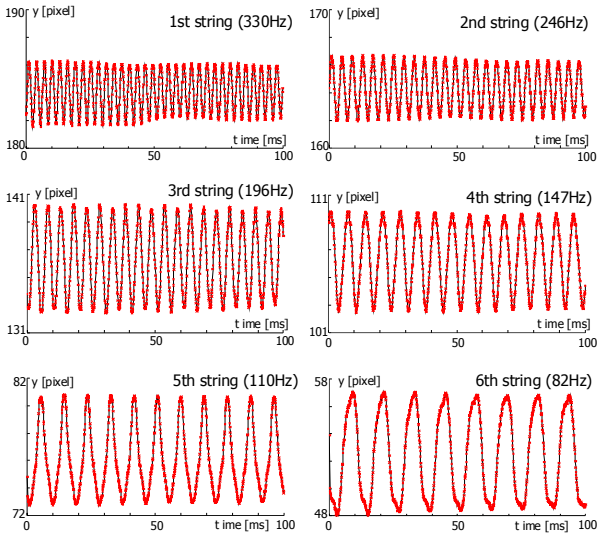



Fig. 6 Vibrating guitar strings

## V. Feature Point Tracking

### A. Outline of Algorithm

Feature point tracking by the Kanade-Lucas-Tomasi (KLT) Tracker [18] automatically extracts feature points suitable for tracking in a frame. This is realized by a) image feature calculation based on the intensity gradient, b) feature point selection, and c) feature point correspondence between the preceding and succeeding frames.

In this paper, when the number of feature points is $M$, image feature calculation for conversion from dimension $N^2$ to dimension $N^2$ and feature point selection for conversion from dimension $N^2$ to dimension $M$ are implemented by hardware for feature point tracking. The correspondence processing is PC software-implemented on H$^3$ Vision to set the correspondence on dimension $M$.

The flow of the implemented algorithm is as follows.

(1) Image feature calculation (Processing order: $O(N^2)$)

By noting that the brightness gradient is in all directions at corners as well as other feature points using KLT Tracker, this processing calculates the following image feature for every element from the brightness gradient $(x, y)$ :

$$\lambda'(x,y) = \frac{\det C}{\mathrm{Tr}\, C}, \quad C(x,y) = \sum_{(x,y)\in N(x,y)} \begin{pmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{pmatrix}, \quad (1)$$

where det() is the determinant, Tr() is matrix trace, $N(x, y)$ is the 3×3 pixel adjacent area of pixel $(x, y)$, and $I_x, I_y$ are $x$ and $y$ differentials of the input image .

(2) Feature point selection (Processing order: $O(N^2)$)

KLT Tracker used feature points selection by sorting. Sorting, however, is not necessarily suitable for hardware integration because its memory consumption grows during processing. In this paper, based on the assumption that there are no other feature points near a feature point, the 1024× 1024 pixel image is divided into 16384 blocks of 8×8 pixels, and the maximum values of $\lambda'(x, y)$ is chosen within each block to avoid sorting the entire image. This suppresses memory consumption in feature point selection and enables parallel processing to increase processing speed.

A feature point satisfying the following conditions of i) image feature greater than the threshold and ii) a certain distance away from adjacent feature points, is ultimately selected from candidate feature points for each block. Fig. 7 shows the feature point selection flow.

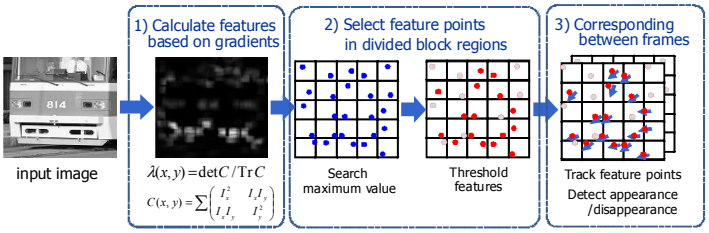



Fig. 7 Feature points selection

(3) Feature point tracking (Processing order: $O(M)$)

Based on the assumption that the displacement of a feature point between frames is small, only the 8×8 pixel local area, where a feature point belongs in the current frame, is searched with its adjacent local areas to set correspondence between the preceding and succeeding frames for tracking.

Therefore, processing of the order $O(N^2)$ implemented as hardware logic and processing of the processing order $O(M)$ is executed by software.

### B. Implementation and Operation

To deal with the parallel input of 10-bit contrast images in units of 16 pixels for H$^3$ Vision, the feature point selection circuit shown in Fig. 8 was implemented to parallelize the processing of image feature calculation and feature point selection of the above algorithm on 16 pixels.

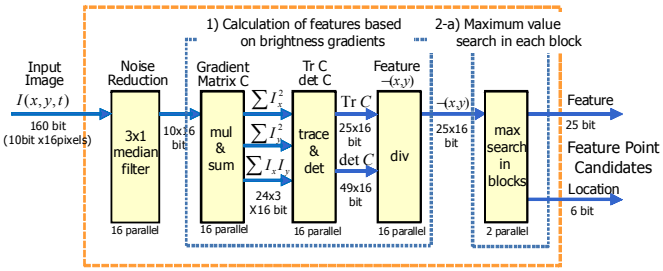
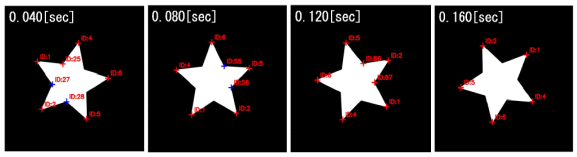



Fig. 8 Feature points selection circuit

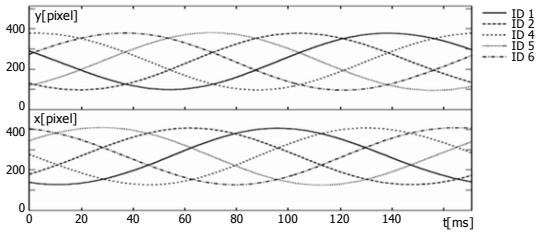
This circuit, working on 16 pixels in parallel, realized high-speed processing through parallel pipeline processing carried out in the following order: calculation of brightness gradient matrix and its trace and determinant, calculation of the image feature, and a maximum value search.

As the result of circuit implementation, the resource consumption of FPGA2 on the FPGA image processing board for $H^3$ Vision was Slice 77% (34243), Slice Register 69% (61204), LUT 26% (23350), Block RAM 31% (138), and MULT10X18 50% (224). Processing of 1024×1024 pixel images at 1000 fps was also confirmed.

Feature point tracking at 1000 fps was also conducted on a star-shaped object rotating at 6 rps. Fig. 9 (a) shows the captured object and positions of tracked feature points and (b) shows the changes in the coordinates of five feature points tracked for 0.2 second. The lower half of Fig. 10 shows the feature points when a soccer ball undergoing parallel transport was tracked. Thus, we could verify the automatic detection of convex vertices on a star-shaped object, the automatic tracking of feature points on an object rotating at a high velocity, and automatic tracking of feature points for movement of a soccer ball in the natural environment.




(a) extracted feature points




(b) tracking results ($x$-$t$, $y$-$t$)

Fig. 9 Tracked feature points (star-shaped object)

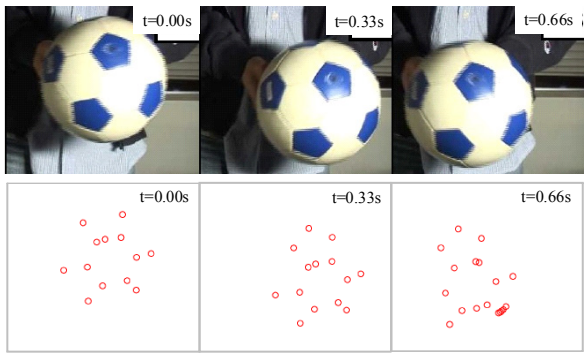



Fig. 10 Tracked feature points (soccer ball)

## VI. Optical Flow Processing

### A. Outline of Algorithm

Optical flow processing is used to measure velocity distribution in a moving image based on the time-space differential value of the image. The time-series differential method, represented by the Lucas-Kanade method [19], is based on the assumption that brightness $I(x, y, t)$ does not vary with time and local velocity $(v_x, v_y)$ is also uniform. This method enables to measure velocity distribution even without feature points in an image by solving $(v_x, v_y)$ in the following simultaneous equations with the sum product of differential values in a certain area, $S_{\xi\eta}(t) = \sum I_\xi I_\eta$,

$$\begin{cases} S_{xx}v_x + S_{xy}v_y + S_{xt} = 0 \\ S_{xy}v_y + S_{yy}v_y + S_{yt} = 0 \end{cases}. \quad (2)$$

The time-space differential method allows sub-pixel analysis but its estimation accuracy decreases as the velocity increases. This is caused by great image deviations between frames. The solution to this problem is operation at a high frame rate. The accuracy of analysis by high-speed vision, however, becomes low for an object moving at a low velocity because image deviations between frames become fine.

Therefore, we adopt a technique of automatically adjusting the frame rate in a pseudo way by adding a time-space differential value in the time direction according to the motion of an object. This technique uses the characteristic that the changes of a space differential image between frames become fine if an object moving at low velocity is captured at a high frame rate. To calculate the sum product of differential values, the frame rate is changed to $1/n$ times by the following approximation:

$$S_{xt}^n(t) = \frac{1}{n}\sum_{i=0}^{n-1} S_{xt}(t+i), \quad S_{yt}^n(t) = \frac{1}{n}\sum_{i=0}^{n-1} S_{yt}(t+i). \quad (3)$$

The cumulative count $n = n(x, y)$ is determined by the distribution of brightness changes in an image. In this paper, the frame rate was adjusted automatically in the above mentioned pseudo way until the sum product of differential values exceeds a threshold.

### B. Implementation and Operation

For $H^3$ Vision, optical flow processing at 1000 fps is realized by dividing a 10-bit contrast image of 1024×1024 pixels into 1024 areas of 32×32 pixels. The sum product of the differential values is calculated for each area by hardware, and postprocessing is done by software. The postprocessing includes the final flow calculation for each of 1024 areas and the removal of error vectors.

Fig. 11 is a block diagram of a differential sum-product

circuit that processes 16 pixels in parallel. A 10-bit contrast image is entered by parallel input in units of 16 pixels. By subtracting the image data at the same position in the preceding frame from the image data of the one line higher stored temporarily in memory, the upper stage of the circuit calculates the space and time differential values and also integrates them in parallel (48-parallel subtraction and 80-parallel integration). The lower stage of the circuit adds the integration results for 32×32 pixels to calculate the sum product of the differential values. This circuit uses parallel pipeline processing to increase the speed.

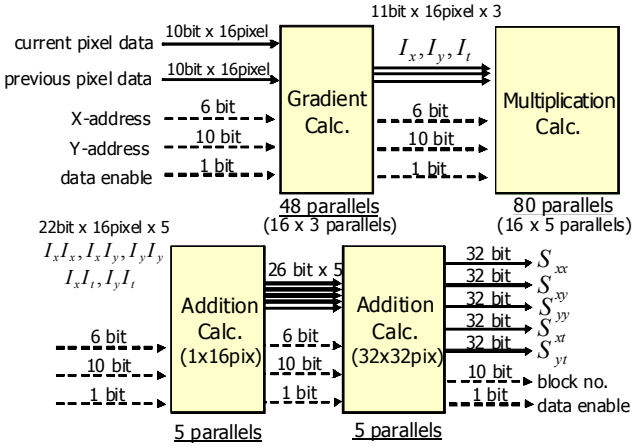
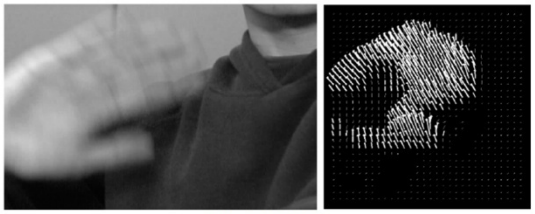


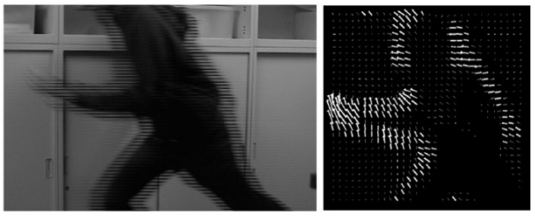

Fig. 11 Differential sum-product circuit for optical flow detection

As the result of implementing the sum product of the differential value calculation circuit and other as hardware, the resource consumption of FPGA2 on the FPGA image processing board for $H^3$ Vision was Slice 31% (13865), Slice Register 29% (25671), LUT 10% (9355), Block RAM 6% (28), and MULT10X18 16% (75). Processing of 1024×1024 pixel images at 1000 fps was confirmed.

Fig. 12 shows examples of optical flow analysis at 1000 fps where the cumulative count of the sum product of differential values was set to 1. Fig. 12 (a) is a hand-waving motion and (b) is a running motion. Optical flow calculation is realized even for quick human motion.



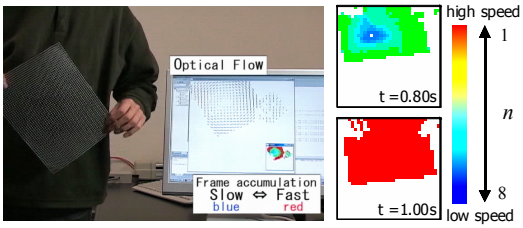
(a) moving hand



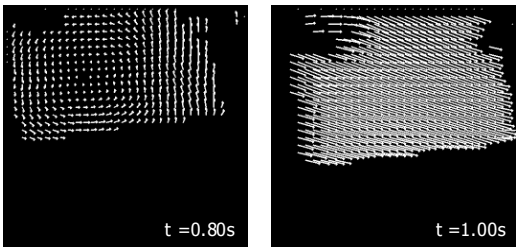
(b) running man

Fig. 12 Optical flow detection for human motion

Fig. 13 shows an example of optical flow analysis for a rotating object. In this example, the lower limit of flow velocity was set to 400 pixels/s and the maximum cumulative count of the sum product of differential values was set to $n$=8. From Fig. 13 (c), showing the cumulative count distribution of the sum product of differential values corresponding to the amplitude of flow velocity, we see that the cumulative count is set large in a low-velocity area and small in a high-velocity area. The cumulative count of the sum product of differential values is thus adjusted automatically to the velocity. Even where high-velocity and low-velocity motions are mixed, adaptive optical flow calculation is realized.




(a) experimental environment (b) $N(x,y)$ map




(c) estimated optical flow

Fig. 13 Optical flow detection for a rotating object

## VII. Pattern Recognition Using HLAC Features

### A. Outline of Algorithm

High-order local auto-correlation (HLAC) features [20] are calculated by using a reference pixel and its adjacent pixels from the following higher-order auto-correlation function:

$$\int_D I(\boldsymbol{r})I(\boldsymbol{r}+\boldsymbol{a}_1)I(\boldsymbol{r}+\boldsymbol{a}_2)\cdots d\boldsymbol{r}\,. \qquad (4)$$

The features are applied to face recognition and error detection as image features effective for shape recognition. Here, $D$ is the target image area, $\boldsymbol{r}$ is the reference pixel position, $I(\boldsymbol{r})$ is the brightness value of reference pixel $\boldsymbol{r}$, and $\boldsymbol{a}_n$ $(n = 1,2,\cdots)$ is the distance between the reference and adjacent pixels.

This paper discusses the secondary auto-correlation feature available from the reference pixel and eight adjacent pixels. As a pixel pattern used for feature calculations, an image area of 1024×1024 pixels is calculated at a frame rate of 1000 fps for the 25 patterns of HLAC features shown in Fig. 14. To satisfy the specifications, HLAC features calculation is speeded up by the hardware, and final

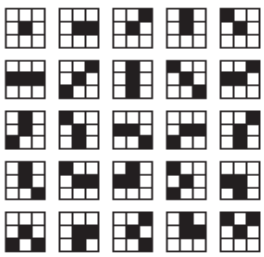


Fig. 14 Pattern of HLAC

recognition processing, such as nearest neighborhood method, is realized by software.

As Equation (4) shows, HLAC features are calculated by multiplication and addition only. To maintain integration and high speed, it is important to suppress the multiplication count that affects the circuit scale and calculation time. The primary feature calculation in Equation (5) is generally the multiplications of adjacent pixels and the calculation results can be used for the secondary feature calculation in Equation (6). In this paper, these calculations are used to suppress the multiplication count in calculating HLAC features.

$$\varepsilon_1(\boldsymbol{a}_1) = I(\boldsymbol{r})I(\boldsymbol{r}+\boldsymbol{a}_1)\,, \tag{5}$$

$$\begin{aligned}\varepsilon_2(\boldsymbol{a}_1,\boldsymbol{a}_2) &= I(\boldsymbol{r})I(\boldsymbol{r}+\boldsymbol{a}_1)I(\boldsymbol{r}+\boldsymbol{a}_2)\\ &= \varepsilon_1(\boldsymbol{a}_1)I(\boldsymbol{r}+\boldsymbol{a}_2).\end{aligned} \tag{6}$$

If the secondary features are calculated without using the primary feature multiplication results, the multiplication count per pixel will be 44 in total - four times that for primary feature calculation and 40 times that for secondary feature calculation. If the primary feature multiplication results are used, the multiplications in the primary feature calculation can be used for 17 of the 20 types of multiplications in the secondary feature calculation. This reduces the necessary multiplication count in the secondary feature calculation to 23, which represents a decrease of 17 per pixel.

### *B. Implementation and Operation*

For image recognition by HLAC features, a high-degree local correlation feature calculation circuit was implemented in $H^3$ Vision for the following processing:

1) Using the high-order 8 bits of a 10-bit brightness value
2) Completely parallelizing the multiplications in Equations (5) and (6) corresponding to the 25 patterns of features (requiring 26 multipliers)
3) Adding the multiplication results of the 25 patterns for 16 pixels after preparing 16 units of the above multiplication circuit to deal with 16-pixel parallel image input
4) Making incremental additions of the integral sum feature quantity for 16 pixels until the image scan finishes

For nearest neighborhood method using the 25 patterns of features and recognition processing based on the analysis, processes were performed by using software on a PC.

As the result of implementing the sum product of the differential value calculation circuit and other as hardware logic, the resource consumption of FPGA2 on the FPGA processing board for $H^3$ Vision was Slice 51% (22808), Slice Register 43% (38321), LUT 18% (16405), Block RAM 30% (137), and MULT10X18 93% (416). Processing of $1024\times1024$ pixel images at 1000 fps was also confirmed.

To confirm real-time image recognition based on HLAC features, we conducted an experiment on recognizing image patterns projected from a digital micro-mirror device (DMD) projector at high speed under the experimental environment shown in Fig. 15. For the experimental image patterns, two sets of image patterns; a) a total of 10 numbers from 0 to 9 in Fig. 16 number recognition, b) a total of 10 face images of 8-level brightness for gray-level pattern recognition. Here we picked up ten face images of H1-H10 in Fig. 17 from the face image database provided by AT&T Laboratories Cambridge [21]. The pattern projection speed of the DMD projector was set to a maximum of 500 fps.

For number recognition, 10 sets of the 25 HLAC features extracted from 0 to 9 were acquired off-line and image patterns were recognized as numbers by the nearest neighbor method based on learning data for 100 sets. For noise removal, thresholding by half-binarization was conducted. The frame rate of $H^3$ Vision was set to 1000 fps.

Fig. 18 shows the results of number recognition when image patterns were projected repeatedly in the order of 0 to 9 and again from 0. The pattern projection speed is 100 fps in (a) and 500 fps in (b). At pattern switching, there is a recognition failure time because the camera and projector are asynchronous. For both pattern projection speeds of 100 and 500 fps, however, number recognition results matched the patterns in the order of 0 to 9.

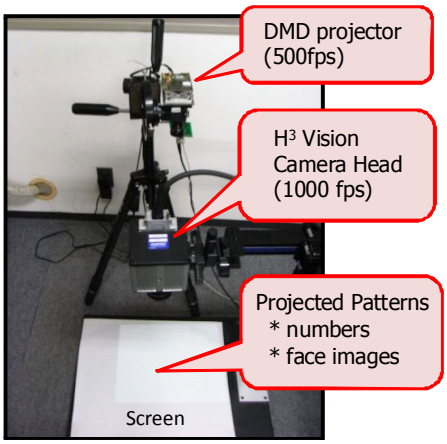



Fig. 15 Experimental environment for HLAC recognition

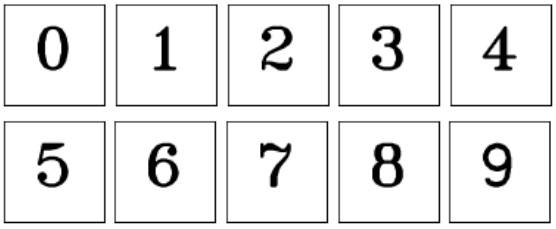
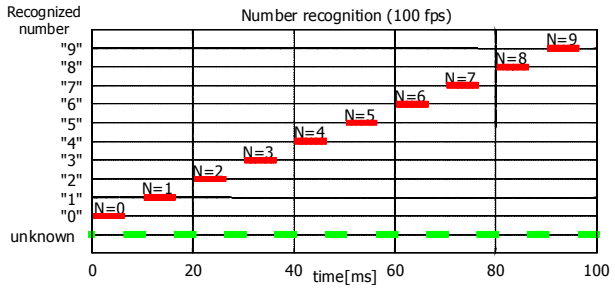
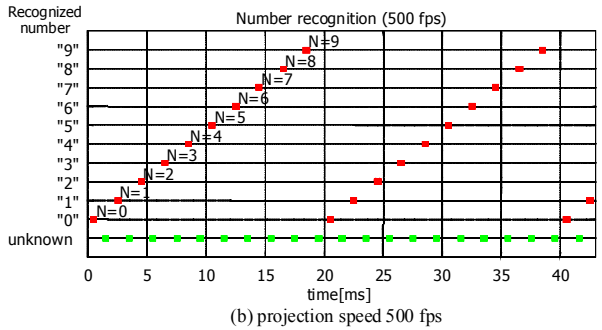



Fig. 16 Number patterns for experiment


Fig. 17 Face images for experiment




(a) projection speed 100 fps




(b) projection speed 500 fps

Fig. 18 Number recognition using HLAC features

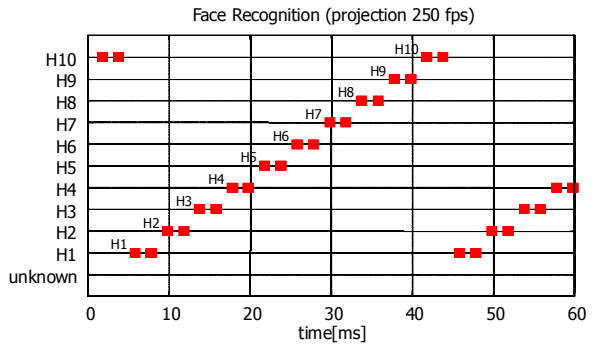



Fig. 19 Face image recognition using HLAC features

For face image pattern recognition, 100 sets of the 25 HLAC features extracted from the ten face images were acquired and were recognized as face numbers of H1-H10 by the nearest neighborhood based on learning data for 1000 sets in the same way as the number recognition experiment.

Fig. 19 shows the result of face image recognition when image patterns were projected repeatedly in the order of H1 to H10 of face images and again from H1. The pattern projection speed was 250 fps, and the frame rate of $H^3$ Vision was set to 500 fps in this experiment. Face image recognition results matched the patterns in the order of H1 to H10.

These experimental results indicate that real-time shape recognition was realized by using HLAC features for image pattern switching at high speeds.

## VIII. Conclusion

This paper introduced a high-speed vision platform $H^3$ Vision capable of processing 1024×1024 pixel images at 1000 fps and 256×256 pixel images at 10000 fps in real time. By conducting experiments, we presented examples of successful implementation of various image processing algorithms for color marker tracking, feature point tracking, optical flow, and high-order local auto-correlation feature calculation. Based on these results in developing the high-speed vision platform, we will further extend the system application in the robot control, factory automation, multimedia, and biomedical fields and make high-speed vision systems more functional, less costly, and more compact.

## References


[1] T.M. Bernard et al, "A Programmable Artificial Retina," *IEEE J. Solid-State Circuits*, Vol. 28, No. 7, pp. 789-797, 1993.

[2] J. Eklund, et al, "VLSI Implementation of a Focal Plane Image Processor - A Realization of the Near-sensor Image Processing Concept," *IEEE Trans. VLSI Systems*, Vol. 4, No. 3, pp. 322-335, 1996.

[3] T. Komuro et al, "A Dynamically Reconfigurable SIMD Processor for a Vision Chip," *IEEE J. Solid-State Circuits*, Vol. 39, No. 1, pp. 265-268, 2004.

[4] I. Ishii et al, "Higher Order Autocorrelation Vision Chip," *IEEE Trans. Electron Devices*, Vol.53, No.8, pp.1797-1804, 2006.

[5] K.Tajima et al, "Development of a High-resolution, High-speed vision System using CMOS Image Sensor Technology Enhanced by Intelligent Pixel Selection Technique," *Proc. SPIE*, Vol. 5603, pp. 215--224, 2004.

[6] Y. Watanabe et al, "955-Fps Real-Time Shape Measurement of a Moving/Deforming Object Using High-Speed Vision for Numerous -Point Analysis," *Proc. IEEE Int. Conf. Robotics and Automation,* pp.3192-3197, 2007.

[7] S. Hirai et al, "Realtime FPGA-Based Vision System," *J. Robotics and Mechatronics*, Vol.17, No.4, pp.401-409, 2005.

[8] I. Ishii et al, "Ultrafast Hyper Human Vision and Its Application," *J. Institute of Electronics, Information, and Communication Engineers*, Vol.90, No.10, pp. 838-841, 2007. (in Japanese)

[9] I. Ishii et al, "Target Tracking Algorithm for 1ms Visual Feedback System Using Massively Parallel Processing," *Proc. IEEE Int. Conf. Robotics and Automation*, pp.2309-2314, 1996.

[10] A. Namiki et al, "Development of a High-speed Multifingered Hand System and Its Application to Catching," *Proc. IEEE/RSJ Int. Conf. Intelligent Robots and Systems*, pp.2666-2671, 2003.

[11] T. Senoo et al, "High-Speed Batting Using a Multi-Jointed Manipulator," *Proc. IEEE Int. Conf. Robotics and Automation*, pp.1191-1196, 2004.

[12] Y. Nakamura et al, "Heartbeat Synchronization for Robotic Cardiac Surgery," *Proc. IEEE Int. Conf. Robotics and Automation*, pp.2014-2019, 2001.

[13] H. Oku et al, "Tracking a Protozoon Using High-Speed Visual Feedback," *Proc. Annu. Int. IEEE-EMBS Special Topic Conf, Microtechnologies in Medicine & Biology*, pp.156-159, 2001.

[14] K. Yamamoto et al, "A Real-time Finger-tapping Interface Using High-speed Vision System," *Proc. IEEE Int. Conf. Systems, Man, and Cybernetics*, pp.296-303, 2006.

[15] I. Ishii et al, "Automatic Scratching Pattern Detection for Laboratory Mice using High-speed Video Images," *IEEE Trans. Automation Science and Engineering*, Vol.5, No.1, pp.176-182, 2008.

[16] M. Kaneko et al, "Dynamic Sensing of Human Eye," *Proc. IEEE Int. Conf. Robotics and Automation*, pp.2882-2887, 2005.

[17] Photron Ltd, http://www.photron.co.jp

[18] J. Shi et al, "Good Features to Track," *Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition*, pp.593-600, 1994.

[19] B.D. Lucas et al, "An Iterative Image Registration Technique with an Application to Stereo Vision," *Proc. Imaging Understanding Workshop*, pp 121-130, 1981.

[20] N. Otsu et al, "A New Scheme for Practical, Flexible and Intelligent Vision Systems," *Proc. IAPR Workshop Computer Vision*, pp.431-435, 1988.

[21] F.S. Samaria et al, "Parameterisation of a stochastic model for human faceidentification," *Proc. IEEE Workshop on Applications of Computer Vision*, pp.138-142, 1994.