electronics

MDPI

Article

Fast Execution of an ASIFT Hardware Accelerator by Prior Data Processing

Joohyuk Yum [1], Jin-Sung Kim [2,*] and Hyuk-Jae Lee [3]

[1] System LSI Division, Samsung Electronics Corporation, Hwaseong 18448, Korea; joohyuk86@capp.snu.ac.kr
[2] Department of Electronic Engineering, Sun Moon University, Asan 31460, Korea
[3] Department of Electrical and Computer Engineering, Seoul National University, Seoul 08826, Korea; hyuk_jae_lee@capp.snu.ac.kr
* Correspondence: jinsungk@sunmoon.ac.kr

Received: 29 September 2019; Accepted: 15 October 2019; Published: 17 October 2019

Abstract: This paper proposes a new ASIFT hardware architecture that processes a Video Graphics Array (VGA)-sized (640 × 480) video in real time. The previous ASIFT accelerator suffers from low utilization because affine transformed images are computed repeatedly. In order to improve hardware utilization, the proposed hardware architecture adopts two schemes to increase the utilization of a bottleneck hardware module. The first is a prior anti-aliasing scheme, and the second is a prior down-scaling scheme. In the proposed method, 1 × 1 and 0.5 × 1 blurred images are generated and they are reused for creating various affine transformed images. Thanks to the proposed schemes, the utilization drop by waiting for the affine transform is significantly decreased, and consequently, the operation speed is increased substantially. Experimental results show that the proposed ASIFT hardware accelerator processes a VGA-sized video at the speed of 28 frames/s, which is 1.36 times faster than that of previous work.

Keywords: affine transform; affine-invariant extension of SIFT(ASIFT); hardware accelerator

1. Introduction

Local features have been widely used for scene matching, which is important in many computer vision applications such as object detection, tracking, and motion estimation. For robust scene matching, scale-invariant feature transform (SIFT) proposed by Lowe [1] has been used as one of the most reliable local features because translation, rotation, and scale invariances are effectively supported. Unfortunately, the performance of SIFT is degraded when the direction of a camera view is changed. To overcome this limitation, Morel et al. proposed an affine invariant extension of SIFT (ASIFT) [2].

Since a large amount of computation is required in a SIFT algorithm, optimized hardware accelerators for SIFT have been proposed [3–7]. The ASIFT algorithm generates many images transformed by affine transforms in order to simulate the view change of a camera. Then, SIFT features are extracted in the simulated images. This means that the computational complexity of an ASIFT algorithm is much higher than that of a SIFT algorithm. In order to increase the processing speed of a complex ASIFT algorithm, Yum et al. [8] proposed an ASIFT hardware architecture that adopts a modified affine transform to reduce the latency of an external memory, and consequently, the operation speed of an ASIFT algorithm increases significantly. Nonetheless, this hardware accelerator processes a VGA-sized (640 × 480) video sequence at 20 frames/s (fps), which is not fast enough for real-time processing.

In order to increase the operation speed of an ASIFT hardware implementation, this paper proposes two schemes that increase the utilization of an affine transform module, which is a bottleneck of the hardware accelerator [8]. The first is a *prior anti-aliasing scheme* that computes a 1 × 1 blurred image

and stores it in an external memory. By reusing the stored image for generating various simulated images, redundant data fetching for generating the 1×1 blurred image is removed. The second is a *prior down-scaling scheme*. A 0.5×1 blurred image is generated and reused for generating the simulated images of which the width is scaled less than 0.5 times. A word of the 0.5×1 blurred image includes more valid pixels than that of the 1×1 blurred image. Thus, the stall cycles to wait for valid data are decreased. As a result, the proposed ASIFT hardware implementation processes a VGA-sized video at 28 fps.

2. Previous Work

2.1. ASIFT Algorithm

An ASIFT algorithm is proposed to achieve full affine invariance such that it can find correspondences in two images representing the same scene even though they are obtained from any viewpoints [2]. In an ASIFT algorithm, simulated images for various camera viewpoints are generated by transforming a source image with affine transform matrices. Then, SIFT features are computed in the simulated images. Because these SIFT features are obtained by considering the viewpoint change, correspondences can be found between two images for which the camera viewpoints are different.

The images captured by a camera at various positions can be interpreted as affine decomposition. The camera position is represented on hemispherical coordinates as shown in Figure 1. The center (o) of the hemisphere is located at the center of a source image u. The latitude and longitude of the position of the camera are represented by θ and φ, respectively. The affine distortion caused by the change of the camera position is interpreted as the rotation and scaling of an image. The affine transform is represented by Equation (1). In this equation, image rotation and scaling are represented by a rotation matrix (R_φ) and a scaling matrix ($T_{1,1/t}$), respectively.

$$A = T_{1,1/t} R_\varphi = \begin{bmatrix} 1 & 0 \\ 0 & 1/t \end{bmatrix} \begin{bmatrix} cos]\varphi & -sin\varphi \\ sin\varphi & cos\varphi \end{bmatrix}, \; t = {}^{1}/_{cos\theta} \tag{1}$$

Morel et al. proposed the proper range and sampling step of t and φ in [2]. In Equation (1), t ranges from 1 to $4\sqrt{2}$, and φ ranges from 0° to 180°. The sampling step of tilt (Δt) is $\sqrt{2}$, and the sampling step of φ ($\Delta\varphi$) is $72\ ^\circ/t$. The number of simulated images is 42 when the sampling range and step in Reference [2] are used. When a simulated image is obtained by affine transform, the SIFT algorithm in Reference [1] is used to generate SIFT features.

When an ASIFT hardware fetches a source image from the external memory implemented by a Dynamic Random-Access Memory (DRAM), the image is fetched in a rotated manner because of the rotation matrix R_φ of A in Equation (1). This means that the ASIFT accelerator accesses external memory in discontinuous order so that a burst transfer cannot be requested, which slows down DRAM access significantly.

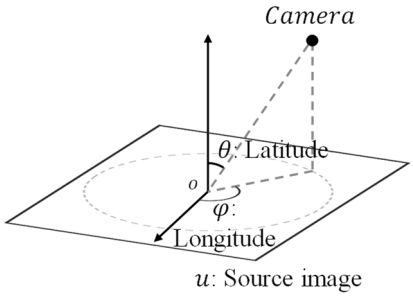


Figure 1. An interpretation of a camera position of the affine decomposition.

2.2. ASIFT Hardware Accelerator

In order to increase the processing speed of an ASIFT algorithm, Yum. et al. [8] proposed a hardware implementation of an ASIFT algorithm. This hardware adopts a modified affine transform matrix B to reduce the latency of an external memory, which is given by Equation (2). Matrix B consists of a scaling matrix $T_{sx,sy}$ and a skewing matrix S_g, but a rotation matrix is not included. Thus, the source image is fetched in a continuous order, and a burst transfer mode can be used.

$$B = S_g T_{sx,sy} = \begin{bmatrix} 1 & g \\ 0 & 1 \end{bmatrix} \begin{bmatrix} sx & 0 \\ 0 & sy \end{bmatrix}$$
$$s_x = \frac{1}{t}\sqrt{t^2 cos(\varphi)^2 + sin(\varphi)^2},\ s_y = \frac{1}{\sqrt{t^2 cos(\varphi)^2 + \sin(\varphi)^2}} \tag{2}$$
$$g = \tan(\tau) = \left(\frac{1}{t} - t\right)\sin(\varphi)\cos(\varphi)$$

The ASIFT hardware architecture proposed by Yum et al. [8] is shown in Figure 2. In this figure, gray blocks are internal and external buffers, and a striped block stands for a bus system connecting the ASIFT accelerator and the external memory. In order to increase the operation speed, the ASIFT hardware accelerator adopts one of the state-of-the-art architectures of a SIFT hardware accelerator proposed in Reference [7].

In order to reduce the computational load, the ASIFT hardware accelerator proposed by Yum et al. [8] increases the tilt sampling step (Δt) from $\sqrt{2}$ to 2, which further reduces the computational load to 43%. Due to this simplification, the number of viewpoints is decreased from 42 to 16.

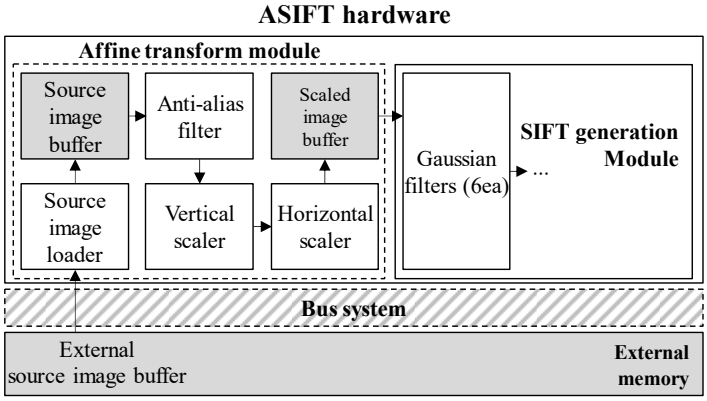


Figure 2. ASIFT hardware architecture proposed by Yum et al. [8].

2.3. Analysis of Hardware Utilization of Previous Work

The throughput of the affine transform module of the ASIFT hardware proposed by Yum et al. [8] is limited by the anti-alias filtering and the down-scaling operation. Figure 3 explains the cause of the slow operation. This figure shows an example in which a simulated image is generated with sx = 0.25 and sy = 0.25 in the affine transform module. In Figure 3, the gray circles represent the pixels of the source image stored in the source image buffer, and the white circles are the pixels of the source image to be fetched from the external memory. A dotted rectangular box corresponds to a kernel of anti-alias filter, and the black circles are the pixels that have been already filtered by the kernel. A vertical scaler employs nearest neighbor (NN) interpolation and provides the valid row-address to the anti-alias filter so that it can process only the required pixel lines. A horizontal scaler also performs NN interpolation, and the striped circles indicate the sampled pixels by the interpolation.

The affine transform module is the throughput bottleneck of the ASIFT operation. The first reason is the slow operation time of the anti-alias filter. The anti-alias filter is applied to a pixel at each cycle;

thus, the throughput of the filter is 1 byte/cycle, which is slower than that of source image loader (2.46 bytes/cycle). The second reason is that the speed of data fetch required for the filter is not fast enough. When $1/sy$ is large, pixels are filtered sparsely in the vertical direction, which means that the speed of data fetch needs to be increased. Figure 3 shows an example in which the filtering for the fifth line is completed, and the filtering for the ninth line should start. However, the 12th line data is not fetched yet. In this case, the vertical scaler waits for the required data on the 12th line. The third reason is that the ratio of the valid data in a line is low when $1/sx$ is large. In Figure 3, only every fourth pixel, which is a striped circle, in the horizontal direction is selected for the down-sampled image, and the rest are discarded. Thus, the throughput of the horizontal scaler is decreased because of waiting for valid data.

In order to fully utilize the SIFT generation module, the throughput of the affine transform module needs to be improved by up to 31 bytes/cycle. As shown in Table 1, however, the affine transform module proposed by Yum et al. [8] does not satisfy this condition for ASIFT(2)–ASIFT(15). ASIFT(i) represents the ASIFT operation for viewpoint i. This means that the SIFT generation module stays idle waiting for data from the affine transform module.

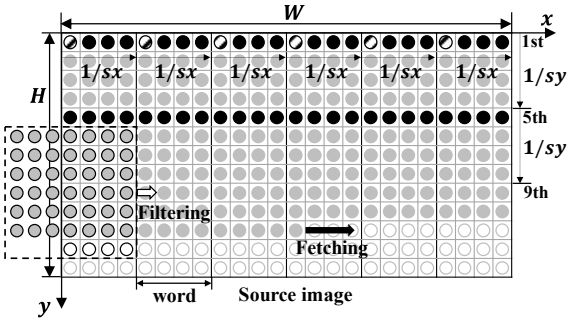


Figure 3. An interpretation of the throughput of the affine transform module in the ASIFT hardware proposed by Yum et al. [8].

Table 1. Throughput of the affine transform module proposed by Yum et al. [8].

Viewpoint Index	t	Longitude (°)	sx	sy	Throughput (pixels/cycle)
0	1	0	1.000	1.000	31.00
1	2	0	1.000	0.500	31.00
2	2	36	0.861	0.581	26.69
3	2	72	0.567	0.882	17.58
4	2	108	0.567	0.882	17.58
5	2	144	0.861	0.581	26.69
6	4	0	1.000	0.250	19.08
7	4	18	0.954	0.262	19.07
8	4	36	0.822	0.304	19.07
9	4	54	0.622	0.402	19.08
10	4	72	0.390	0.640	12.09
11	4	90	0.250	1.000	7.75
12	4	108	0.390	0.641	12.09
13	4	126	0.621	0.403	19.10
14	4	144	0.822	0.304	19.07
15	4	162	0.954	0.262	19.07

3. Proposed Schemes and Hardware Architecture

3.1. Increasing the Throughput of Affine Transform Module

For the increase of the throughput of the affine transform module, this paper proposes a *prior anti-aliasing scheme* and a *prior down-scaling scheme*. In the *prior anti-aliasing scheme*, a 1×1 blurred image is generated and stored in the external memory when ASIFT(0) is processed. For ASIFT(1)–ASIFT(15), the ASIFT hardware reads the proper pixel lines from the 1×1 blurred image stored in the external

memory. Figure 4a shows an example in which anti-alias filtering is not performed, but filtered data (black circle) are fetched from the external memory. Because the filtering computations for ASIFT(1)–ASIFT(15) are removed, the slow operation time of filtering does not limit the throughput of the affine transform module, and the speed of data fetch does not cause a stall in the vertical scaler with large $1/sy$. The throughput of the vertical scaler is increased up to the throughput of the source image loader (2.46 bytes/cycle).

The proposed *prior down-scaling scheme* computes a 1/2 down-sampled image of the 1×1 blurred image in the horizontal direction. This down-sampled image is referred to as a 0.5×1 blurred image. It is stored in the external memory when ASIFT(0) is processed. The *prior down-scaling scheme* increases the throughput of the horizontal scaler by 2 times when sx is smaller than 0.5. Figure 4b presents an example when the *prior down-scaling scheme* is not adopted and $sx = 0.25$. In this figure, the horizontal scaler samples only the first byte of each word and transfers it to the scaled image buffer at a cycle, and consequently, the throughput is decreased to sx. Figure 4c shows an example when the scheme is adopted. In the proposed scheme, the affine transform module fetches the 0.5×1 blurred image, and down-samples it with a modified scaling ratio $sx' = sx \times 2$. As a result, the throughput of the horizontal scaler is increased by 2 times.

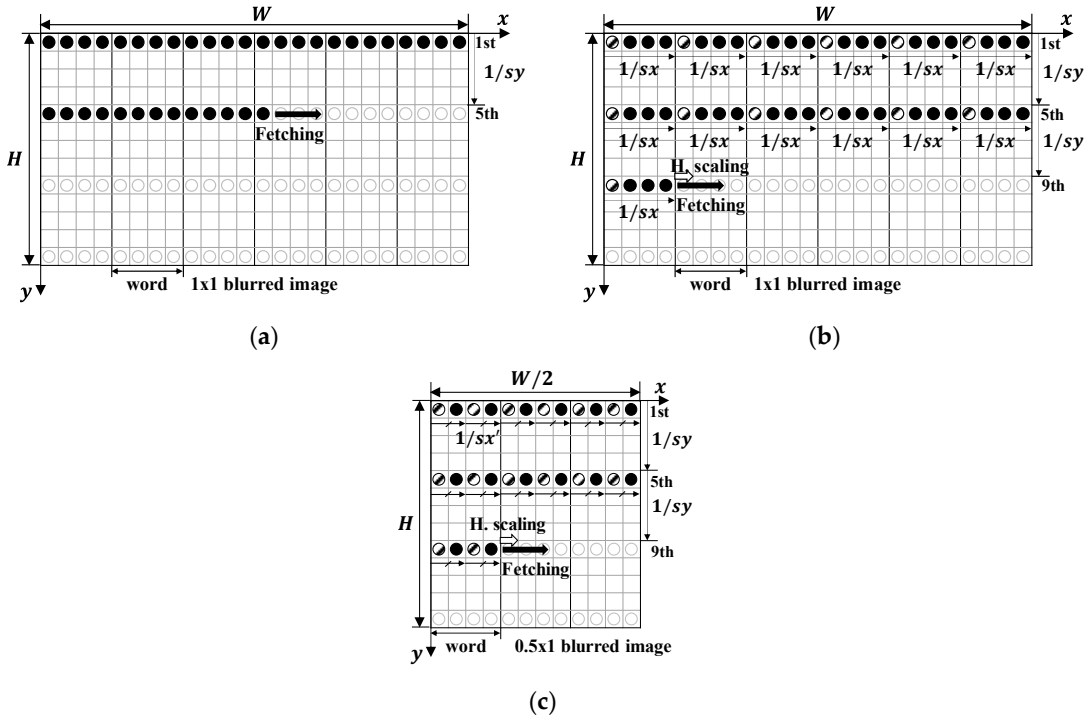


Figure 4. Interpretations of the improvement in throughput; (**a**) proposed *prior anti-aliasing scheme*; (**b**) 1×1 blurred image in the *prior down-scaling scheme*; (**c**) 0.5×1 blurred image in the *prior down-scaling scheme*.

3.2. Proposed ASIFT Hardware Architecture

In this paper, a modified ASIFT hardware architecture is proposed as shown in Figure 5. The proposed hardware architecture consists of an affine transform module and SIFT generation module. The affine transform module is a modified version of that in previous work [8], and the architecture of the SIFT generation module is the same as that in Reference [8]. In order to obtain the ASIFT features for all viewpoint indices, source mux selects the proper source data among three external buffers according to a current viewpoint index, and this operation is performed repeatedly. The first operation of the ASIFT hardware accelerator is the derivation of ASIFT(0) using the source image. For this operation, the source mux selects the external source image buffer. Because a transform

matrix for ASIFT(0) is the identity matrix, the fetched source image is transferred to the SIFT generation module without any scaling operation. At the same time, it is provided to the anti-alias filter and horizontal 1/2 scaler to generate the 1×1 blurred image and the 0.5×1 blurred image, and then they are stored in an external memory. After the ASIFT(0) operation, the derivation of ASIFT features for any other viewpoint indices can be operated as there is no dependency among ASIFT(1)–ASIFT(15). The ASIFT hardware uses a proper image between the 1×1 blurred image and the 0.5×1 blurred image according to the prior horizontal scaling ratio $psx_i = ceil(sx_i * 2)/2$. If psx_i is 1, the 1×1 blurred image is selected. Otherwise, the 0.5×1 blurred image is used.

Table 2 shows the design specification of the proposed ASIFT hardware. The proposed hardware is synthesized with 130-nm process technology by Design Compiler DC Ultra Version L-2016.03 designed by Synopsys (California, United States). The gate count of the hardware is 505 K, the maximum operating frequency is 190 MHz, and the size of internal memory is 467.5 Kbits. The proposed hardware uses 12.6 Mbits of space of the external memory for a VGA-sized image.

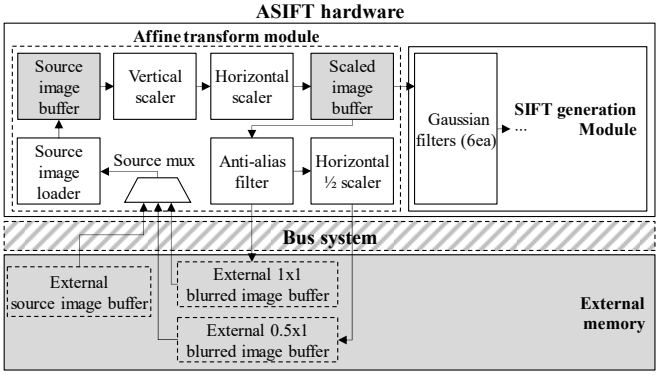


Figure 5. Proposed ASIFT hardware architecture. This hardware consists of an affine transform module, which is a modified version of previous work [8], and a SIFT generation module, which is the same as that in Reference [8]. The affine transform module computes 1×1 and 0.5×1 blurred images and stores them in the external memory when the viewpoint index is 0. Then, this module reuses proper source data among three types of data in the external memory according to a viewpoint index.

Table 2. Implementation results of the proposed hardware architecture.

Technology	130 nm
Maximum operating frequency	190 MHz
Gate count (except memory)	505 K
Internal memory size	467.5 Kbits
External memory size	12.6 Mbits

4. Results and Discussion

Experiments were carried out under the same conditions as the previous work [8]. The proposed ASIFT hardware uses a Synchronous Dynamic Random-Access Memory (SDRAM) as an external memory. The initial latency of the SDRAM is 11 cycles. In the proposed hardware, the first word is received after 11 cycles, while the next data are received in every cycle. A bus system connecting the ASIFT hardware to the external memory supports a burst transfer of length 16, and a word consisting of 4 bytes is transferred for each cycle by the bus system. The size of one pixel is 1 byte.

4.1. Throughput

In order to evaluate an enhancement by the proposed schemes, Table 3 presents a comparison of the throughput of three affine transform modules. The first column represents the viewpoint index, and the second column presents the throughput of the affine transform module proposed by Yum et al. [8]. Except for viewpoints 0 and 1, the throughput of the affine transform module is less than 31 bytes/cycle,

which means the SIFT generation module is not fully utilized for almost all viewpoints. The third column shows the throughput of the affine transform module with the *prior anti-aliasing scheme*, and the fourth column presents the results with both the *prior anti-aliasing* and *prior down-scaling schemes*. When the *prior anti-aliasing scheme* is adopted, the average of the throughput is increased to 30.10 bytes/cycle. When the *prior down-scaling scheme* is adopted additionally, the throughput of the proposed affine transform module is increased to 31 bytes/cycles on average, which means the SIFT generation module of the proposed architecture is fully utilized. For the viewpoints of index 10–12, the *sx* is smaller than 0.5, and the throughput of the horizontal scaler increases by using 0.5×1 blurred image.

Table 3. Comparison of the throughput of the three affine transform modules.

Viewpoint Index	Previous Work [8] (bytes/cycle)	Prior Anti-Aliasing (A) (bytes/cycle)	A + Prior Down-Scaling (bytes/cycle)
0	31.00	31.00	31.00
1	31.00	31.00	31.00
2	26.69	31.00	31.00
3	17.58	31.00	31.00
4	17.58	31.00	31.00
5	26.69	31.00	31.00
6	19.08	31.00	31.00
7	19.07	31.00	31.00
8	19.07	31.00	31.00
9	19.08	31.00	31.00
10	12.09	29.76	31.00
11	7.75	19.08	31.00
12	12.09	29.76	31.00
13	19.10	31.00	31.00
14	19.07	31.00	31.00
15	19.07	31.00	31.00
Average	19.75	30.10	31.00

4.2. Operation Speed

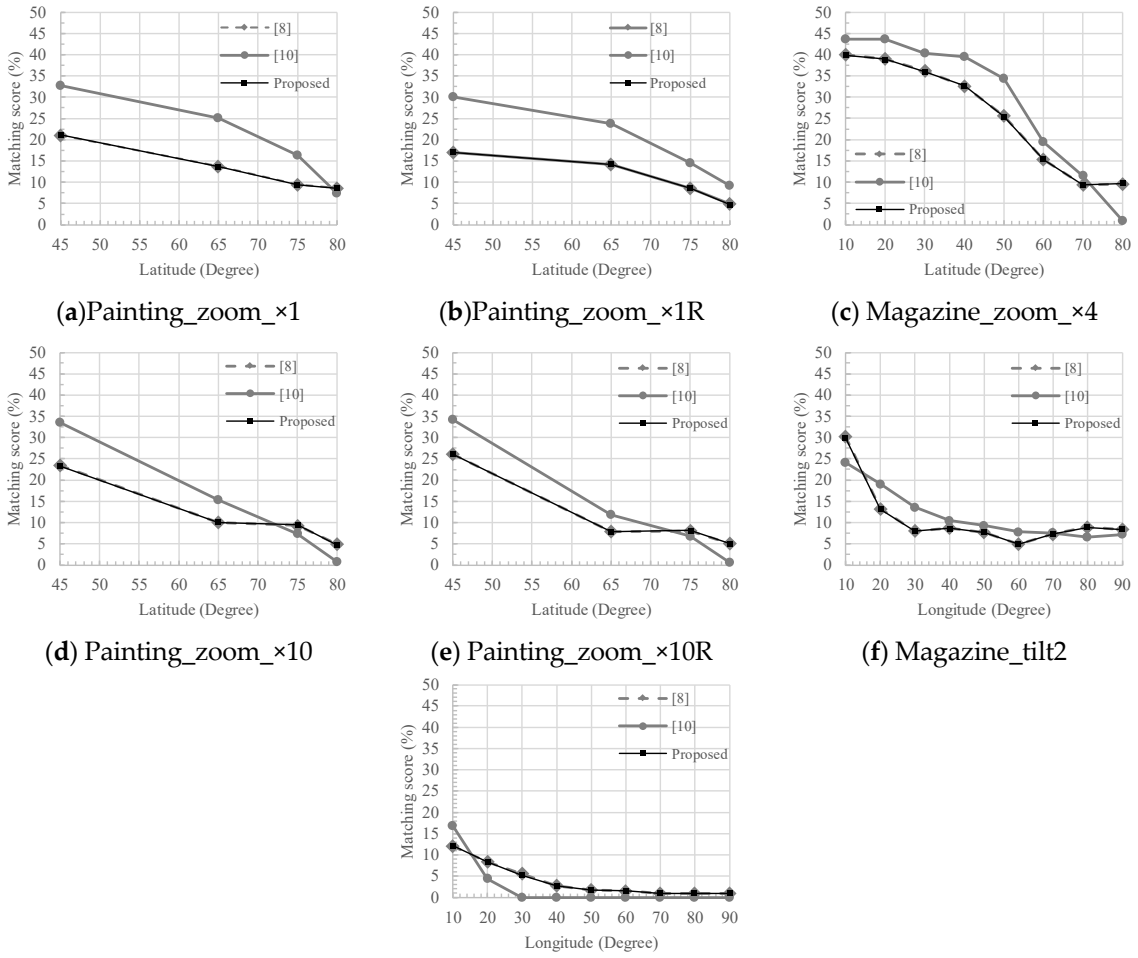
In order to measure the operation speed of the ASIFT hardware accelerator for a VGA-sized image, Register-Transfer Level (RTL) simulation was carried out with the operating frequency of 190MHz, which is the maximum frequency of the proposed hardware. The test images proposed by Mikolajczyk et al. [9] were used. The experimental results are shown in Table 4. The first column represents test images. The second and third columns show the number of keypoints and the operation time of the ASIFT hardware accelerator proposed in Reference [8]. As shown in Table 4, by adopting the proposed pre-processing scheme, the operation speed is increased to 1.36 times on average. The number of keypoints is not exactly the same because the scaled images computed in the previous work and by the proposed hardware accelerator are not exactly the same.

Table 4. Comparison of the operation time in previous work [8] and that obtained with the proposed hardware for VGA images.

Test Image	Previous Work [8]		Proposed Method	
	Number of Keypoints	Operation Time (ms/frame)	Number of Keypoints	Operation Time (ms/frame)
graf1	2598	46.51	2605	33.04
bark1	4277	57.93	4278	46.40
boat1	3133	49.76	3131	37.07
tree1	3191	49.80	3199	37.32
leuven1	2237	44.33	2221	30.86
ubc1	2558	46.11	2550	33.22
bike1	2612	46.22	2606	33.35
wall1	2405	44.78	2398	31.98
Average	**2877**	**48.18**	**2874**	**35.41**

4.3. Matching Accuracy

In order to compare the matching accuracy, Figure 6 presents the matching scores of the proposed accelerator, the previous design [8], and another previous design, GPU-ASIFT, proposed by Codreanu et al. [10]. The matching score is a metric for evaluating the matching accuracy of local features [9]. The test image sets given by Morel et al. [2] were used for the experiments. In Figure 6, the matching scores of the proposed method are the same as those of the previous design [8]. The average matching score of GPU-ASIFT [10] is higher than that of the proposed architecture in most images. As the latitude and longitude increase, however, the matching score of GPU-ASIFT drops significantly and even falls to zero in certain conditions. This means that GPU-ASIFT does not maintain the characteristic of affine invariance. On the other hand, the proposed ASIFT hardware accelerator derives the correspondences in all ranges of the latitude and longitude.



(**a**)Painting_zoom_×1 (**b**)Painting_zoom_×1R (**c**) Magazine_zoom_×4

(**d**) Painting_zoom_×10 (**e**) Painting_zoom_×10R (**f**) Magazine_tilt2

(**g**) Magazine_tilt4

Figure 6. Comparisons of the matching score of previous works [8,9] and the proposed hardware.

5. Conclusions

This paper proposes an ASIFT hardware architecture for enhancing operation speed. By increasing the throughput of the bottleneck module, the utilizations of the ASIFT hardware modules are increased, and the throughput of the entire hardware accelerator is increased as well. The proposed *prior anti-aliasing scheme* reuses the anti-alias filtered image. The computation speed is improved by removing the redundant operation of anti-alias filtering. The proposed *prior down-scaling scheme* reuses the filtered image that is down-sampled by 1/2 in the horizontal direction. This scheme doubles the throughput of the horizontal scaler module when the width of a simulated image is scaled lower than

0.5. Thanks to the proposed methods, the throughput of the bottleneck module, which is an affine transform module, is increased up to 31 bytes/cycle, which makes the ASIFT hardware fully utilized for all viewpoints. The operation speed of the proposed accelerator is increased up to 1.36 times on average compared with previous work [8] without a degradation of matching accuracy. As a result, the proposed ASIFT hardware processes a VGA image at 28 frames/second.

Author Contributions: Conceptualization, methodology, simulation, J.Y.; data analysis, J.Y. and J.-S.K.; validation, J.Y., J.-S.K., and H.-J.L.

Funding: This work was supported by the 2018 Sun Moon University Research Grant.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Lowe, D. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vis.* **2004**, *60*, 91–110. [CrossRef]
2. Morel, J.-M.; Yu, G. ASIFT: A new framework for fully affine invariant image comparison. *SIAM J. Imaging Sci.* **2009**, *2*, 438–469. [CrossRef]
3. Hsu, P.H.; Tseng, Y.C.; Chang, T.S. Low memory cost bilateral filtering using stripe-based sliding integral histogram. In Proceedings of the 2010 IEEE International Symposium on Circuits and Systems, Paris, France, 30 May–2 June 2010; pp. 3120–3123.
4. Bonato, V.; Marques, E.; Constantinides, G.A. A parallel hardware architecture for scale and rotation invariant feature detection. *IEEE Trans. Circuits Syst. Video Technol.* **2008**, *18*, 1703–1712. [CrossRef]
5. Huang, F.C.; Huang, S.Y.; Ker, J.W.; Chen, Y.C. High-performance SIFT hardware accelerator for real-Time image feature extraction. *IEEE Trans. Circuits Syst. Video Technol.* **2012**, *22*, 340–351. [CrossRef]
6. Kim, E.S.; Lee, H.-J. A novel hardware design for SIFT generation with reduced memory requirement. *J. Semicond. Technol.* **2013**, *13*, 157–169. [CrossRef]
7. Yum, J.; Lee, C.-H.; Kim, J.-S.; Lee, H.-J. A Novel Hardware Architecture with Reduced Internal Memory for Real-time Extraction of SIFT in an HD Video. *IEEE Trans. Circuits Syst. Video Technol.* **2016**, *26*, 1943–1954. [CrossRef]
8. Yum, J.; Lee, C.-H.; Park, J.; Kim, J.-S.; Lee, H.-J. A hardware architecture for the affine-invariant extension of SIFT. *IEEE Trans. Circuits Syst. Video Technol.* **2017**, *28*, 3251–3261. [CrossRef]
9. Mikolajczyk, K.; Tuytelaars, T.; Schmid, C. A Comparison of Affine Region Detectors. *Int. J. Comput. Vis.* **2005**, *65*, 43–72. [CrossRef]
10. Codreanu, V.; Dong, F.; Liu, B. GPU-ASIFT: A fast fully affine-invariant feature extraction algorithm. In Proceedings of the IEEE High Performance Computing and Simulation (HPCS), Helsinki, Finland, 1–5 July 2013; pp. 474–481.