SIFT-Based Low Complexity Keypoint Extraction and Its Real-Time Hardware Implementation for Full-HD Video

Takahiro Suzuki and Takeshi Ikenaga
Waseda University, Tokyo, Japan
E-mail: takahir0@toki.waseda.jp Tel/Fax: +81-3-5286-2350

***Abstract*—Scale-Invariant Feature Transform (SIFT) has lately attracted attention in computer vision as a robust keypoint detection algorithm which is invariant for scale, rotation and illumination change. However, its computational complexity is too high to apply practical real-time applications. This paper proposes a low complexity keypoint extraction algorithm based on SIFT descriptor and utilization of the database, and its real-time hardware implementation for Full-HD resolution video. The proposed algorithm computes SIFT descriptor on the keypoint obtained by corner detection and selects a scale from the database. It is possible to parallelize the keypoint detection and descriptor computation modules in the hardware. These modules do not depend on each other in the proposed algorithm in contrast with SIFT that computes a scale. The processing time of descriptor computation in this hardware is independent of the number of keypoints because its descriptor generation is pipelining structure of pixel. Evaluation results show that the proposed algorithm on software is 12 times faster than SIFT. Moreover, the proposed hardware on FPGA is 427 times faster than SIFT and 61 times faster than the proposed algorithm on software. The proposed hardware performs keypoint extraction and matching at 60 fps for Full-HD video.**

I. Introduction

Recently, Scale-Invariant Feature Transform (SIFT) [1] has attracted attention in computer vision because of its robustness in keypoint detection. Since SIFT can describe scale, rotation and illumination invariant features from images, matching between distinct images is executed accurately. By fully utilizing this characteristics, wide range of application is being considered. For example, it is used for object recognition [2], human or other object tracking [3], [4], recognizing panorama [5], 3-D reconstruction [6].

However, due to its complicated structure, there is a problem that conventional SIFT requires high computational complexity. Especially, SIFT's Difference of Gaussian (DoG) process is high complexity because an input image is repeatedly convolved with Gaussian filter. It is difficult to perform in real-time on software by even Speeded-Up Robust Features (SURF) [7] and approximated SIFT [8] which SIFT is speeded-uped. There are several methods that improve the matching performance, for example GLOH [9], PCA-SIFT [10], CSIFT [11] and ASIFT [12], but many calculations are added to SIFT in these methods. Moreover, many serial parts which SIFT contains make it difficult to accomplish hardware-implementation. It leads to hardware expansion without parallelizing and pipelining. Recently, parallelized hardwares of SIFT [13], [14] have been proposed. However, their target is VGA video and there is few real-time hardware for Full-HD video up to the present.

This paper proposes low complexity keypoint extraction algorithm based on SIFT descriptor and utilization of the database on the software for VGA resolution video, and its real-time hardware implementation for Full-HD resolution video to accomplish real-time processing. First, the complexity of SIFT is reduced because hardware implementation of conventional SIFT requires a lot of hardware resources. The detection method is replaced by the corner detection and SIFT descriptor is generated in each keypoint. The utilization of database make it deals with scale-change. Next, it parallelizes the serial processing part such as computing histogram to reduces clocks to take for computation and construct keypoint pipeline architecture for matching processing to reduce hardware resources largely. In this way, realistic scale hardware with high performance is designed.

II. SIFT

SIFT is an algorithm which describes scale, rotation and illumination invariant keypoints from images. The algorithm is divided into following two key parts.

- Keypoint detection by the DoG
- SIFT descriptor computation

The keypoint detection is the process which decides keypoint's position near characterized region. The SIFT descriptor computation makes the histograms with information about neighboring region. These are primary processes of a keypoint extraction. 2nd section shows the details of processes and problems.

A. Keypoint Detection by the DoG

SIFT detects scale invariant keypoints by the DoG function. DoG function computes difference of images convolved by Gaussian filters. An image, $I(x, y)$, a variable-scale Gauss function, $G(x, y, \sigma)$, and a smoothed images, $L(x, y, \sigma)$, define the DoG image, $D(x, y, \sigma)$:

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma), \end{aligned} \quad (1)$$

Fig. 1. The DoG detector.

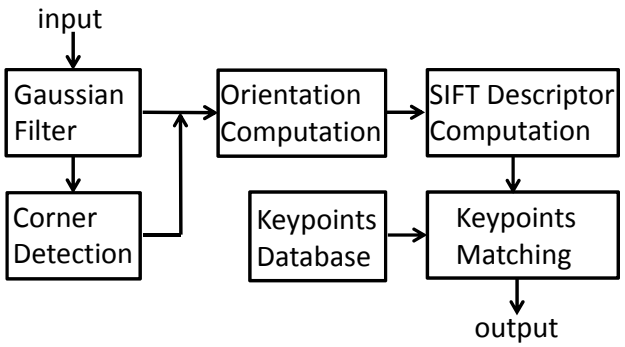

Fig. 2. Flowchart of the proposed algorithm.

where $*$ is the convolution operation. $D(x, y, \sigma)$ is repeatedly computed by a constant multiplicative factor k. Computational complexity becomes higher and higher when σ increases. Fig.1 is schema of the DoG detector. Thus, this process is very complex. After this, detections of extreme value and localizations of keypoints are performed. It also high computational complexity because localizations use matrix calculation.

B. SIFT descriptor computation

In computation of SIFT descriptor, firstly, keypoint's orientation is obtained. The histogram is calculated by gradient magnitude $m(x, y)$ and orientation $\theta(x, y)$:

$$m(x,y) = \sqrt{L_x(x,y) + L_y(x,y)}, \quad (2)$$

$$\theta(x,y) = tan^{-1}\frac{L_y(x,y)}{L_x(x,y)}. \quad (3)$$

When its sum of magnitude is max, the orientation becomes the keypoint's one. After this, SIFT descriptor is computed. The region is rotated by the keypoint's orientation. The size of region depends on scale obtained by the DoG detector. It is divided into 4×4 and histogram is computed by 8 directions in each region. Total 128 dimension vector, SIFT descriptor, is generated. This process's computational complexity changes depending on keypoint's scale.

III. SIFT-BASED LOW COMPLEXITY KEYPOINT EXTRACTION

In section III, we show the method which reduces computational complexity of SIFT. This paper mainly proposes two methods to accomplish the real-time processing.

- The approximation of Harris detector and using the integral image
- Utilization of multi-scale images in the database

The flowchart which summarizes process of this algorithm is shown in Fig.2. The DoG detector is the highest computational complexity part in SIFT algorithm as shown in section II. However, the keypoints obtained by DoG is positioned near corners in an image. Therefor, we propose that the DoG is replaced with corner detection. The computational complexity is drastically reduced by this because corner detection is relatively low complexity. The gaussian filter in Fig.2 is used for the noise reduction. When SIFT descriptor is computed, the size of described regions is a constant 15×15. This also simplifies SIFT, because the regions increase in size depending on scale in the case of SIFT.

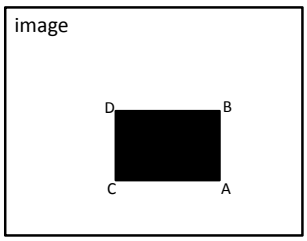

Fig. 3. How to use an integral image.

A. The approximation of Harris detector and using the integral image

Harris detector [15] is one of the corner detection methods. Its positioning corner is very suitable. It uses filer which computes 2nd-order difference of adjacent pixels. It need to refer many adjacent pixels during detection from general images with nose. According to the number of refereed pixels, the process time becomes very long. Thus, an integral image and box filter are utilized for speeding up.

First, an integral image is explained. The integral image, $I_{integral}(x, y)$, is the sum of pixels from top left corner of image to intended pixel (x,y):

$$I_{integral}(x,y) = \sum_{i=0}^{i \le x} \sum_{j=0}^{j \le y} I(i,j). \quad (4)$$

For example, the case of Fig. 3 is considered. At this time, sum of the rectangular region, S:

$$S = A - B - C + D. \quad (5)$$

This method speed up the calculation of rectangular region's sum. Moreover, there is a merit that the process time does not depend on the size of region. It has many merits during software processing. However, in the case of hardware, it is difficult to reserve memory because integral image use a lot of memory. Thus, it does not use integral image.

Next, corner detection by box filter is shown. Harris's method computes Hessian matrix, $\mathbf{H}$, which is composed of

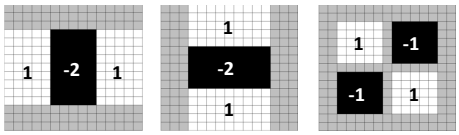

Fig. 4. Approximated Filter (L_{xx}, L_{yy}, L_{xy}).

elements are the 2nd-order difference of adjacent pixels:

$$\mathbf{H} = \mathbf{G}(\sigma) \begin{bmatrix} L_{xx} & L_{xy} \\ L_{xy} & L_{yy} \end{bmatrix}. \quad (6)$$

In general, their elements is weighed by Gaussian function, $\mathbf{G}(\sigma)$. However, it is not suitable for an integral image because weighing is detail. Thus, this filter is approximated and it becomes easy to compute by integral image. Approximated filter is shown in Fig. 4. L_{xx}, L_{yy}, L_{xy} is obtained by filter process of integral image. After that, they are used to compute function which decides corners. When the position is a corner, it satisfies the equation,

$$det(\mathbf{H}) - \omega tra(\mathbf{H}) > T, \quad (7)$$

where ω is a parameter and T is a threshold. If the threshold becomes larger, corners decreases and becomes better position as corners. The threshold is determined experimentally to obtain the appropriate number of keypoints.

B. Utilization of multi-scale images in the database

This proposed algorithm removed the DoG detector of SIFT. In other words, it does not compute scale of each keypoint. It is impossible to deal with scale changes as it is. Thus, next, we propose the solution that prepares various size images in the database and decides the scale during keypoint matching. SIFT descriptor can deal with some scale-changes because it is very robust for various image changes or transformations. Considering this feature, we prepare three images of various size at regular intervals. For example, images of three sizes (1, 1.5, 0.5) is registered as objects of matching. Keypoints are extracted from them and tag is added to keypoints. The tag (=1, 1.5, 0.5) shows which size image the keypoint is obtained from. In keypoint matching process, tag is checked and counted if registered keypoint match with input image's kepoints. The tag with the most matches is considered as nearly input image's scale. Finally, input image matched with the registered image of decided tag. Keypoint matching process is speeded up by Approximated Nearest Neighbor (ANN) [16] in comparison with Nearest Neighbor (NN). NN is a computation of distance between two feature vectors. In the case of NN, it searches all keypoints, but the process can be avoided by approximation of ANN. Software processing uses ANN, but it uses NN to simplify the hardware structure in the case of hardware. Fig. 5 is schema of this process.

IV. HARDWARE IMPLEMENTATION OF PROPOSED KEYPOINT EXTRACTION ALGORITHM

This section describe the hardware architecture using pipelining and the method that parallelizes the complex processes in the proposed algorithm. If real-time keypoint extraction hardware for Full-HD is hardware-implemented simply, hardware resources become too large to be loaded in realistic scale FPGAs. This keypoint extraction algorithm has the following problems to implement on FPGAs. The proposed solutions are also shown.

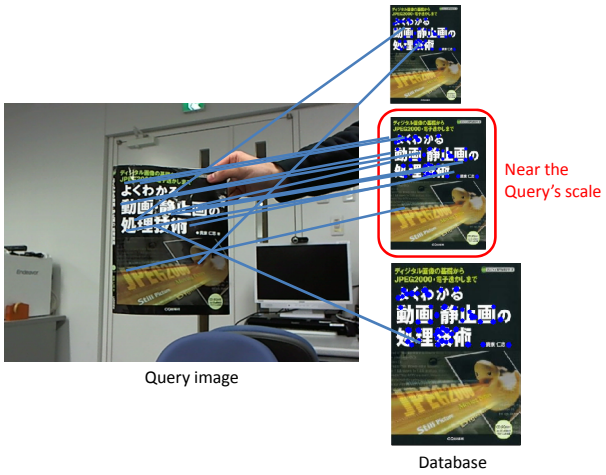

Fig. 5. Proposed matching process.

1) Complexity calculations
   ⇒ (A) Pipelining by fetching pixel lines from block RAM
   ⇒ (B) Approximation based on SAD, bit-shift and LUT
2) Increasing processing clock are accompanied by increasing register
   ⇒ (C) Parallelization of the detection module and the descriptor module
3) Serial processing of the histogram computation
   ⇒ (D) Parallelization of descriptor computation
4) Processing large amount of keypoins data
   ⇒ (E) Matching process by keypoint pipelining

These methods are shown in following section in more detail. First, The entire structure and flow are shown.

The entire block diagram is expressed in Fig.6. The inputted RGB data is entered into the keypoint extraction modules. In the keypoint extraction module (Fig.7), 1 descriptor is outputted for 1 RGB data. The RGB data is converted to the gray scale data. It is accumulated in 5×5 and smoothed by gaussian coefficients. Next, these are entered into line buffer and vertical 15 pixel is kept. These are inputted to the detection module and the descriptor module in parallel. In the detection module, 15×15 pixel data is kept and weighted by the filter in Fig.4. In the descriptor module, after the orientation computation (9×9 region), the SIFT descriptor (8 bit×128 D=1024 bit) is generated in 15×15 region. The descriptor is generated in all pixels because the structure is a pixel pipelining. If the pixel is detected as a keypoint (the signal of is_keypoint is 1), the position and descriptor data are memorized in block RAM.

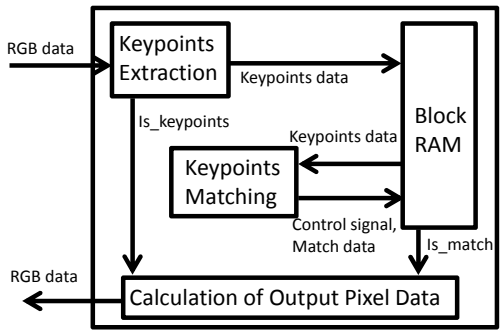

Fig. 6. The block diagram of entire structure.

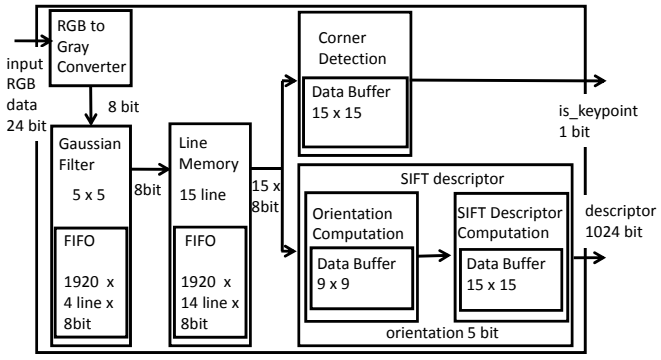

Fig. 7. The structure of keypoint extraction module.

Keypoint matching process is performed with 1 frame delay in keypoint matching module (Fig.9). The descriptor data in the block RAM are fetched and SAD is computed. It is a distance of feature vector. The minimum data between 1 inputted keypoint and all database keypoints is kept and sent to block RAM. In all clock, the address of block RAM is computed. Finally, an output image is generated by using the kepoint data and matching results.

A. Pipelining by fetching pixel lines from block RAM

To calculate SIFT descriptor in each pixel, it is necessary to process 9×9 region for orientation and 15×15 region for descriptor in 1 clock. It requires 9×9+15×15=306 calculators if it is computed simultaneously. To reduce the hardware resources , pipelining using 15 pixel lines is proposed. It deals with problem 1). When the vertical 15 pixel is inputed, these are calculated and m, θ are generated. After that, obtained values are kept in 14 clocks by registers. Only 9+15=24 calculator is required by this pipelining. This architecture is shown in the line buffer of Fig.7.

B. Approximation based on SAD, bit-shift and LUT

To this point, the proposed low complexity algorithm remains relatively complex set of calculations. This consumes a large amount of hardware resources. Approximated calculator reduces the use of resources and enhances frequency of circuits. This also deals with problem 1).

First, equation (2) incorporates a square root. To solve the square root, many methods are proposed, such as extraction of square root. However, these methods use iterations, which may generate hardware delays because its number is unknown. Thus, the magnitude of gradient is considered. It is replaced by Sum of Absolute Difference (SAD) because the magnitude is a weight of histogram computation. By this, magnitude of gradient is redefine as

$$m(x,y) \quad = \quad |L_x(x,y)| + |L_y(x,y)|. \tag{8}$$

It is calculated more simply. The distance of two descriptors is computed similarly during matching process.

Next, the direction of gradient in equation (3) is computed approximately. It includes division and arctan. It occurs in two steps. First, the division is approximated by bit shift. Concretely, numerator is shifted by the value of denominator. Second, the arctan is computed by the Look Up Table (LUT). the value of arctan is decided by each result of the division. The direction of magnitude is quantized in 32 directions.

These approximations do not cause large precision losses. It is considered that these processes do not need high precision calculation because they includes quantization.

C. Parallelization of detection module and descriptor module

SIFT has the dependency between keypoint detection and computation of descriptor because a scale has to be computed and it determines the descriptor region. However, the proposed algorithm in section *III* does not compute a scale. Therefore, it becomes possible to parallelize keypoint detection and computation of descriptor. It reduces clocks to compute and solve the problem 2). This structure is shown in Fig.7. 15 pixel data is inputted from line memory. After that, detection and description are computed simultaneously. After the results are obtained, they are synchronized.

D. Parallelization of descriptor computation

The SIFT descriptor is computed by histogram computation. In general, it is serial processing because a conflict between registers occurs. It takes many clocks to generate descriptor. Concretely, 225 clocks is required because it uses 15×15 region. To solve this, SIFT descriptor computation is parallelized. SIFT descriptor divides the region into 4×4 grids. The bins of histogram do not have a dependence on each other. Thus, it is possible to parallelize distinct grid. The simulation result shows that 9 pixels at regular intervals do not depend on each other. It computes 15×15 keypoint region by placing 25 descriptor registers. After registers have values, they are added together in each bins. It computes descriptor in 4 clocks with finality. It deals with problem 3). Fig.8 is a schema that shows this processing.

E. Matching process by keypoint pipelining

Each keypoint has the descriptor data (1024 bit) and the position data (22 bit). It is very long bit data. If plural keypoints is processed simultaneously, a lot of hardware resources are required. To deal with problem 4), 1 keypoint data is fetched

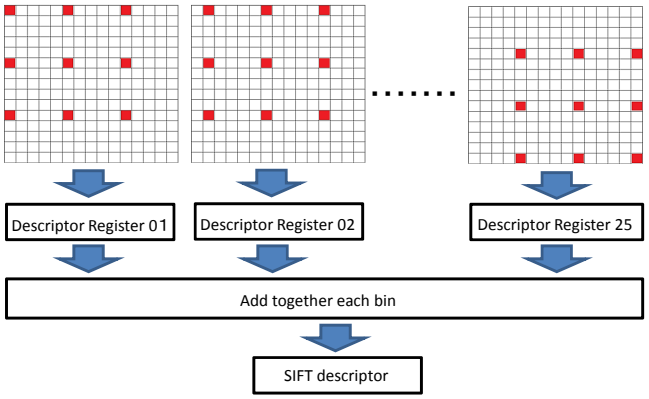

Fig. 8. Parallelization of histogram computation.

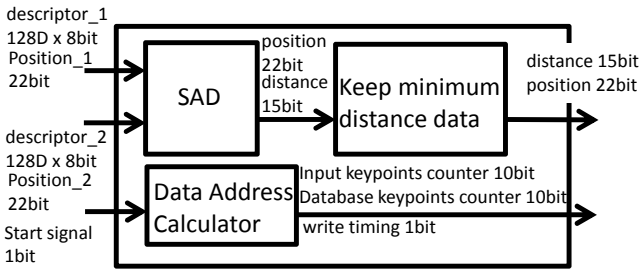

Fig. 9. The structure of keypoint matching module.

from block RAM in 1 clock and entered into the matching module of pipelining structure. The nearest neighbor search is performed. The obtained keypoint pairs are memorized in block RAM. The structure is shown in Fig.9.

V. EXPERIMENTAL RESULTS

The proposed algorithm is compared with SIFT with respect to performance and speed. The development environment on software is Visual Studio C++ 2008. CPU is Intel Core i5 CPU M 450 2.40GHz. Vertex-5 (XC5VLX330-1FF1760C) as FPGA offered by Xilinx, Inc. is used for hardware evaluation. The logic synthesis on FPGA is performed by ISE11.4.

First, both methods are examined with accuracy when the object has a scale-changes. In this paper, matching accuracy (ACC) is evaluated by

$$\mathrm{ACC} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}, \tag{9}$$

where True Positives (TP) is the number of correct matches and False Negatives (FN) is the number of incorrect matches. The true vales of correct matches is decided by RANdom SAmple Consensus (RANSAC) [17]. Matches which is satisfied with the equation,

$$|\mathbf{x_e} - \mathbf{x_r}| < \mathbf{Error}, \tag{10}$$

are counted to compute TP. TP+FM is the number of all matches. It is a constant Error=10 in this case. The experimental result is Fig.10. This result shows the proposed algorithm is almost same ACC with SIFT.

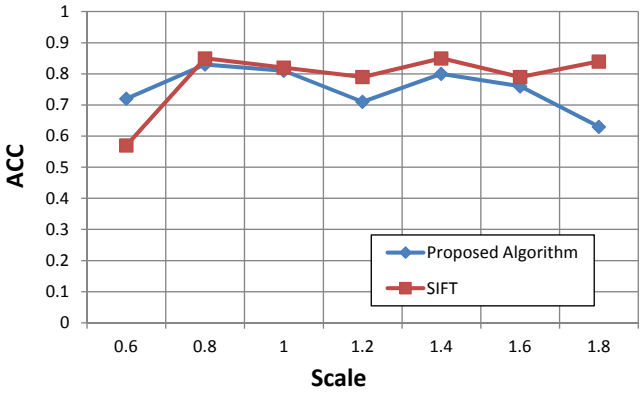

Fig. 10. The accuracy during scale-change.

Next, the proposed hardware on the FPGA and the proposed algorithm on software are compared with SIFT. TABLE I is a result comparing processing time per frame between SIFT and the proposals. The result comparing Proposal (SW/VGA) with SIFT (SW/VGA) shows that the proposed algorithm is about 12 times faster than SIFT on software. It performs 10 fps keypoint extraction and matching for VGA video. The result comparing Proposal (HW/Full-HD) with SIFT (SW/Full-HD), Proposal (SW/Full-HD) shows that the proposed hardware on FPGA is about 427 times faster than SIFT and about 61 times faster than the proposed algorithm on software. The proposed hardware performs at 60 fps keypoint extraction and matching for Full-HD video. The number of keypoints is also shown, but the processing time of keypoint extraction does not depend on the number of keypoints due to pixel pipelining. On the other the hand, keypoint matching depends on the number of keypoints due to keypoint pipelining. We have verified that it is possible to perform keypoint matching up to 1024 keypoints on the FPGA.

The total FPGA resource utilization and maximum frequency of the proposed hardware is shown in TABLE II. In the table, the available resources of the used FPGA is also described. The utilization shows that the proposed hardware can be implemented in Virtex-5. Especially, the number of DSP48Es is much low because the approximation and the use of LUTs are very effective for hardware reduction.

Fig.11 is the software simulations of keypoint matching. It deals with scale-changes in high accuracy. Fig.12 is a demonstration of keypoint extraction hardware. In this case, we uses Virtex-6 to draw output images. It matches input image with right bottom parts which keypoints are registered in advance. It is drawing lines that express matches between an input image and a registered image.

VI. CONCLUSIONS

This paper proposed the low complexity keypoint extraction algorithm for VGA and its hardware architecture for Full-HD video. First, this paper reduces the computational com-

TABLE I
COMPARISON OF PROCESSING TIME (SW OR HW/VGA OR FULL-HD)

			processing time [ms]	number of keypoints
SIFT	SW	VGA	1026	252
Proposal			**85**	258
SIFT		Full -HD	6840	986
Proposal			982	950
Proposal	HW		**16**	1024

TABLE II
TOTAL FPGA RESOURCE UTILIZATION AND MAXIMUM FREQUENCY

FPGA Resource	Used	Available	Utilization
Number of Slice Registers	55,407	207,360	26%
Number of Slice LUTs	108,322	207,360	52%
Number of occupied Slices	32,741	51,840	63%
Number of BlockRAM/FIFO	89	288	30%
Number of DSP48Es	3	192	1%
Maximum Frequency	168.464MHz		

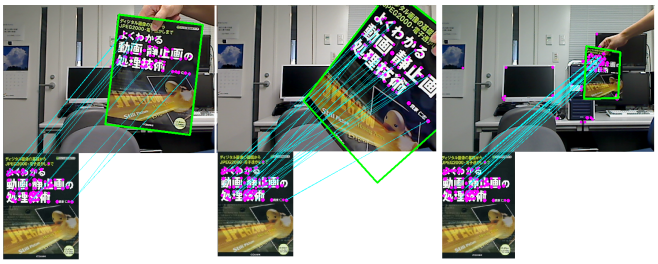

Fig. 11. The software simulation of keypoint matching.

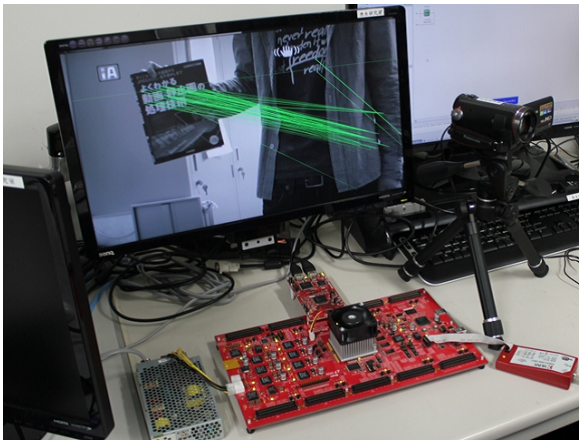

Fig. 12. The demonstration of keypoint extraction hardware.

plexity of SIFT by the corner detection, SIFT descriptor and utilizing the database. After that, this proposed algorithm is implemented on FPGA. The proposed hardware uses pixel pipelining, parallelization of detection and descriptor module, parallelization of the descriptor computation to accomplish low resources and high performances.

The software simulation shows that the proposed algorithm is about 12 times faster than SIFT maintaining almost same accuracy. The comparison of the proposed hardware, the proposed software algorithm and the SIFT shows that the proposed hardware is about 427 times faster than SIFT and about 61 times faster than the proposed algorithm on software. This hardware performs 60 fps keypoint extraction and matching for Full-HD video. Moreover, the processing time of keypoint extraction does not depend on the number of keypoints. It becomes possible to apply this keypoint extraction hardware to real-time application which has been widely proposed.

ACKNOWLEDGMENT

This work was supported by KAKENHI (23300018).

REFERENCES

[1] D. G. Lowe, “Distinctive image features from scale-invariant keypoints”, Int.Journal of Computer Vision, 60, pp. 91-110, 2004.
[2] D. G. Lowe,”Object recognition from local scale-invariant features”, In International Conference on Computer Vision, Corfu, Greece, pp. 1150-1157, 1999.
[3] Yuji Tsuzuki, Hironobu Fujiyoshi, Takeo Kanade,”Mean Shift-based Point Feature Tracking using SIFT”, Journal of Information Processing Society ,Vol. 49, No. SIG 6, pp. 35-45, 2008.
[4] Huiyu Zhou , Yuan Yuan , Chunmei Shi, ”Object tracking using SIFT features and mean shift”, Computer Vision and Image Understanding, v.113 n.3, pp. 345-352, March, 2009.
[5] Matthew Brown and David G. Lowe, ”Recognising panoramas”, International Conference on Computer Vision , pp. 1218-25, 2003.
[6] S. Agarwal, N. Snavely, I. Simon, S. M. Seitz, R. Szeliski, ”Building rome in a day”, In ICCV, 2009.
[7] H.Bay, T.Tuytelaars and L. V. Gool, “SURF:speeded up robust features”, In ECCV, pp. 404-417, 2006.
[8] G. Michael, G. Helmut, B. Horst, “ Fast approximated SIFT ” , Proc. of ACCV, pp. 918-927, 2006.
[9] Krystian Mikolajczyk, Cordelia Schmid, ”A performance evaluation of local descriptors”, IEEE Transactions on Pattern Analysis and Machine Intelligence, 10, 27, pp. 1615–1630, 2005.
[10] Y. Ke, R. Sukthankar. PCA-SIFT, ”A More Distinctive Representation for Local Image Descriptors”, Proceedings of Computer Vision and Pattern Recognition, 2004.
[11] A.E. Abdel-Hakim, A.A. Farag,”Csift: a sift descriptor with color invariant characteristics”, Proceedings of the Computer Vision and Pattern Recognition, pp. 1978–1983, 2006.
[12] J. M. Morel and G. Yu, “ Asift: A new framework for fully affine invariant image comparison ” , SIAM Journal on Imaging Sciences, 2, 2, pp. 438-469, 2009.
[13] V. Bonato , E. Marques, G. A. Constantinides ,”A parallel hardware architecture for scale and rotation invariant feature detection”, IEEE Trans. Circuits Syst. Video Technol., vol. 18, pp. 1703 -1712, 2008.
[14] J. Qiu, T. Huang, and T. Ikenaga, “A FPGA-based dual-pixel processing pipelined hardware accelerator for feature point detection part in SIFT, ” in Proc. 5th Int. Joint Conf. INC IMS IDC, pp. 1668-1674, 2009.
[15] C. Harris and M.J. Stephens,”A combined corner and edge detector”, InAlvey Vision Conference, pp. 147-152, 1988.
[16] S. Arya, D. M. Mount, N. S. Netanyahu, R. Silverman and A. Wu, ”An optimal algorithm for approximate nearest neighbor searching”, Journal of the ACM, 45, 6, pp. 891-923, 1998.
[17] M.A. Fischler, R.C. Bolles,”Random sample consensus: a paradigm for model fitting with application to image analysis and automated cartography”, Commun. Assoc. Comp. Mach., pp. 24-95, 1981.