Fast SIFT Design for Real-Time Visual Feature Extraction

Liang-Chi Chiu, Tian-Sheuan Chang, *Senior Member, IEEE*, Jiun-Yen Chen, and Nelson Yen-Chung Chang

***Abstract*— Visual feature extraction with scale invariant feature transform (SIFT) is widely used for object recognition. However, its real-time implementation suffers from long latency, heavy computation, and high memory storage because of its frame level computation with iterated Gaussian blur operations. Thus, this paper proposes a layer parallel SIFT (LPSIFT) with integral image, and its parallel hardware design with an on-the-fly feature extraction flow for real-time application needs. Compared with the original SIFT algorithm, the proposed approach reduces the computational amount by 90% and memory usage by 95%. The final implementation uses 580-K gate count with 90-nm CMOS technology, and offers 6000 feature points/frame for VGA images at 30 frames/s and ∼2000 feature points/frame for 1920 x 1080 images at 30 frames/s at the clock rate of 100 MHz.**

***Index Terms*— Feature extraction, SIFT, VLSI design.**

I. Introduction

VISUAL feature extraction is a key technology of computer vision for intelligent video processing. Among extraction techniques, scale invariant feature transform (SIFT) [1] is the most widely adopted approach that provides stable visual feature points for reliable object detection. SIFT detects and describes local feature points in images for object detection in different scenes or different angles. These feature points describe the strength and direction of the object, which resists the frame distortion such as zooming, rotation, movement, perspective change, shading and noise. To provide these stable points, SIFT computation involves the image convolution with Gaussian filters at different scales, and local maximum or minimum of the difference of Gaussians (DoG) blurred images at multiple scales. However, the real time implementation of this algorithm faces the challenges of heavy computation, large memory storage and long computational latency because of its frame level computation with iterated Gaussian blur operations on images and the frame difference operations on blurred images for feature extraction. Thus, a fast algorithm and its VLSI design are demanded.

Manuscript received November 5, 2012; revised February 19, 2013; accepted April 19, 2013. Date of publication April 24, 2013; date of current version June 4, 2013. This work was supported in part by the Industrial Technology Research Institute under Grant A301AR3710. The associate editor coordinating the review of this manuscript and approving it for publication was Prof. Mark H.-Y. Liao.

L.-C. Chiu is with PixelArt Technology, Hsinchu 78229, Taiwan (e-mail: oboe.chu@gmail.com).

T.-S. Chang is with the Dept. Electronics Engineering, National Chiao Tung University, Hsinchu 300, Taiwan (e-mail: tschang@twins.ee.nctu.edu.tw).

J. Y. Chen and N. Y. C. Chang are with the Industrial Technology Research Institute, Hsinchu 31040, Taiwan (e-mail: itri990075@itri.org.tw; NelsonChang@itri.org.tw).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIP.2013.2259841

Previous implementations of SIFT can be classified into following categories: multi-core, GPU, or FPGA based approaches. The multi-core [3], [4] or GPU approaches [5]–[8] used massive parallel computing resource to speed up the computation. Works [9]–[12] based on the FPGA platform used parallel FPGA resources to speed up the Gaussian blurring in SIFT. However, the corresponding feature point calculation was still computed by the processor, which results in heavy data bus congestion and reduces the performance. Beyond above implementations, the fast algorithm like SURF [2] used the integral image concept to speed up the Gaussian blur computation, which was followed by several works [13]–[15]. Integral image reduces the computation significantly. However, all these approaches still need iterations of Gaussian blur, which hinders their real time applications on larger frame size and increasing number of features points.

To solve above issues, this paper proposed a layer parallel SIFT (LPSIFT) with integral image and its parallel hardware design for real time application needs. First, to avoid the long latency due to the frame level computation, we adopted the layer parallel restructured box kernel to replace iterated Gaussian blur operations. The computation of box kernel was further simplified by the integral image approach with reuse of sub-kernel sum. With this, the latency of the first feature point was reduced to several image lines (depending on the location of that feature point) from several frames. For the keypoint localization, we simplified the complex low contrast analysis to be a low brightness test. For hardware design, we adopted the on-the-fly feature extraction flow so that only partial temporal results have to be stored. Furthermore, the costly inverse square root and divider in keypoint localization were implemented by a low cost universal operation unit with precision equivalent cycles (PECs) to reduce the gate count. With above methods, SIFT can be implemented with low hardware cost for real time high definition videos.

The rest of the paper is organized as follows. In Section II, we will briefly describe the operation of SIFT. In Section III, we will present the proposed algorithm. Its corresponding hardware design is shown in Section IV. Then, in Section IV, we will show the implementation results and comparison. Finally, a conclusion will be made in Section V.

II. Overview of SIFT

Fig. 1 shows the overall algorithm flow of SIFT [1], which consists of scale space extrema detection, accurate keypoint localization, orientation assignment and the local image descriptor.

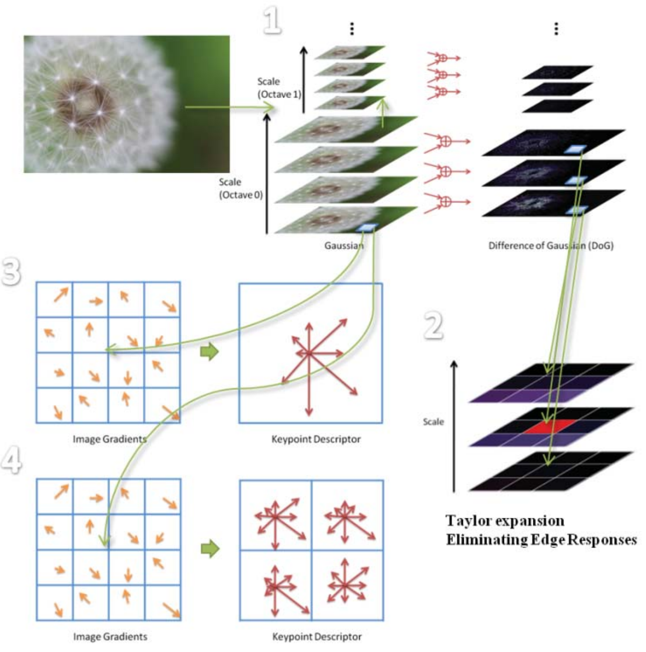


Fig. 1. SIFT algorithm: (1) scale-space extrema detection, (2) accurate keypoint localization, (3) orientation assignment, and (4) the local image descriptor.

The first step, scale space extrema detection, identifies keypoint candidates. It first convolves the image S_k with Gaussian filters $GF(x, y, k\sigma)$ at different scale $k\sigma$ to build a pyramid of Gaussian blurred images. In which,

$$S_k = S(x, y, k\sigma) = GF(x, y, k\sigma) * S_{k-1} \tag{1}$$

where for $k = 1$, $S_{k-1} = S_0 = I(x, y)$ is the input image, and

$$GF(x, y, k\sigma) = \left(\frac{1}{2}\pi(k\sigma)^2\right) e^{-\frac{(x^2+y^2)}{2(k\sigma)^2}}. \tag{2}$$

The convolved images are grouped by octave (an octave corresponds to doubling the value of σ), and the value of k is selected for a fixed number of convolved images per octave. The frame at the highest scale per octave is downsampled by four as the input frame for next octave, as depicted in Fig. 1. In this paper, we denote the frame in a pyramid of Gaussian blurred images by its octave and scale, G[Octave, Scale]. With this, the Gaussian pyramid can be defined as follows:

$$G[0, 0] = I(x, y): \text{ the original frame} \tag{3}$$

$$G[0, 1] = GF(x, y, \sigma) * G[0, 0] \tag{4}$$

$$G[m, k] = GF(x, y, k\sigma) * G[m, k-1] \quad \text{for } m, k \geq 0 \tag{5}$$

where $G[m, 0]$ is obtained by downsampling $G[m-1,$ *maximum* k in a octave] by four.

Then, we can compute the DoG blurred images by

$$\text{DoG}[m, k] = G[m, k+1] - G[m, k] \tag{6}$$

with DoG, we can find the local maximum or minimum in a 3×3 window as the keypoint candidates by comparing the pixels at neighboring scales.

Among the candidates, stable keypoints are localized by rejecting some bad keypoints in the keypoint localization step with a low contrast and edge detection test. The low contrast test rejects the unstable extrema location, defined by

$$\hat{x} = -\left(\frac{\partial^2 \text{DoG}^{-1}}{\partial x^2}\right)\left(\frac{\partial \text{DoG}}{\partial x}\right) \tag{7}$$

with a low contrast threshold derived from the Taylor expansion of DoG$[m, k]$

$$\text{DoG}(\hat{x}) = \text{DoG} + \frac{1}{2}\left(\frac{\partial \text{DoG}^T}{\partial x}\right)\hat{x}. \tag{8}$$

Edge detection is decided by

$$\frac{\text{Tr}(H)^2}{\text{Det}(H)} < \frac{(r+1)^2}{r} \tag{9}$$

where r is a coefficient to adapt the sharpness of object edge, and Tr(H) and Det(H) are the trace and determinant of 2×2 Hessian matrix

$$H = \begin{vmatrix} \text{DoG}_{xx} & \text{DoG}_{xy} \\ \text{DoG}_{xy} & \text{DoG}_{yy} \end{vmatrix} \tag{10}$$

$$\text{Tr}(H) = \text{DoG}_{xx} + \text{DoG}_{yy} = \alpha + \beta \tag{11}$$

$$\text{Det}(H) = \text{DoG}_{xx}\text{DoG}_{yy} - \text{DoG}_{xy}{}^2 = \alpha\beta. \tag{12}$$

Based on above test, if the candidate is verified to be a feature point, SIFT begins to calculate its orientation assignment.

In the orientation assignment step, each keypoint is assigned one or more orientations based on local image gradient directions to achieve invariance to rotation. In which, SIFT selects the largest vector, S-vector, to represent the orientation assignment. These steps ensure invariance to image location, scale and rotation. Finally, in the local image detector step, a descriptor vector for each keypoint is computed such that it is highly distinctive and partially invariant to other variations.

III. SIFT Analysis and Proposed LPSIFT Design

A. Design Analysis of SIFT

Table I shows the design analysis of SIFT for parameters [octave,scale] = [2, 4]. In this table, A is the number of pixels in a frame and its first level down sampled frame defined as

$$A = WH + \frac{1}{4}WH = 1\frac{1}{4}WH, \tag{13}$$

where W and H are width and height of the original frame. B represents the number of feature points.

From this table, we can find that the scale-space extrema detection and the keypoint localization contribute to most of overall computation, and require a lot of memory to save intermediate data frames. In addition, the keypoint localization, orientation assignment, and the local frame descriptor calculation use a lot of dividers to compute Taylor expansion and normalization. This causes hardware implementation difficult. Thus, how to solve these problems is an important issue for SIFT hardware design.

Another problem for SIFT hardware is its highly data dependent computational structure. In the data flow of SIFT, a new scale image has to wait for the completion of the previous scale image in the Gaussian pyramid. Thus, different scale images are computed sequentially. This results in long latency and prohibits the following DoG computation in parallel that needs

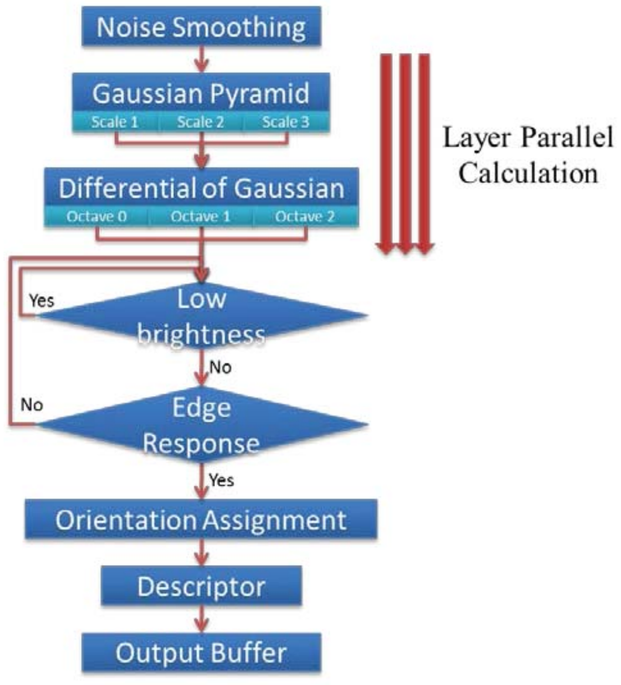


Fig. 2. Algorithm flow of the proposed LPSIFT.

at least three scale images at the same time to calculate the candidates of features points. Thus, these intermediate scale images have to be stored until completing DoG computations, which results in large memory storage.

B. Proposed LPSIFT Algorithm

Fig. 2 shows the proposed LPSIFT algorithm, which is based on SIFT with three major modifications: fast scale space extreme detection with layer parallel Gaussian pyramid and integral image, and simplified keypoint localization with a brightness threshold.

In this flow, we propose a layer parallel algorithm to solve the data dependency problem, which not only calculate Gaussian pyramid and DoG pyramid at the same time but also calculate keypoint candidates simultaneously. This parallel computation reduces storage and latency from multiple whole images to several image rows. To assist this parallel computation while reduce the computational complexity, we use the integral image approach to create a box kernel with the response similar to that of the Gaussian filter so as to keep the feature matching performance as close as that of SIFT but without complex Gaussian filters. By adopting the integral image approach for multiple box kernels in each scale, we reduce computational complexity significantly and enable parallel computation for all scales.

Another problem is the high complexity of Taylor expansion. We propose a low brightness method instead of the low contrast method to reduce complexity.

1) Fast Scale Space Extreme Detection With Layer Parallel Gaussian Pyramid and Integral Image: Fig. 3 shows the concept of the layer parallel Gaussian pyramid. In the original Gaussian pyramid, each new pyramid level depends on its

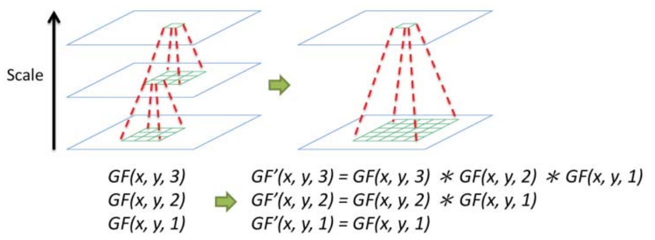

Scale
GF(x, y, 3)
GF(x, y, 2)
GF(x, y, 1)
GF'(x, y, 3) = GF(x, y, 3) * GF(x, y, 2) * GF(x, y, 1)
GF'(x, y, 2) = GF(x, y, 2) * GF(x, y, 1)
GF'(x, y, 1) = GF(x, y, 1)

Fig. 3. Concept of the layer parallel Gaussian pyramid.

previous level. To eliminate such dependency, we can fully expand the pyramid level and merge the multiple kernels into one

$$\begin{aligned} G[m,k] &= GF(x,y,k\sigma) * G[m,k-1] \\ &= (GF(x,y,k\sigma) * GF(x,y,(k-1)\sigma) \cdots \\ &\quad * GF(x,y,\sigma)) * G[m,0] \\ &= GF'(x,y,k) * G[m,0] \\ &\qquad \text{for positive integer } m, \text{and } k. \end{aligned} \tag{14}$$

With this independent Gaussian filtering, we can compute all scale layers in parallel and generate the DoG images on demand.

However, the merged kernel will become larger with more kernels merged as shown in Fig. 3, which will demand high computational complexity and storage. To solve this, we use the integral image concept that can compute this large kernel with constant time. By using integral image, no matter what the kernel size is, we need only four points of information and three additions, and reduce a lot of computation. With this simple computation, we can compute all scale layers in parallel without worry about the heavy complexity and storage due to layer parallel computation.

The integral image works best for the box kernel, but the box kernel results in inferior matching performance. Thus, we use restructured box kernels to approach the performance by the Gaussian filter. Take the first scale 5×5 box kernel in Fig. 4(c) as an example. We modify this simple kernel to be the kernel in Fig. 4(d) to approach the performance of Gaussian filter. Its computation with integral image can be very simple as shown in Fig. 4(b). In this paper, we select the size of box kernels as 5×5, 7×7, and 3×3 to construct the filters for different scales with appropriate modifications as the example illustrated in Fig. 4.

2) Simplified Keypoint Localization With a Brightness Threshold: SIFT uses Taylor expansion to exclude the low contrast candidates, but Taylor expansion needs complex computation. In practical, the low contrast candidates almost have low brightness, too. Therefore, low brightness candidates can be excluded according to this threshold

$$\text{threshold}_{\text{new}} = (2^{\text{precision of DoG}} - 1) * \text{coefficient}_{\text{bright}}.$$

In which, Coefficient$_{\text{bright}}$ is a ratio of the gray level luminance in the DoG image with the default value 0.04. This simplification has similar performance as that by Taylor expansion according to our simulation, but this method saves a lot of calculation and is more suitable for hardware computation.

TABLE I
COMPUTATIONAL COMPLEXITY AND MEMORY STORAGE ANALYSIS OF SIFT

	Memory (byte)	Multiplication	Division	Addition
Smoothing	A	81 A	0	80 A
Scale-space extrema	4 A	2916 A	0	2933 A
Keypoint localization	3 A	59 A	50 A	132 B
Orientation assignment	0	450 B	225 B	673 B
The local frame descriptor	0	450 B	225 B	705 B
Total	8 A	3056 A + 900 B	50 A + 450 B	3013 A + 1510 B

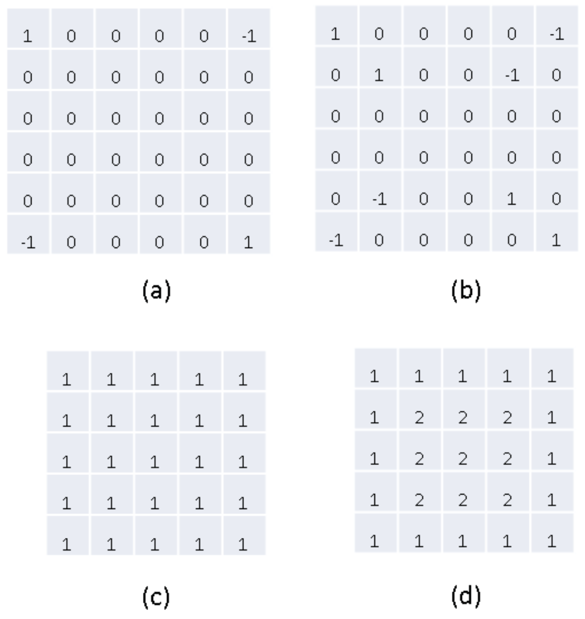


Fig. 4. Mask for 5 × 5 box kernel, its integral image representation of (a) original one (b) and modified one, and the corresponding pixel domain of (c) original one, (d) and modified one.

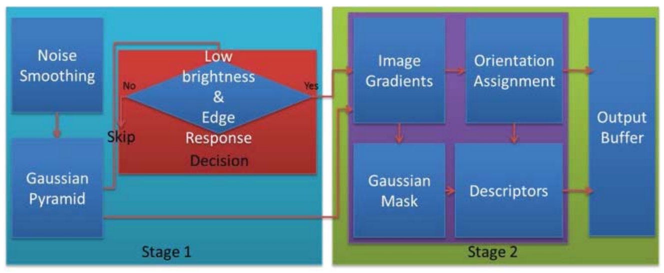


Fig. 5. Proposed architecture diagram.

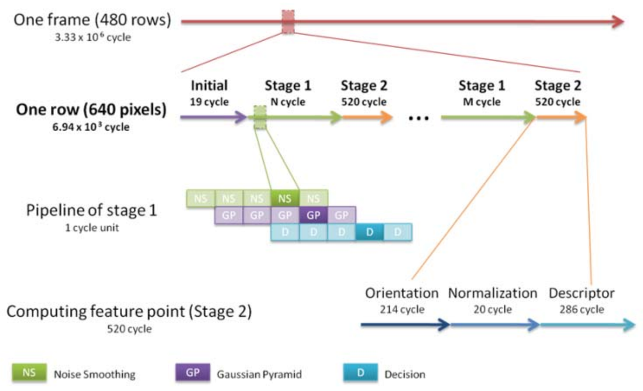


Fig. 6. Scheduling for a VGA image.

Table II shows the complexity analysis of our proposed algorithm for parameters [octave,scale] = $[2, 4]$, with the comparison of VGA and full HD images in Table III. In Table II, C is the number of frame width in a group of down sample twice ($C = W + (1/4)W = 1(1/4)W$), and W is the width of the original frame. In the first three steps, computation with integral image only uses additions to eliminate all multiplications and data dependent computation. In the keypoint localization, divisions to compute Taylor expansion are also eliminated. Beyond above computation reduction, we only need a few rows of the integral image for the box kernel based computation instead the whole integral image. In summary, the proposed algorithm can reduce a lot of computation and memory usage, which are useful for the following hardware implementation.

IV. PROPOSED ARCHITECTURE

Fig. 5 shows the proposed architecture that consists of two stages: scale-space extrema detection and keypoint localization in the stage one, and orientation assignment and the local frame descriptor in the stage two. In the stage one, we adopt the proposed LPSIFT with integral image to reduce the computation significantly, and further reduce the required integral image buffer by the on-the-fly computation scheduling. In the stage two, we adopt the proposed brightness threshold to simplify computation and implement the normalization in keypoint localization with a low cost universal operation unit with precision equivalent cycles (PEC).

Fig. 6 shows the proposed scheduling for a VGA image, operated row by row. Our proposed design specification is [octave, scale] = [2, 4], and supports more than 1000 feature points in one VGA frame with 30 frames per second at 100MHz. Each row spends 19 cycles to calculate the initial information in the beginning. After that we find the feature point candidates of the stage one. If a candidate is a feature point, the stage two starts to compute the required information of that feature point. When the former step completes, operation is back to the stage one. In summary, the computation time of the stage one is not fixed until a feature point is found, while the stage 2 takes 520 cycles.

TABLE II
COMPUTATION COMPLEXITY OF PROPOSED ALGORITHM

	Memory (byte)	Multiplication	Division	Addition
Smoothing	7 C	0	0	58A
Integral image	80 C	0	0	3 A
Scale-space extrema	893	0	0	383 A
Keypoint localization	27	27 A	0	12 A
Orientation assignment	256	450 B	225 B	673 B
The local frame descriptor	0	450 B	225 B	705 B
Total	87 C + 1176	27 A + 900 B	450 B	456 A + 1378 B

TABLE III
COMPARISON OF PROPOSED AND SIFT ALGORITHM

	VGA Size				Full HD Size			
	Mem. (byte)	Mul.	Div.	Add.	Mem. (byte)	Mul.	Div.	Add.
SIFT	3.07 E+06	1.17 E+09	1.97 E+07	1.16 E+09	2.07 E+07	7.92 E+09	1.30 E+08	7.81 E+09
Proposed	7.08 E+04	1.13 E+07	4.50 E+05	1.76 E+08	2.10 E+05	7.09 E+07	4.50 E+05	1.18 E+09
Saving (%)	97.7	99.0	97.7	84.8	99.0	99.1	99.7	84.9

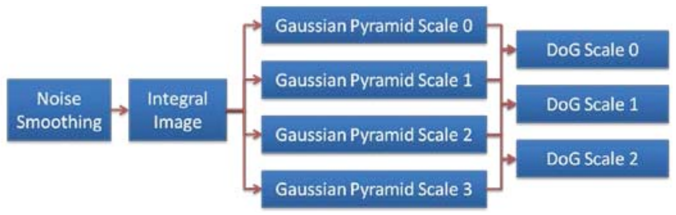


Fig. 7. Architecture of the stage one.

A. Stage One With On-the-Fly Computation Scheduling

Fig. 7 shows the architecture of the stage one, which applies the noise smoothing to the input image and builds the integral image for following filtering operations. With enough rows of integral image, we begin to compute scale images and corresponding DoG images to find feature point candidates at the same time as shown in Fig. 8, called on-the-fly computation. In which, when we calculate the last frame in the first layer ($G[0, 3]$), we will downsample this frame to the first frame of next layer ($G[1, 0]$) simultaneously. Therefore, we not only compute different scales in an octave at the same time but also compute different octaves at the same time. With this on-the-fly computation, we need only a few rows of buffer cost instead of multiple frames due to the Gaussian pyramid and DoG pyramid.

B. Architecture of the Stage Two

Fig. 9 shows the design of the keypoint localization after obtaining the DoG. We can compute the feature point candidate by examining conditions such as the local maximum and minimum value, low brightness and edge detection. Since all these conditions are independent to each other, we can compute each condition simultaneously with a flag bit to denote if it matches the feature point criteria. If all flag bits are zeros, that indicates the candidate is approved to be a new feature point.

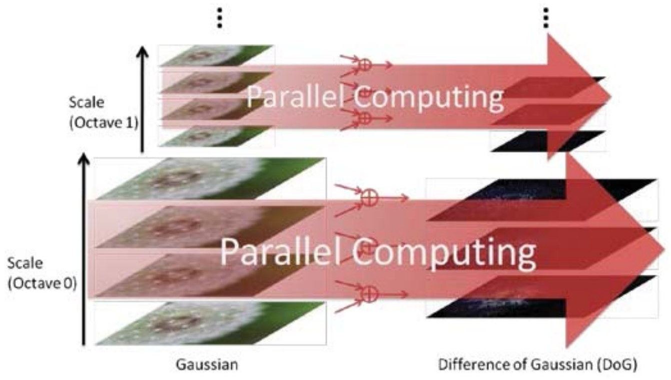


Fig. 8. Parallel computation of the Gaussian pyramid, and DoG pyramid to find feature points.

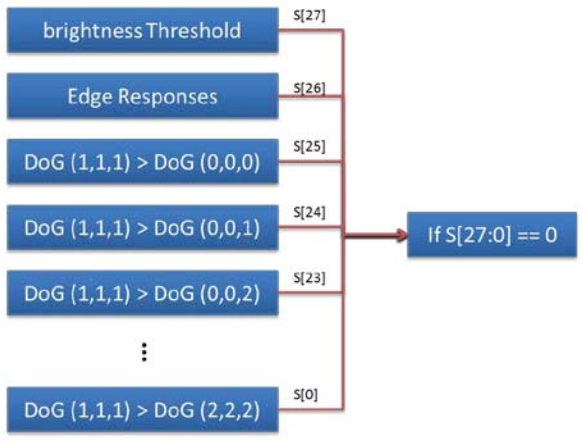

brightness Threshold
Edge Responses
DoG (1,1,1) > DoG (0,0,0)
DoG (1,1,1) > DoG (0,0,1)
DoG (1,1,1) > DoG (0,0,2)
DoG (1,1,1) > DoG (2,2,2)
S[27]
S[26]
S[25]
S[24]
S[23]
S[0]
If S[27:0] == 0

Fig. 9. Hardware architecture of the keypoint localization.

Fig. 10 shows the design of the orientation assignment. In this design, 19×19 pixels of integral image will be fetched again to re-compute 15×15 information of scale 1 since no scale image is stored. In this recomputation step, we will

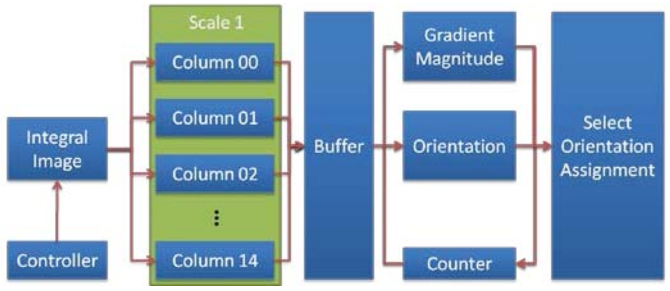


Fig. 10. Hardware architecture of the orientation assignment.

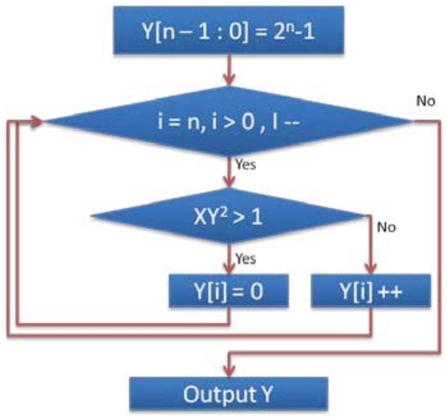


Fig. 11. Flow chart of the universal operation with precision equivalent cycle.

compute one row at a time, and store the results at the buffer. With three rows of data, we can compute the gradient magnitude and orientation. However, these computations involve the square root, inverse trigonometric and inverse square root.

To avoid high cost of single cycle design, we design a shared multi-cycle component for the inverse square root and division, and implement the inverse trigonometric function with a look-up table and the multi-cycle component.

Fig. 11 shows the flow chart to implement the inverse square root, $Y = 1/\sqrt{x}$ with one cycle per quotient bit, and n is the precision of Y. The operation is as follows. First, above equation can be rewritten as $X^2Y = 1$, where X is a known input and Y is the desired answer. Assume Y is the maximum value at its accuracy now. If $X^2Y > 1$, it implies that Y is too large. Thus, we change the current bit of Y with 0, and compute it again. If $X^2Y < 1$, it implies that Y is too small. Thus, we will add one to this bit and compute it again. Repeat above steps to reach the lowest bit of Y, and finish the calculation. For division, we need to replace X with X^2. Similar derivation can be applied to the square root operation. The total cycle number is equal to the precision of Y. This feature meets the precision limited hardware design because the fixed cycle latency eases the overall scheduling and controller design.

V. Simulation and Implementation Results

A. Simulation Results

Fig. 12 and Fig. 13 show the training patterns for threshold decision and test patterns used in this paper. The first part is mainly monotonous indoor frames with few objects, which is difficult to find feature points. The second part selects the monotonous, regular, short personal distance, indoor, outdoor and crowd images for test.

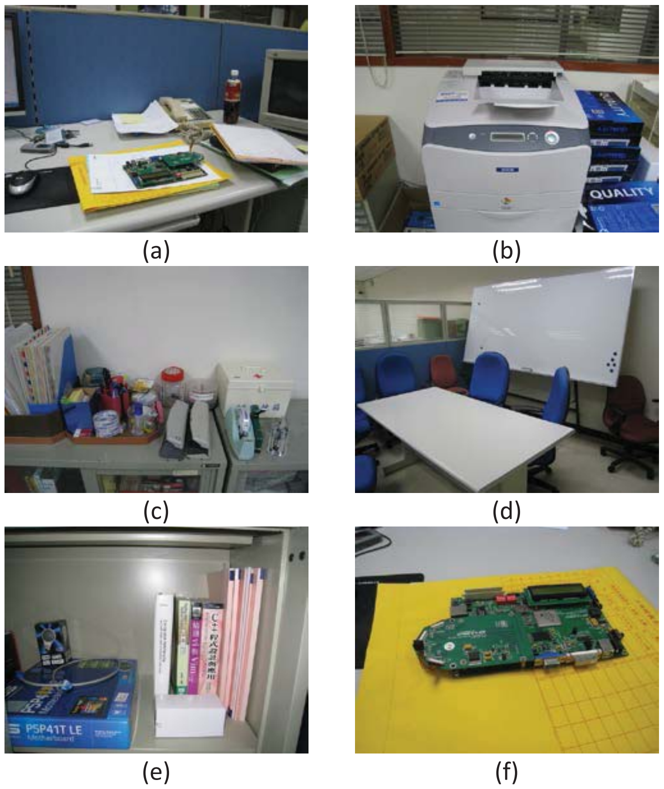


Fig. 12. The training patterns, (a) desktop, (b) photocopier, (c) iron cabinet, (d) meeting room, (e) books, (f) FPGA board.

TABLE IV
Test Conditions Used in the Simulation

	Unit	Best Case	Normal Case	Worst Case
Light	(Mean, variance)	(10, 10)	(10, 30)	(10, 50)
Focus	σ	1	1	3
Rotation	Angle	3	10	40
Stretch	Ratio	0.98	0.8	0.6
Movement	(x-axis, y-axis)	(−10, −5)	(−30, −30)	(−60, −60)

The simulation emphasizes on the following conditions: movement toward random directions, the change of lights in scenes light, focus, rotation in different angles and stretch. Table IV shows the test conditions and their parameters. The best case will be like images taken by the burst mode shooting. The normal case will be like images taken by two persons on the same scene but with some relative motion. The worst case will be like images taken casually by two persons, which is rarely occurred in feature extraction applications. Fig. 14 shows the matching results and images for different cases.

The performance is measured by the accuracy defined as

$$\text{Accuracy} = \frac{\text{Number of Correct Match Points}}{\text{Number of Match Points}}$$

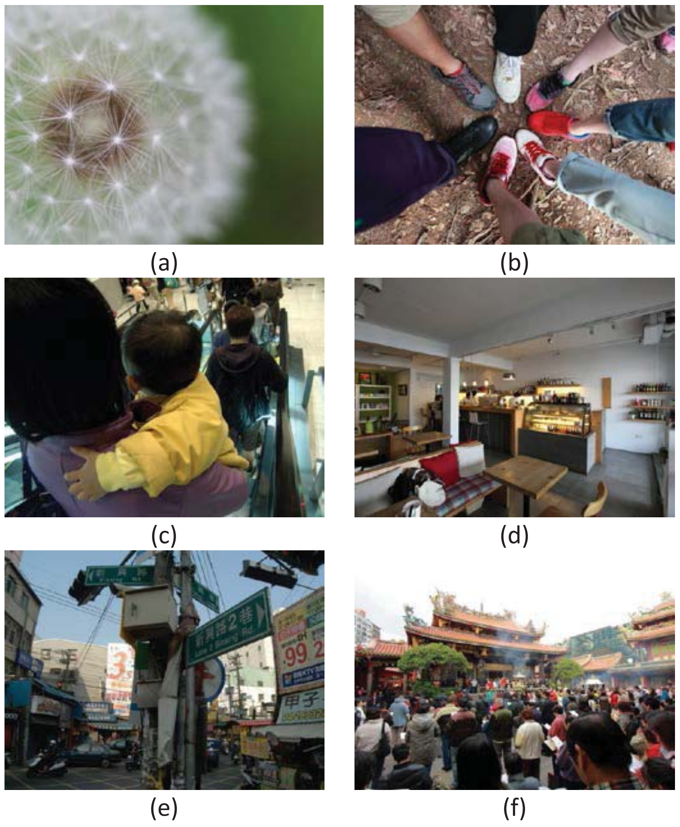


Fig. 13. The test patterns, (a) Dandelion, (b) shoes, (c) person, (d) cafe, (e) road row, (f) temple.

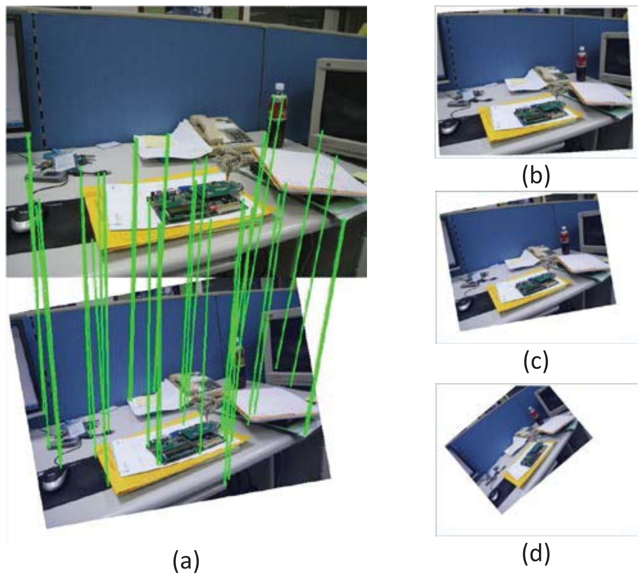


Fig. 14. (a) Matching point for normal case. Test images at (b) the best case, (c) the normal case, (d) the worst case.

where match points means that feature points in two different frames have the same local image descriptors, and correct match points mean that two feature points with the identical information have the same coordinates after the transformation of the golden pattern. If their coordinates are different, the match point is incorrect.

TABLE V
RESULTS OF THE NORMAL CASE FOR TRAINING PATTERNS

Pattern	Feature Points on Original Frame	Feature Points on Conversion Frame	Match Point	Correct Match Point	Accuracy (%)
Desktop	956	1447	66	55	83.33
Photocopier	936	1525	43	33	76.74
Cabinet	254	839	36	31	86.11
Whiteboard	455	1159	51	43	84.31
Books	708	1245	33	25	75.76
FPGA board	476	543	39	33	84.62

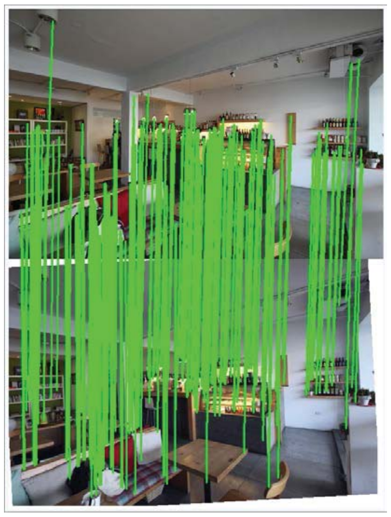
TABLE VI
ACCURACY COMPARISON WITH SIFT FOR TRAINING PATTERNS

	Best Case		Normal Case		Worst Case	
Pattern	SIFT	Ours	SIFT	Ours	SIFT	Ours
Desktop	97.41	90.00	89.80	83.33	100.00	71.93
Photocopier	96.32	88.65	95.30	76.74	97.96	70.91
Cabinet	98.23	96.70	97.85	86.11	100.00	74.19
Whiteboard	93.94	84.78	96.00	84.31	91.67	81.82
Books	98.98	96.88	93.33	75.76	97.22	70.97
FPGA board	98.80	96.09	96.08	84.62	97.30	54.55
Average	97.28	92.18	94.73	81.81	97.36	70.73

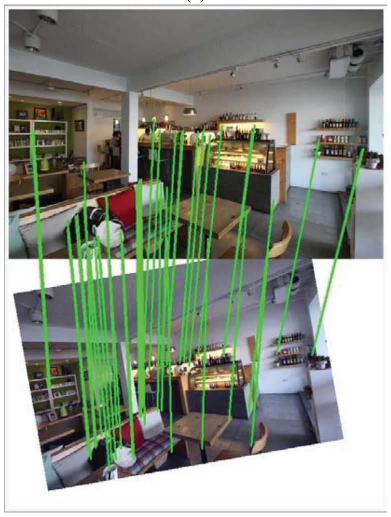
TABLE VII
ACCURACY COMPARISON WITH SIFT FOR TEST PATTERNS

	Best Case		Normal Case		Worst Case	
Pattern	SIFT	Ours	SIFT	Ours	SIFT	Ours
Dandelion	99.02	92.62	94.48	87.23	97.37	64.10
Shoes	96.75	91.73	91.39	83.33	92.47	64.71
Person	99.07	95.24	99.44	89.47	100.00	68.75
Café	96.25	81.58	92.31	88.89	80.95	88.89
Road row	99.35	97.22	98.33	88.89	97.58	50.00
Temple	98.48	88.41	98.95	100.00	98.72	64.29
Average	98.15	91.13	95.82	89.64	94.52	66.79

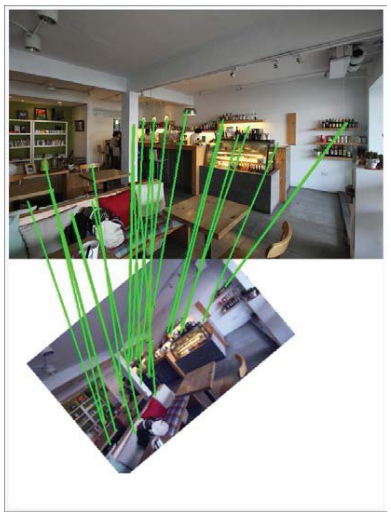
The simulation result shows that the average accuracy for training pattern is 71%, 82%, and 92% for worst, normal and best cases respectively, 63%, 92% and 98% for test patterns at worst, normal and best cases respectively. Table V shows the detailed results of the normal case for training patterns. Tables VI and VII shows the accuracy comparisons with SIFT. The SIFT accuracy is more than 90 % but also much complex. In contrast, our proposed algorithm has much lower complexity and approaches to the accuracy of original SIFT at the best case and normal case. The lower performance of the worst case is because our design cannot handle images with dramatic change of focus and stretch very well. The reason for that is that the proposed approach adopts integer instead of floating point format and only two octaves instead of more octaves. Fig. 15 shows the matching result of the image *cafe* for different cases.



(a)



(b)



(c)

Fig. 15. Matching result for (a) best case (b) normal case (c) worst case.

B. Implementation Results

Table VIII shows the comparison of FPGA implementations. Our design supports about 6000 feature points per frame at VGA 30 frames/sec and about 2000 feature points per frame at HD1080p 30 frames/sec, which is due to the proposed layer parallel scheme. Although we spend more LUTs and registers to compute LPSIFT with integral image, our design only uses 8 DSP48E1s and less Block RAM at the same frame size.

Table IX shows the comparisons of ASIC implementations. Our design can easily support HD1080p30 with 2000 feature points at 100MHz and saves 56% gate count and 90.4% memory cost when compared to the previous design for the VGA image. This high throughput and lower cost is due to the layer parallel scheme and related hardware implementation techniques.

Fig. 16 illustrates the prototype of the realtime feature extraction system based on the FPGA implementation. The system captured VGA-sized live video from a Panasonic CCD camera, extracted features, and displayed the captured video and extracted features on a screen in realtime. The features were drawn as small red crosses on the screen. We have

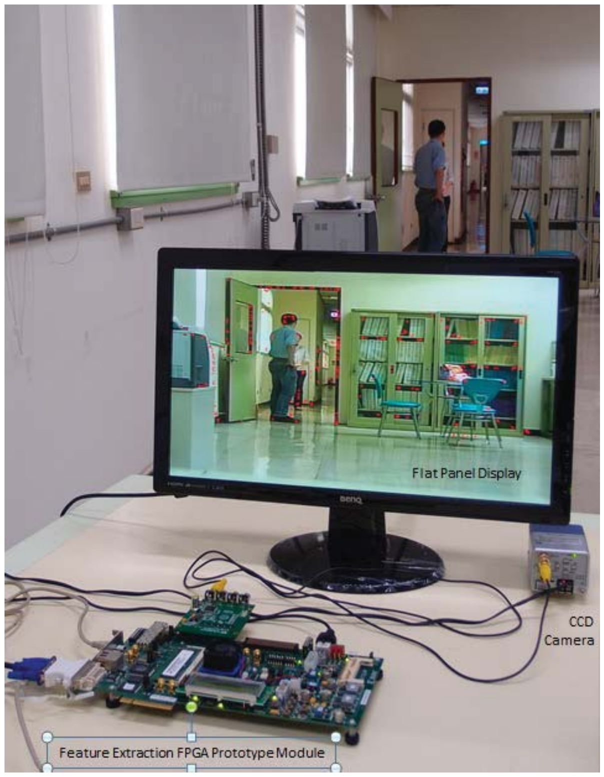


Fig. 16. Prototype of the FPGA-based real-time feature extraction system.

TABLE VIII
COMPARISON OF FPGA IMPLEMENTATIONS

	[11]	[15]	Proposed
Frame size	QVGA	VGA	VGA/ HD1080p
FPGA	Altera Stratix II	Xilinx V-5 ML507	Xilinx V-6 ML605
LUT	43366	35889	57598
Registers	19100	19529	24988
Block RAM (Kbits)	1350	3240	1206/2286
Others	64 DSP blocks	97 DSP48E	8DSP48E1

TABLE IX
COMPARISON OF ASIC IMPLEMENTATIONS

	[17]	Proposed	Proposed
Technology	180 nm	90 nm	90 nm
Frequency	100 MHz	100 MHz	227 MHz
Frame size	VGA30fps	HD1080p30	HD1080p30
Feature point/frame	890	6000 (VGA) 2000 (HD1080)	11000 (HD1080)
Gate count	1320 K	580 K	704 K
Memory	5.729 Mb	553 Kbit	553 Kbit

verified the features extracted from this prototype with the result of our feature extraction software. A video recording of the prototype system running feature extraction can be found on Youtube (http://youtu.be/NDDBkVQugp0).

VI. CONCLUSION

This paper presents a real time SIFT based feature extraction engine that is capable to compute 2000 feature points for HD1080p30 at 100 MHz. The proposed design adopts the layer parallel restructured box kernel to replace iterated Gaussian blur operations for simple and parallel computation. This also reduces the latency to a few image lines instead of several frames. The final flow uses the on-the-fly feature extraction flow so that only partial intermediate results have to be stored. With these techniques, the presented design easily achieves the real time demand with significantly lower cost, which saves 56% gate count and 90.4% memory cost when compared to the previous design.

REFERENCES

[1] D. G. Lowe, "Distinctive image feautres from scale-invariant keypoints," *Int. J. Comput. Vis.*, vol. 60, no. 2, pp. 91–110, 2004.

[2] H. Bay, T. Tuytelaars, and L. J. V. Gool, "SURF: Speeded up robust features," in *Proc. Eur. Conf. Comput. Vis.*, 2008, pp. 404–417.

[3] Q. Zhang, Y. Chen, Y. Zhang, and Y. Xu, "SIFT implementation and optimization for multi-core systems," in *Proc. IEEE Int. Parallel Distrib. Process.*, Apr. 2008, pp. 1–8.

[4] H. Feng, E. Li, Y. Chen, and Y. Zhang, "Parallelization and characterization of SIFT on multi-core systems," in *Proc. IEEE Int. Symp. Workload Characterizat.*, Sep. 2008, pp. 14–23.

[5] S. N. Sinha, J.-M. Frahm, M. Pollefeys, and Y. Genc, "GPU-based video feature tracking and matching," in *Proc. Workshop Edge Comput. Using New Commod. Archit.*, 2006, pp. 1–15.

[6] S. Heymann, K. Müller, A. Smolic, B. Froehlich, and T. Wiegand, "SIFT implementation and optimization for general-purpose GPU," in *Proc. Int. Conf. Central Eur. Comput. Graph., Visualizat. Comput. Vis.*, 2007, pp. 144–159.

[7] N. Cornelis and L. Van Gool, "Fast scale invariant feature detection and matching on programmable graphics hardware," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit. Workshops*, Jun. 2008, pp. 1–8.

[8] T. Terriberry, L. French, and J. Helmsen, "GPU accelerating speeded-up robust features," in *Proc. 4th Int. Symp. 3D Data Process., Visualizat. Transmiss.*, 2008, pp. 1–8.

[9] D. Kim, K. Kim, J. Y. Kim, S. Lee, S. J. Lee, H. J. Yoo, "81.6 GOPS object recognition processor based on a memory-centric NoC," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 17, no. 3, pp. 370–382, Mar. 2009.

[10] D. Kim, K. Kim, J. Y. Kim, S. Lee, S. J. Lee, and H. J. Yoo, "Vision platfrom for mobile intelligent robot based on 81.6 GOPS object recognition processor," in *Proc. 45th ACM/IEEE Design Autom. Conf.*, Jun. 2008, pp. 96–101.

[11] V. Bonato, E. Marques, and G. A. Constantinides, "A parallel hardware architecture for scale and rotation invariant feature detection," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 18, no. 12, pp. 1703–1712, Dec. 2008.

[12] N. Sawasaki, M. Nakao, Y. Yamato, and K. Okabayahi, "Embedded vision system for mobile robot navigation," in *Proc. Int. Conf. Robot Autom.*, 2006, pp. 2693–2698.

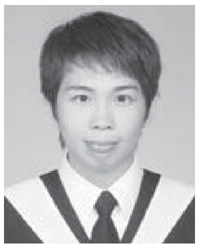
[13] M. Grabner, H. Grabner, and H. Bischof, "Fast approximated SIFT," in *Proc. Asian Conf. Comput. Vis.*, 2006, pp. 918–927.

[14] K. G. Derpanis, E. T. Leung, and M. Sizintsev, "Fast scale-space feature representations by generalized integral images," in *Proc. IEEE Int. Conf. Image Process.*, Sep.–Oct. 2007, pp. 521–524.

[15] L. Yao, H. Feng, Y. Zhu, Z. Jiang, D. Zhao, and W. Feng, "An architecture of optimised SIFT feature detection for an FPGA implementation of an image matcher," in *Proc. Int. Conf. Field-Program. Technol.*, 2009, pp. 30–37.

[16] P. H. Hsu, Y. C. Tseng, and T. S. Chang, " Low memory cost bilateral filtering using stripe-based sliding integral histogram," in *Proc. IEEE Int. Symp. Circuits Syst.*, May–Jun. 2010, pp. 3120–3123.

[17] F. C. Huang, S. Y. Huang, J. W. Ker, and Y. C. Chen, "High-performance SIFT hardware accelerator for real-Time image feature extraction," *IEEE Trans. Circuits Syst. Video Technol.*, vol. 22, no. 3, pp. 340–351, Mar. 2012.



Liang-Chi Chiu received the B.S. and M.S. degrees in electronic engineering from National Chiao-Tung University (NCTU), Hsinchu, Taiwan, in 2009 and 2011, respectively. He is currently an Engineer with PixelArt Technology, Hsinchu. His current research interests include system-on-a-chip design, VLSI signal processing, and computer architecture.

Tian-Sheuan Chang (S'93–M'06–SM'07) received the B.S., M.S., and Ph.D. degrees in electronic engineering from National Chiao-Tung University (NCTU), Hsinchu, Taiwan, in 1993, 1995, and 1999, respectively.

He was a Deputy Manager with Global Unichip Corporation, Hsinchu, from 2000 to 2004. In 2004, he joined the Department of Electronics Engineering, NCTU, where he is currently a Professor. In 2009, he was a Visiting Scholar in IMEC, Heverlee, Belgium. His current research interests include system-on-a-chip design, VLSI signal processing, and computer architecture.

Dr. Chang was a recipient of the Excellent Young Electrical Engineer from Chinese Institute of Electrical Engineering in 2007, and the Outstanding Young Scholar from Taiwan IC Design Society in 2010. He has been actively involved in many international conferences as an organizing committee or technical program committee member. He is currently an Editorial Board Member of the IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS FOR VIDEO TECHNOLOGY.

Jiun-Yen Chen received the M.S. degree in electrical engineering from the National Yunlin University of Science and Technology, Yunlin, Taiwan, in 2006. He was an Associate Engineer with the Mechanical and Systems Research Laboratories (MSL), Industrial Technology Research Institute (ITRI), Hsinchu, Taiwan, from 2010 to 2013.

Nelson Yen-Chung Chang (S'06) received the B.S. degree in electrical engineering from National Tsing-Hua University, Hsinchu, Taiwan, in 2000, and the M.S. and Ph.D. degrees in electronic engineering from National Chiao-Tung University, Hsinchu, in 2002 and 2009, respectively.

He is currently an R&D Manager with the Division of Intelligent Robotics Technology, Mechanical and Systems Research Laboratories, Industrial Technology Research Institute, Hsinchu, Taiwan. His current research interests include real-time vision system, robot vision, advanced robotic embedded systems, and vision algorithms.