Multimedia Tools and Applications (2024) 83:6297–6317
https://doi.org/10.1007/s11042-023-15789-w

Hardware architecture design for real-time SIFT extraction with reduced memory usage

Tsung-Han Tsai[1] · **Rui-Zhi Wang**[1] · **Nai-Chieh Tung**[1]

Received: 6 December 2022 / Revised: 16 March 2023 / Accepted: 2 May 2023 /
Published online: 25 May 2023
© The Author(s), under exclusive licence to Springer Science+Business Media, LLC, part of Springer Nature 2023

Abstract
Scale-invariant feature transform (SIFT) is considered one of the best algorithms to get feature points in an image. It maintains the accuracy in results even in image scaling, rotation, deformation, and light changes. However, memory requirements are the bottleneck to achieving real-time performance as SIFT has high computation complexity. Therefore, this paper has proposed the improved hardware architecture of the scale-invariant feature transform (SIFT) algorithm. The Gaussian pyramid is constructed using parallel operations instead of the original cascade operation to reduce memory requirements. Coordinate Rotation Digital Computer (CORDIC) has been adapted to perform trigonometric operations to reduce computational complexity. The ASIC design has been implemented using TSMC 90 nm technology. The system can achieve the performance of 35.6 FPS for an image resolution of 1280×720 while using only 237.4 Kbit of memory.

Keywords Real-time · Key-point · Object recognition · Gaussian pyramid · Low memory VLSI design

1 Introduction

Computer vision has enabled the computer to have human-like intelligence and perception through image recognition using image processing. At the core of many applications, object detection and matching are inevitable where the operations depend upon image features such as surveillance systems [5], human identification [42], etc. The main purpose of feature extraction is to obtain important information from the color, texture, and shape of the original image and present it in a lower-dimensional space. These key points contain the position and descriptor, where the position provides information about the key-point location, and the descriptor provides key-point for unique feature data.

In many object detection algorithms, objects are detected using different image features. Key-point does not represent just a pixel where it includes different information from a series of images. In computer vision and image processing, key-point is produced

✉ Tsung-Han Tsai
han@ee.ncu.edu.tw

[1] Department of Electronics Engineering, National Central University, Taoyuan, Taiwan

by color, texture, and shape, where color, texture, and shape are information that can be used to extract important information. The main purpose of feature extraction is to obtain important information from the original image and present it in lower-dimensional space. These key-points contain the position and descriptor, where the position provides information about the key-point location and the descriptor provides key-point unique feature data.

There are multiple categories of feature detection algorithms, among which the following three are more common. They are edge detection, corner detection, and blob detection. Edge detection is to find the edge where the brightness changes and uses the grayscale value to detect brightness change using gradient and Laplace calculations. Sobel and Canny edge detection algorithm [8] are a common example of the edge detection algorithm. For corner detection, corners in the image are detected where the junction of two edges is considered a corner. This class contains algorithms such as the Harris operator [16], SUSAN operator [38], FAST [36], etc. Blob detection can detect different regions based on different properties such as color, contrast, or brightness. In the first step, the Difference of Hessian (DoH) or Laplacian of Gaussian (LoG) is used to detect key points. Then these key points are used to detect blobs. This class includes algorithms such as SIFT [31], SURF [6], etc.

Key point acquisition in image processing often determines the stability and performance of the whole system. SIFT is considered one of the best algorithms in many key points detection algorithms. The key points detected by SIFT are local features of the image, and those key points are unaffected by image rotation, scale zoom, light, and shadow variation, with good robustness to affine changes, and also have good noise immunity. Based on these five characteristics, they are highly visible and easy to capture, and it is easy to identify objects with few misidentifications in a large database of features. The detection rate of obscured parts of objects described by SIFT features is quite accurate, and even only 3 or more SIFT features are needed to calculate the position and orientation of an object. Therefore, SIFT can be utilized for various applications, such as image retrieval [19, 25], object tracking [15, 45], object detection [4, 13], and image registration [21].

Although SIFT has excellent performance [1, 2, 14], it is hard to achieve real-time computation because of complex calculations and large memory requirements for high-resolution images. And this algorithm has the other disadvantage that when the image is blurred then only a few of the key-points can be detected [10, 22, 39]. The algorithm is not able to detect the key-points correctly in case of changes to the image.

In this paper, a memory-efficient hardware design for SIFT algorithm has been proposed for feature extraction. With this advantage, it can reduce memory usage and save hardware complexity. The proposed hardware is highlighted as follows:

- We construct the multiple parallel Gaussian filters, which discard the original cascade operation and replace it with the parallel operation to detect the extreme values.
- We use a smaller initial scale in the Gaussian pyramid to prevent the loss of high-frequency image information while preserving the high spatial frequency.
- We eliminate only the key points with low contrast, which not only eliminates 50% to 60% of the total number of key-points, but also reduces the number of calculations.
- The computation complexity is further reduced by using Coordinate Rotation Digital Computer (CORDIC) architecture to perform trigonometric calculations.

We organize the rest of the paper as follows. The related work for feature extraction algorithms and different hardware implementations is provided in Section II. Section III includes the details of SIFT algorithm. Section IV includes a description of the hardware

implementation of the proposed algorithm. Finally, the implementation results are shown in section V, and the conclusion is provided in section VI.

2 Related works

Due to the good performance of SIFT in feature extraction from the image, many improved methods have been proposed based on SIFT, such as PCA-SIFT [23], SURF [6], GLOH [33], etc. Among these improved methods, Principal components analysis (PCA) is a type of statistical analysis and simplification of variables. The principle of Principal Components Analysis (PCA) is to find the correlation between multiple variables and use a few principal component variables to represent the relationship between the variables. The main purpose is to reduce the dimension of the dataset while preserving the features that contribute most to the variance of the dataset. The method decomposes the variance matrix to obtain the eigenvectors and eigenvalues. The eigenvector represents the main components of data, and the eigenvalue represents the weight of the main components.

In [23], PCA has been applied to SIFT. Its purpose is used to normalize the gradient near the key-point, and the histogram is replaced to represent the gradient of the local area. Compared to SIFT, PCA-SIFT's eigenvector has been reduced, and Euclidean distance can be used to calculate whether the key- point matches the same key-point at the same time. In PCA-SIFT, the gradient values of the key point neighbors are sampled after calculating the principal direction, which is 41×41 image blocks. Then the horizontal and vertical gradient images of each image block are concatenated to obtain an input vector of size $39 \times 39 \times 2$. This vector is normalized to reduce the effect of light and shadow variance. In the final step, the dimension of the input vector is reduced by applying PCA.

Speeded Up Robust Features (SURF) is a modification of the SIFT algorithm to improve the computational speed. The difference between SURF and SIFT is the method of finding the key-point. SIFT uses the Difference of Gaussian (DoG) to approximate LoG, while SURF uses the concept of the Haar-wavelet feature and integral image to approximate DoH. SURF simplifies the Gaussian second-order differentiation into a box filter to reduce the computational effort. The simplified filter consists of only a few moment blocks and each with the same value. Therefore, when convolving with the simplified filter, the concept of an integral diagram can be applied to accelerate the operation.

Gradient Location and Orientation Histogram (GLOH) is a deformed SIFT descriptor purposed by Mikolajczyk and Schmid [33]. GLOG replaces the 4×4 subregion of SIFT with a logarithmic coordinates representation. As shown in Fig. 1, the GLOH descriptor is divided into 17 sub-regions, and the gradient direction statistics are calculated for each sub-region, which is divided into 16 directions. The 272-dimensional feature vector is obtained and then downscaled to 128 dimensions by PCA.

For SIFT and PCA-SIFT, the descriptor generation process takes a lot of time. Since PCA-SIFT uses PCA to reduce the dimension of the key-point, the time required in the key-point matching process is reduced and the overall speed is better than SIFT. However, the disadvantage is that the resistance to image changes such as rotation, enlargement, and blurring is worse than SIFT.

The improvement of SURF to SIFT is to use fast hessian to replace DoG in SIFT, which is about three times faster than SIFT and better than PCA-SIFT. Moreover, SURF has invariance to the change of light and shadow, but other image deformation types such as

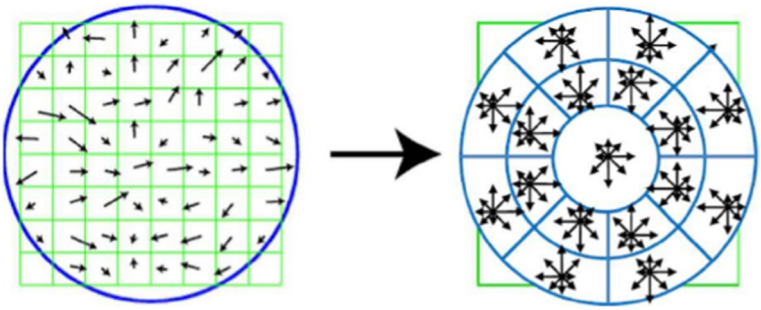


Fig. 1 GLOH descriptor [3]

rotation or scaling result in worst performance than SIFT. Among all other deformation types, rotation invariance causes the worst performance.

GLOH strengthens the part of the descriptor in SIFT and makes it stable. To save the matching time, GLOH uses PCA to reduce dimensions to 128. In GLOH, the key-point invariant performs better than SIFT, but the calculation complexity is higher than SIFT.

There are some hardware designs for SIFT in recent years. Huang et al. [17] have proposed a SIFT accelerator consisting of two interactive hardware components and a segment buffer scheme. The segment buffers scheme is responsible to send data to the computing modules. A Layer Parallel SIFT (LPSIFT) with an integral image and feature extraction has been proposed by Chiu et al. in [9]. The hardware implementation focuses on a highly parallel hardware design to achieve a real-time application. However, the key-point detection and descriptor generation part performs operations in series. So when the image size increases, the key-point detection consumes more time. As we use the parallel hardware design for the key-point detection part, it will not consume much time. Yum et al. [43] reduced the amount of internal storage required by storing data in external memory. The power consumption of [43] is quite high which makes it inefficient for a low-power implementation. Our design uses a smaller scale in the Gaussian pyramid because a smaller scale means that a smaller filter is required. Therefore it can reduce the memory requirement. Jiang et al. [20] designed a high-speed hardware SIFT architecture. It employs parallel and pipelined techniques for real-time image feature extraction. Doménech-Asensi et al. [12] utilized a pipeline structure in both the keypoint extraction and descriptor generation phases to achieve real-time processing.

In [28], VLSI implementation of SIFT descriptors has been proposed with a focus on efficient computing to achieve high processing speed. This architecture consists of the rotation module (RM), the Gaussian Weight Generation Module (GWGM), the Trilinear Interpolation Module (TIM), the Normalization Module (NM), and the controller. We designed the SIFT descriptor using the CORDIC architecture to reduce computational complexity. In [35], Qasaimeh et al. proposed a new hardware architecture to reduce the hardware complexity by the multiplierless multiple constant multiplications and the symmetrical property of the Gaussian mask when generating the Gaussian scale space.

The affine-SIFT algorithm even has high computation, it achieves robustness in image illumination, image rotation, and image scale transformation. The hardware design of Affine-SIFT has been proposed in [34, 44]. Ouyang et al. [34] proposed three optimized methods to increase computational efficiency, among the three methods, the most important method is reverse affine-based pipelined computing to optimize memory access. However, this technique results in high memory usage, which is not feasible for FPGA

implementation. Yum et al. [44] modified the affine transform algorithm to access external memory with a low accuracy drop in the raster-scan order.

Vourvoulankis et al. [40, 41] proposed a fully pipelined hardware architecture to generate a real-time SIFT descriptor. Then they use a Random Sample Consensus (RANSAC) algorithm to avoid false correspondences. To reduce the requirement of the logic element, Li et al. [27] present a new model to eliminate low-contrast without using any dividers. In the normalization module, they proposed a new architecture where only one divider is implemented. In addition, the SIFT module in [27] does not have a valid signal design. This means that only continuous input of data is allowed. Kreowsky et al. [24] present a generic pipelined hardware implementation that is capable of computing both keypoints and rotationally invariant descriptors in a single-precision floating-point format, which allows the original algorithm to be implemented with little or no modification. Sanchez et al. [37] propose a stream-based hardware acceleration architecture for SIFT feature extraction. The execution time is reduced by using multiple processing elements (PEs) to compute multiple SIFT descriptors in parallel. In addition, the proposed architecture supports key-point detection for an arbitrary number of octaves and allows various parameters to be configured at runtime.

3 Design of SIFT algorithm

3.1 Motivation

Although the performance of SIFT is good, it results in complex calculations and huge memory requirements, and is not easy to achieve real-time performance for a high-resolution image. In this paper, a low-power, memory-efficient hardware design for SIFT algorithm has been proposed for feature extraction. Multiple parallel Gaussian filters to detect extreme values have been used to address the need for high-speed design in hardware architecture. To prevent the loss of high-frequency image information, preserve the high spatial frequency, and reduce calculation complexity, a smaller initial scale is used to construct the Gaussian pyramid. Moreover, the computation complexity is further reduced by using CORDIC architecture to perform trigonometric calculations.

3.2 Overall algorithm design

SIFT algorithm performs the key-point detection in four steps: the first step involves the construction of the scale space and detection of the extremum. In the second step, key points are located through the fitting function, and the bad extremum is eliminated. In the next step, the density of the gradient and image direction is calculated. In the last step, the density of the gradient and the direction of an image around the selected scale are calculated and converted into another presentation to describe the key point. The algorithm flow is shown in Fig. 2.

The theory of scale-space means that visual information at different scales can be obtained by changing the scale. Then the information is combined to get the feature of the image. Lindeberg has proved in [29] that the Gaussian convolution kernel is the only linear transformation kernel to achieve the scale transformation. Thus the scale-space of the image is defined as the convolution of Gaussian function $G(x, y, \sigma)$ and the original image $I(x, y)$ as shown in (1), where σ represents the size of scale space. The larger scale image is

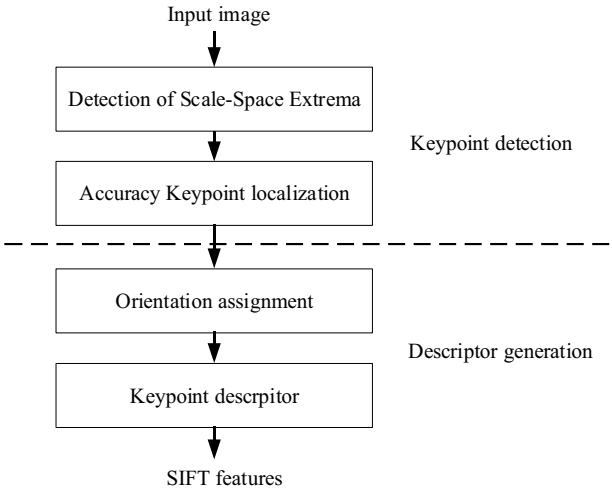


Fig. 2 SIFT algorithm

blurred to show the overall image overview, and the smaller scale shows the delicate lines in the image. * indicates the convolution operation.

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{1}$$

To detect extremum, the first step is to establish the Gaussian pyramids, which can be constructed by continuous scale change and downsampling. In contrast to the direct convolution of the original image with the Gaussian function, it means that the highest frequency spatial frequencies will be discarded. So in [31], D. Lowe used up-sampling of the original image to obtain higher frequency and then started building the Gaussian pyramid. In the next step, the extremum in scale space is found through the scale-normalized LoG. In the experiment in [32], Mikolajczyk has shown that the extrema of the scale-normalized LoG function can produce more stable features than other eigenfunctions, but the operational complexity is still too huge. SIFT uses DoG instead of LoG to carry out extremum detection. The DoG is subtracted from the image of two adjacent layers in the Gaussian pyramid. This method can effectively reduce the computation, and we can obtain the local extremum as shown in Fig. 3. Currently, each pixel needs to be compared with the other

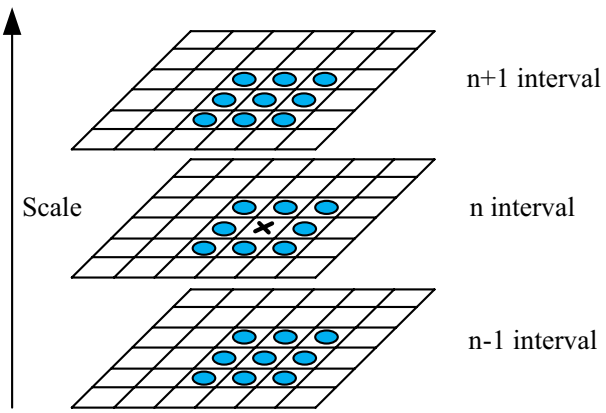


Fig. 3 DoG space extremum detection

eight points in the same interval and nine points in the upper and lower interval through the DoG pyramid. If this pixel is the maximum or minimum value of these 27 points, the pixel is considered the candidate key point.

The extremum found in the previous step is also the key point of discrete space. To improve the stability of the key point, the curve fitting of the DoG space is needed. Besides, these worse extremums are needed to filter out because the DoG operator is sensitive to noise. Here the Taylor expansion is used as a fitting function to correct the extreme points. The equation is shown in (2).

$$D(X) = D + \frac{\partial \mathrm{D}}{\partial X} X^T + \frac{1}{2} X^T \frac{\partial^2 D}{\partial X^2} \tag{2}$$

where $\mathrm{X} = (x, y, \sigma)^{\mathrm{T}}$ presents the deviation to the key-point. In any dimension, if the deviation is greater than 0.5, the extreme position should shift to the other points. So it is necessary to fit the extreme position to the new position, and then continue iterative operation until all three-dimensional converge has been performed. At the same time, if the number of iterations is greater than a certain number of times and the three dimensions are not converged, the extreme value will be discarded. The DoG function value of the extremum is used to discard the unstable extremum caused by the low contrast. Also, the unstable edge response points are eliminated through the Hessian matrix to find all the key points.

3.3 Rotation invariance

Another important property of SIFT is the rotation invariance. To make each key-point rotation-invariant, local image information around the key-point is calculated and each key-point is given a consistent main direction. Then the descriptor is represented relative to this main direction. The direction of the key point is based on the Gaussian image in the scale space where the key point is located. The circle region with a radius of r is selected as the center of the key-point and the gradient intensity $m(x,y)$ and the angle $\theta(x,y)$ of all pixels in the region are calculated. The equations are shown in (3) and (4).

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2} \tag{3}$$

$$\theta(x, y) = \tan^{-1}\left(\frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)}\right) \tag{4}$$

As the importance of neighbor points to key-point is different, it is necessary to weigh the gradient intensity and angles of all pixels in the region. Gaussian distribution of $1.5^* \sigma$ is adopted for weighting. After the gradient operation is completed, a histogram is created and all information is counted. Then the direction of the maximum peak value is selected as the main direction of the key-point.

Till this point, the coordinate position, the scale position, and the main direction of key-points have been obtained. In the next step, a descriptor for each key-point is created. In SIFT, the descriptor represents the Gaussian image gradient of points which lies in the neighborhood of key-points. A set of vectors is selected to describe this key-point. This descriptor should be highly unique and invariant for light, shadow, rotation, affine, etc.

As the descriptor is related to the scale of key-point, the Gaussian image descriptor with the scale of key-point is selected. The surrounding image is divided into 4×4 sub-regions

with the key-points as the center. To maintain the rotation invariance of the key-points, the main direction of each key point is rotated to the x-axis, and then the gradient direction histogram is recalculated for each sub-region. The direction histogram has eight columns for every 45 degrees by dividing 360 by 45. So the histogram can be obtained by a 128-dimensional ($4\times4\times8$) eigenvector. In addition, the eigenvectors need to be normalized to make the descriptor invariant to the change of light and shadow. So the threshold value of 0.2 is selected to improve the matching stability. This process results in the generation of SIFT eigenvector.

4 Hardware architecture

We implemented SITF on the CPU and present the result of different modules in Table 1. Because of SIFT's complex computation, it is difficult to achieve real-time computing on the CPU. Overall, it takes about 5 s to process an image with a resolution of 1280×720 in which scale-space extremum detection accounts for about 60% of the total computation time. Besides, SIFT algorithm needs to calculate and store the Gaussian pyramid, DoG pyramid, gradient magnitude orientation, and other data, which results in large memory storage. The huge memory requirements consume the hardware design more power and cause delays. Therefore, we propose a novel hardware architecture design to reduce the required memory storage. Detailed information will be introduced in this section.

4.1 Overall hardware architecture

To reduce the overall memory requirement without affecting the matching accuracy, we propose a suitable hardware architecture based on the consideration of reducing memory allocation. The overall hardware architecture is shown in Fig. 4. It can be divided into key-point detection and key-point descriptor. Key-point detection is responsible for calculating Gaussian blur images, and Gaussian differential images, and detecting extremes from DoG space. In the next step, bad key-points are filtered out and stored in memory. In the key-point descriptor module, when key-point FIFO is not empty, the key-point location information is extracted from the key-point, and then the pixels around the key-point are read from the external memory to calculate the density of gradient and direction. After counting the main direction, the region around the key points is rotated to count the eigenvectors. And then the calculated descriptors are stored in the external memory. The key-point FIFO is triggered to read the next data.

Table 1 SIFT calculation time analysis

	Run time(sec)	Percentage
Scale-space extrema detection	2.770	58.2%
Key-point localization	0.791	15.7%
Orientation assignment	0.166	3.3%
Key-point descriptor	1.148	22.8%
Total	5.031	100%

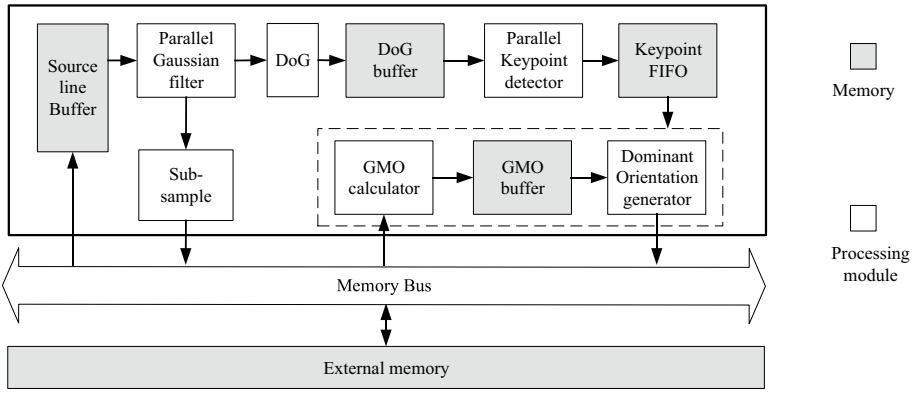


Fig. 4 Hardware architecture

4.2 The initial scale of the Gaussian pyramid

To keep the high-frequency image information and reduce the computation, we use a small initial scale to construct a Gaussian pyramid to obtain the high-frequency spatial frequency. In general, using a smaller-scale way means the smaller filter is used and derives a reduction of the calculation complexity and memory requirements for the Gaussian function. In this paper, we tested the feature point matching effect at a rotation angle of 30 degrees and an image size scaling ratio of 80%. As shown in Fig. 5, the number of key- points is the same as that of the original algorithm when the initial scale is in the range of 0.95 to 1.05, and the accuracy lost is just less than 2%.

4.3 Gaussian filter

As the Gaussian image has superposition, the Gaussian image of the present interval can be completed by performing Gaussian blur on the previous interval's Gaussian image. Using the characteristic of superposition, the Gaussian function can be calculated by a smaller standard deviation which can reduce the number of calculations effectively. However, the use of superposition arises the problem of data dependency by generating

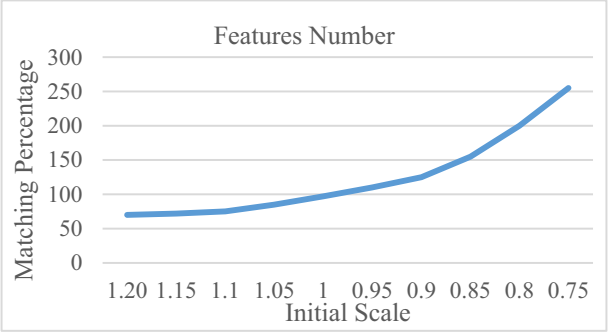


Fig. 5 Relationship between the initial scale and features

extra temporary data and delay. In [7], V. Bonato used a cascade architecture to construct the scale spaces and detect extreme values. But in cascade architecture, Gaussian images at different intervals are not calculated at the same time. And the previous interval results are required to calculate the current interval results.

To address the need for computational efficiency in hardware design, we use parallel architecture to detect extreme values. Figure 6 shows a parallel Gaussian filter architecture. The original image is stored inline buffer. We use 9 line buffers where each line buffer is 8 bits. The total size of the line buffer is 72 bits. Then the image information is filtered with different scales to construct a continuous scale space at the same time. To find the key point, the DoG space is subtracted from every two layers of the Gaussian image and stored in the DoG buffer.

In addition, based on the separability of the Gaussian function, the two-dimensional Gaussian function is separated into a one-dimensional Gaussian function based on (5). Here $G(x, y, \sigma)$ is a two-dimensional Gaussian function with a standard deviation of σ, $I(x, y)$ represents an original image, and $g1(x,\sigma),g2(y,\sigma)$ is denoted as one-dimensional Gaussian function. To calculate the two-dimensional Gaussian image, the Gaussian blur is performed to the original image in the x-direction and the y-direction. This method can effectively reduce the number of calculations as shown in Fig. 7. The same column of data is read from the line buffer at the same time for vertical Gaussian blurring, and then the calculated one-dimensional Gaussian image is blurred in the horizontal direction one pixel at a time in sequence.

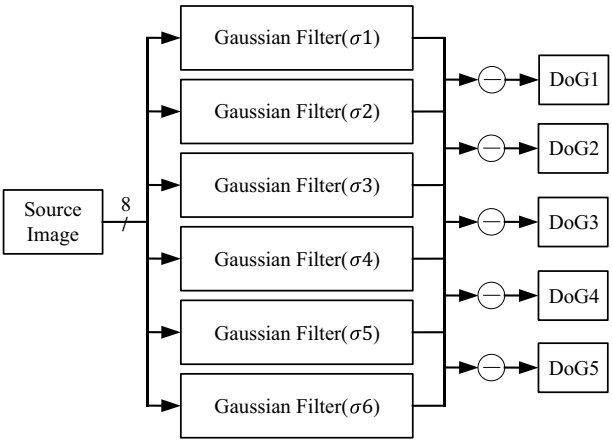


Fig. 6 Parallel Gaussian filter

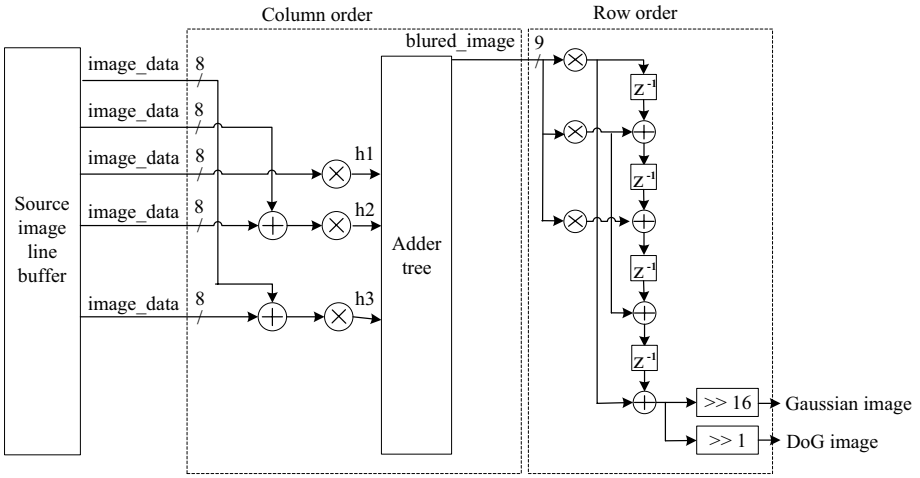


Fig. 7 Separate Gaussian filter

$$
\begin{aligned}
&\mathrm{G(x,y},\sigma) * I(x,y) \\
&\quad = \sum_{k=1}^{m}\sum_{l=1}^{n} G(k,l)I(x-k,y-l) \\
&\quad = \sum_{k=1}^{m}\sum_{l=1}^{n} ke^{\frac{-k^2}{2\sigma^2}} I(x-k,y-l) \\
&\quad = \mathrm{k}\sum_{k=1}^{m} e^{\frac{-k^2}{2\sigma^2}} \left\{ \sum_{l=1}^{n} e^{\frac{-l^2}{2\sigma^2}} I(x-k,y-l) \right\} \\
&\quad = I(x,y) * g_1(x,\sigma) * g_2(y,\sigma)
\end{aligned}
\tag{5}
$$

4.4 Key-point detection

The module of the key-point positioning includes three-stage parallel architecture. This module can be divided into three parts: extreme point detection, low contrast elimination, and edge response. Figure 8 shows the parallel key-point detection architecture. The data is read from DoG line memory and stored in the 3×3 register, then extremum point detection is performed. The range of extreme value detection is 8 pixels around the detection point and the corresponding 9 pixels on the upper and lower levels. Then the three adjacent registers can be used to calculate whether the point is an extreme value by the comparator, and perform edge detection at the same time. Firstly, we need to calculate the gradient value of the point in different directions to obtain the Hessian matrix. Then we use this matrix to calculate the principal curvature of the point. Finally, the extreme points with principal curvature less than the set threshold are retained and perform curve-fitting on the retained extreme points to obtain the exact positioning.

In the part of key-point positioning, the original points are used in discrete space to get the accurate position and obtain the extremum points in continuous space through the interpolation method. However, this step requires multiple iterations and the calculation of

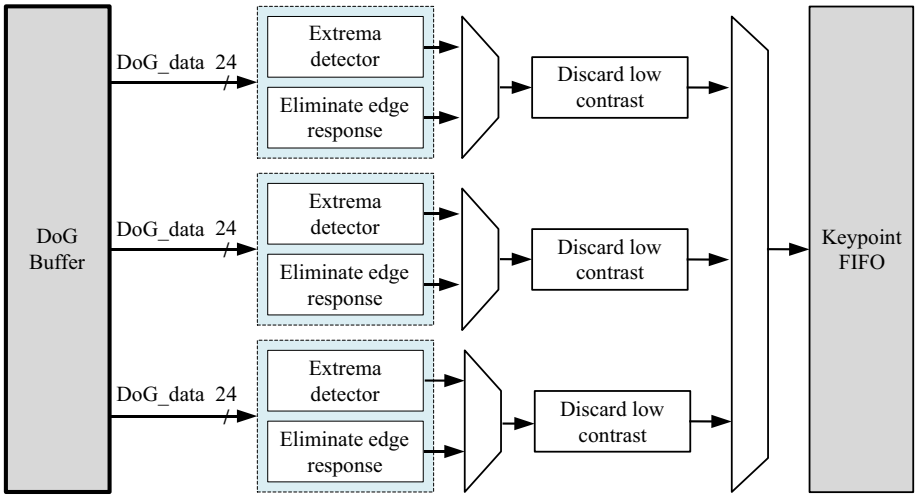


Fig. 8 Parallel key-point detection architecture

the third-order inverse matrix. This is not suitable for hardware implementation. Thus we consider the extremum value which can be eliminated. These can be divided into three categories: excessive deviation of positioning points, excess boundaries in the iteration process, and low contrast. Key points that are eliminated because of the low contrast are about 50% to 60% of total key points. Therefore, we delete the part about calculating positioning in continuous space. Only the key points with low contrast are eliminated.

4.5 Key-point descriptor

The key-point descriptor system is shown in Fig. 9. The information about the key points calculated in the previous step is stored in key-point FIFO, and the image information of the region around the key-point is read from external memory. As the external memory is

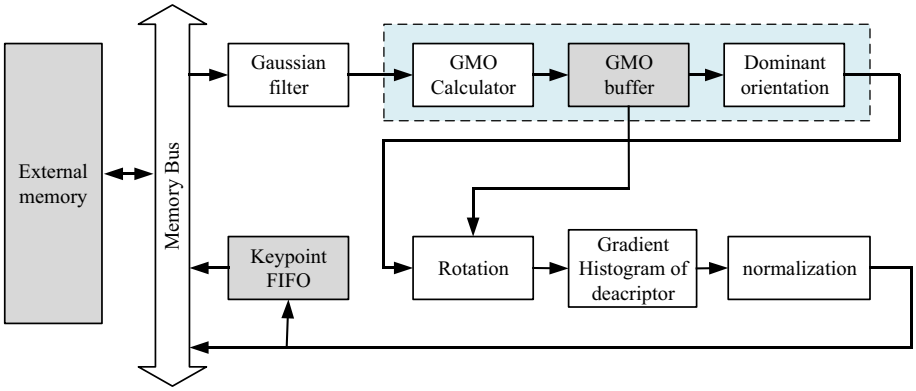


Fig. 9 Feature point descriptor system block diagram

stored with the first interval of the Gaussian image, Gaussian blur is needed to run again once a higher-scale Gaussian image is desired. In the next step, the density of the gradient and gradient direction of the neighborhood of the key-point is calculated. To reduce the amount of computation in hardware architecture, the CORDIC algorithm is selected to calculate the above-mentioned functions. The main concept behind CORDIC is the calculation of the vector rotation without the trigonometric function. By just using simple shift,

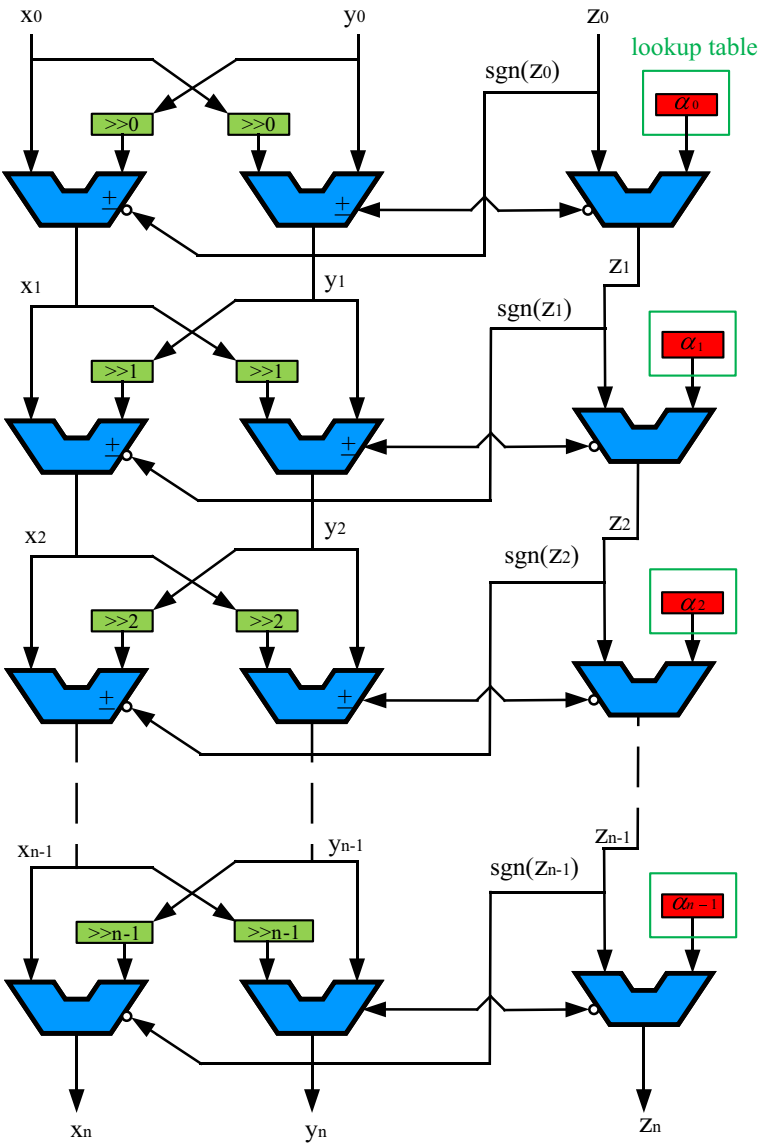
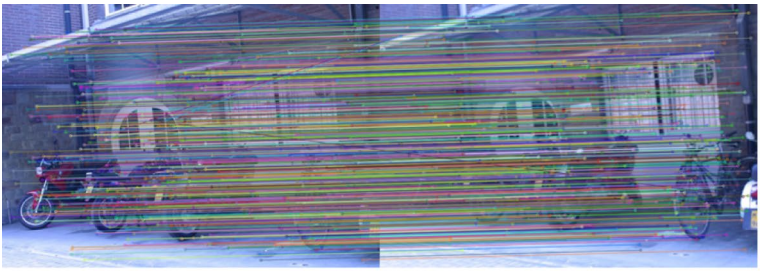
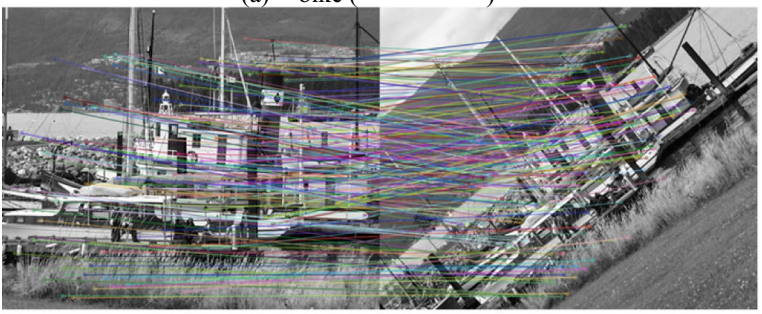


Fig. 10 CORDIC hardware architecture

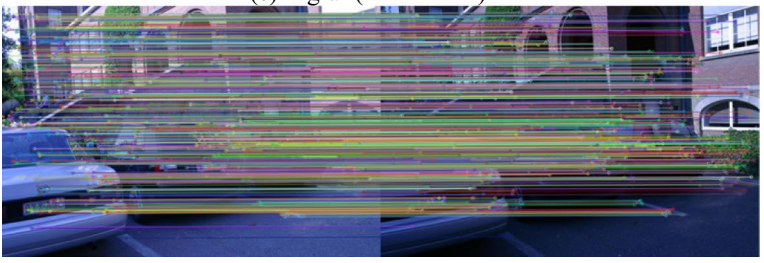


(a) bike (1000×700)



(b) boat (800×640)

(c) graf (800×640)



(d)leuven (921×614)

Fig. 11 The matching result of the affine covariant regions datasets

addition, subtraction, and iteration operation can be performed which is suitable for implementation on hardware.

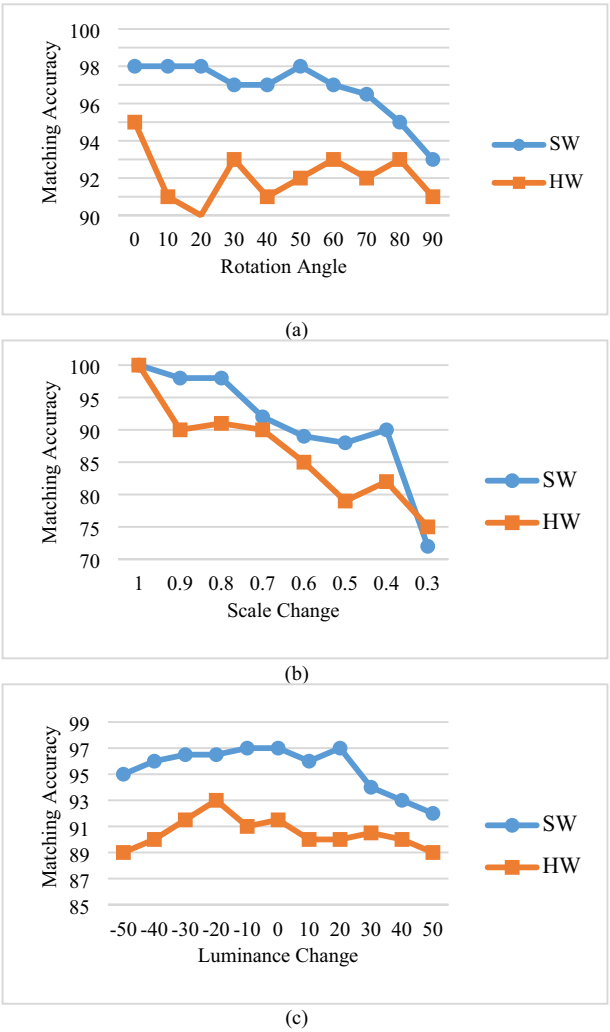


Fig. 12 Evaluation between software accuracy and hardware accuracy design. (**a**) The test of rotating invariance. (**b**) The test of scale invariance. (**c**) The test of light and shadow invariance

We use the CORDIC architecture to calculate the root operation and inverse trigonometric function. The results are stored in the Gradient Magnitude Occurrence (GMO) buffer. The gradient intensity and gradient direction are analyzed by histogram statistics to obtain the main direction. To get the information on the main direction, the gradient direction in the GMO buffer is rotated and the angle of a trigonometric function is calculated by CORDIC architecture. The data in the GMO buffer is cut into 4×4 square blocks, and the histogram statistics are made for each block. Finally, the key-point descriptors are obtained by performing normalization to 128-dimensional data.

In hardware design, a lookup table is used to calculate trigonometric functions. Both of these operations are not suitable for hardware as they consume more resources and cause design frequency to be lower. However, the SIFT algorithm needs to calculate a large

Fig. 13 Matching result

number of trigonometric functions, inverse trigonometric functions, and root operations which makes CORDIC a good choice to perform trigonometric calculations. Figure 10 shows the hardware architecture of the CORDIC [11] We use a lookup table to store the values of arctan, x_0 and y_0 are input data, and z_0 is the initial angle set to zero. First, we determine the quadrant based on the value of x and y. Second, the angle is modified from the lookup table. Third, x_n and y_n are achieved by shifting x_{n-1} and y_{n-1}.

5 Experimental result

The dataset used for this work is the Affine Covariant Regions Datasets provided by the University of Oxford. The matching result of the dataset is shown in Fig. 11. It shows once the rotation of the test image is large, the number of key-point matching will be reduced. But in the change of light and shadow, our proposed work has good robustness.

Key-point matching is calculated through the Euclidean distance of two key-point vectors. The accuracy is evaluated by the match between the original image and the distorted image. For the point that matches successfully, it is checked whether the point's coordinate is the same or not. If two coordinates are the same then this case is marked as correctly the match point. The matching accuracy can be described as (6).

Table 2 Matching accuracy

	Software	Hardware
Original	97.2%	94.2%
Rotation	96.1%	91.6%
Scale	89.8%	85.9%
Luminance	94.7%	89.3%
ALL	87.9%	83.9%

Table 3 Comparison with other research

	[17]	[9]	[43]	[28]	[35]	[30]	Proposed
Technology	180 nm	90 nm	130 nm	130 nm	65 nm	180 nm	90 nm
Frequency	100 MHz	227 MHz	170 MHz	200 MHz	200 MHz	100 MHz	152 MHz
Gate Count	1320 K	704 K	790 K	555 K	942 K	108 K	642 K
Memory	5.729Mbit	553Kbit	396Kbit	–	–	9.25Mbit	237.4Kbit
FPS	30	30	36.8	30	–	–	35.6
Resolution	480p	1080p	720p	800*640	480p	512p	720p
Throughput (pixel/sec)	9.216 M	62.208 M	33.9148 M	15.36 M	–	–	32.8 M
Power	–	–	128.5mW	–	30.46mW	–	71.05mW
NHE[1]	13.96	88.36	62.01	39.97	–	–	51.09
NPE[2]	–	–	644.27	–	–	–	373.89

[1]NPE = $Throughput \bullet (Process/90)/ Power \bullet (1.0/Voltage)^2$

[2]NHE = $Throughput \bullet (Process/90)/GateCount$

$$\text{matching accuracy} = \frac{number\ of\ correct\ match\ points}{number\ of\ match\ points} \tag{6}$$

To test the stability of the algorithm, the stability test is conducted for image rotation, image scaling, and the change of light and shadow. Figure 12(a) shows the change in accuracy results due to rotation, where the original image rotated nine times with 10 degrees each time. The results show that this architecture has good invariance to the rotation. It can be seen that only a little matching percentage drops although the image is rotated to 90 degrees. Figure 12(b) shows the scale invariance test results. In this part, the original image is shrunk by 10% until the image becomes.

30% of the original image. The results show that as the scale reduction increased, the accuracy decreased significantly. Figure 12(c) shows the test result of the light and shadow invariance, which uses $y = \alpha x + \beta$ to adjust the contrast of the original image and the change of light and shadow. Figure 13 shows the matching result when all kinds of deformation are performed, and Table 2 shows the matching accuracy. The average accuracy in software and hardware is 87.9% and 83.9% respectively.

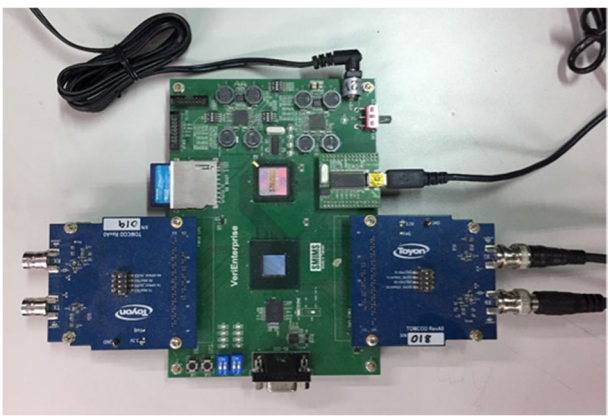


Fig. 14 VEXA7-200 development platform

Table 4 Utilization of FPGA resources

	Used	Available	Utilization
Slice Register	27435	267600	10.2%
Slice LUTs	40453	133800	30.2%
DSP	183	740	24.7%

The hardware architecture of the proposed algorithm has been implemented using TSMC 90 nm technology. The power consumption of the design is only 71.05 mW at the working frequency of 152 MHz. The result of the experiment shows that the proposed algorithm can achieve 35.6 frames per second (fps) for an image resolution of 1280×720 at the working frequency of 152 MHz.

Table 3 presents the comparisons of different parameters with related research work. For fair comparisons among different works, we also introduce two normalized metrics, normalized power efficiency (NPE) [18], and normalized hardware efficiency (NHE) [18]. In [17], all data produced during operations is stored in internal memory so more memory is required, and the data of gradient operation is also stored in int ernal memory. Consequently, this results in the requirement of huge memory to store GMO data. Compared to [17] and [28], we have a large lead in throughput. [9] proposed a layer parallel SIFT (LPSIFT) with an integral image. Its parallel hardware with an on-the-fly feature extraction flow can reduce computation complexity and achieve high throughput. They do not have good stability since the accuracy will decrease in case of a large change in angle and scale. Even [9] can achieve relatively higher throughput than us as an aid of a proprietary and reduced-computation SIFT, the memory usage is 2.33 times larger than ours. [43] worked on memory optimization and proposed the architecture with local-patch reuse and stored the Gaussian image in external memory. The throughput of [43] is slightly higher than ours, but the memory usage is 1.67 times larger than ours. [35] is designed based on 65 nm technology and with lower power consumption. However, the gate count is larger than ours. Compared to [30], we have a higher frequency and lower memory usage.

Additionally, we performed the design in FPGA to verify our proposed design. The proposed work has been synthesized and implemented on the Artix-7 XC7A200T chip on the VEXA7-200 platform as shown in Fig. 14, while the utilization of FPGA resources is shown in Table 4. The detailed result is listed in Table 5. As a comparison, it shows the number of look-up tables (LUTs) of the proposed design is lower than [27, 40, 41] and [26].

Table 5 Implementation result and comparison on FPGA

	[41]	[40]	[27]	[26]	Our
FPS	70	36	150	–	35.6
Resol-ution	480P	480p	480p	512p	720p
Plat-form	Cyclone IV	Cyclone IV	Cyclone IV	Vertex-7	Artix7-200
Slice Register	8372	24079	39482	–	27,435
Slice LUTs*	125644	138,944	65560	74358	40453
DSP	77	–	642	91	183

*One logic element has a four-input LUT for Cyclone IV

6 Conclusions

This paper presents a hardware implementation of SIFT algorithm to achieve real-time computation in high-resolution images and with the optimized structure to reduce memory requirements. We sincerely consider reducing the computation complexity and memory access. We construct the multiple parallel Gaussian filters and replace them with the parallel operation to detect the extreme values. In addition, we use a suitable scale to simulate the high-frequency space instead of amplifying the sampling to further reduce the memory requirement and the amount of subsequent Gaussian filtering. The proposed system has been implemented in TSMC 90 nm CMOS technology. Moreover, the design can run at 152 MHz with a memory usage of 237.4 Kbit only. We also make a comparison of ASIC design and FPGA results with the previous works. It shows the advantage of our work. The experimental results also show that the accuracy of the matching process is only affected by the scale, while the rotation and shadow changes still have good results. Therefore, we can study and modify the scale part to improve the accuracy in future perspectives.

Funding This work was supported by the Ministry of Science and Technology, Taiwan, under Grant MOST 111–2221-E-008 -089 -MY3.

Data availability The datasets analyzed during the current study are available from the corresponding author on reasonable request.

Declarations

Conflict of interests The authors declare no conflict of interest.

References

1. Acharya KA, VenkateshBabu R, Vadhiyar SS (2018) A real-time implementation of SIFT using GPU. J Real-Time Image Proc 14:267–277
2. Alhwarin F, Wang C, Ristic-Durrant D, Peter Graeser AG (2008) Improved SIFT-Features matching for object recognition", in VoCS'08 Proceedings of the 2008 international conference on Visions of Computer Science: BCS International Academic Conference, 179–190
3. Awad AI, Hassaballah M (2016) Image feature detectors and descriptors: foundations and applications. Stud Comput Intell 630(2016):11–45
4. Azad P, Asfour T, Dillmann R (2009) "Combining harris interest points and the SIFT descriptor for fast scale-invariant object recognition," in 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS 2009, 4275–4280
5. Banu VC, Costea IM, Nemtanu FC, Bădescu I (2017) Intelligent video surveillance system. In: 2017 IEEE 23rd International Symposium for Design and Technology in Electronic Packaging (SIITME), pp 208–212
6. Bay H, Tuytelaars T, VanGool L (2006) SURF:Speeded up robust features. In: Proceedings of the 9th European Conference onComputer Vision (ECCV), Graz, 7–13, 404–417
7. Bonato V, Marques E, Constantinides G (2008) A Parallel Hardware Architecture for Scale and Rotation Invariant Feature Detection. IEEE Trans Circuits Syst Video Technol 18(12):1703–1712
8. Canny J (1986) A computatuinal approach to edge detection. IEEE Trans Pattern Anal Mach Intell 8(6):679–698
9. Chiu L-C, Chang T-S, Chen J-Y, Chang N (2013) Fast sift design for real-time visual feature extraction. IEEE Trans Image Process 22(8):3158–3167
10. Daixian Z (2010) "SIFT algorithm analtsis and optimization," in 2010 international Conference on Image Analysis and Signal Processing, 415–419

11. Digital Circuits/CORDIC (2009) In Wikipedia, the free encyclopedia. From https://en.wikibooks.org/wiki/Digital_Circuits/CORDIC. Accessed Mar 2020
12. Doménech-Asensi G, Zapata-Perez J, Ruiz-Merino R, Lopez-Alcantud JA, Díaz-Madrid JÁ, Brea VM, López P (2020) All-hardware SIFT implementation for real-time VGA images feature extraction. J Real-Time Image Proc 17(2):371–382
13. Flitton G, Breckon T, Megherbi Bouallagu N (2010) Object recognition using 3D SIFT in complex CT volumes. In: Proceedings of the British Machine Vision Conference (BMVC), vol 1, pp 1–12
14. Fritz G, Seifert C, Kumar M, Paletta L (2005) Building detection from mobile imagery using informative SIFT descriptors. In: Image Analysis: 14th Scandinavian Conference, SCIA 2005, Joensuu, Finland, June 19-22, 2005. Proceedings 14. Springer Berlin Heidelberg, pp 629–638
15. Ha S-W, Moon Y-H (2011) Multiple object tracking using SIFT features and location matching. Int J Smart Home 5(4):17–26
16. Harris C, Stephens M (1988) A combined corner and edge dectector. In: Alvey vision conference, vol 15, no 50, pp 10–5244
17. Huang FC, Huang SY, Ker JW, Chen YC (2012) High-performance SIFT hardware accelerator for real-time image feature extraction. IEEE Trans Circuits Syst Video Technol 22(3):340–351
18. Huang M-Y, Tsai P-Y (2014) Toward multi-gigabit wireless: Design of high-throughput MIMO detectors with hardware-efficient architecture. IEEE Trans Circuits Syst I Reg Papers 61(2):613–624
19. Jain AK, Lee JE, Jin R, Gregg N (2009) Content-based image retrieval:An application to tattoo image," in Processings – International Conference on image Processing (ICIP), 2745–2748
20. Jiang J, Li X, Zhang G (2014) SIFT hardware implementation for real-time image feature extraction. IEEE Trans Circuits Syst Video Technol 24(7):1209–1220
21. Jinxia L, Yuehong Q (2011) Application of SIFT feature extraction algorithm on the image registration," in IEEE 2011 10th International Conference on Electronic Measurement & Instruments, 177–180
22. Karami E, Prasad S, Shehata M (2017) Image matching using SIFT, SURF, BRIEF and ORB: performance comparison for distorted images. arXiv preprint arXiv:1710.02726
23. Ke Y, Sukthankar R (2004) PCA-SIFT: a more distinctive representation for local image descriptors," in Proceedings of the 2004 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, 2004. CVPR 2004, 2: 506–513
24. Kreowsky P, Stabernack B (2021) A Full-Featured FPGA-Based Pipelined Architecture for SIFT Extraction. IEEE Access 9:128564–128573. https://doi.org/10.1109/ACCESS.2021.3104387
25. Kuffner A, Robles-Kelly A (2006) Image feature evaluation for contents-based image retrieval. ACM Int Conf Proceeding Ser 56:29–33
26. Kwon YH, Jeon JW (2021) Key point extraction hardware using sift algorithm in FPGA. In: 2021 IEEE Region 10 Symposium (TENSYMP). IEEE, pp 1–4
27. Li S-A, Wang W-Y, Pan W-Z, Hsu C-C, Lu C-K (2018) FPGA-based hardware design for scale-invariant feature transform. IEEE Access 6:43850–43864. https://doi.org/10.1109/ACCESS.2018.2863019
28. Lin Y-M, Yeh C-H, Yen S-H, Ma C-H, Chen P-Y, Jay Kuo C-C (2010) Efficient VLSI design for SIFT feature description. In: 2010 International Symposium on Next Generation Electronics (ISNE), pp 48–51
29. Lindeberg T (1994) Scale-space theory: A basic tool for analysing structures at diierent scales. J Appl Stat 21(2):225–270
30. Liu B et al. (2022) "An Energy-Efficient SIFT Based Feature Extraction Accelerator for High Frame-Rate Video Applications." IEEE Transactions on Circuits and Systems I: Regular Papers 69.12: 4930–4943
31. Lowe DG (2004) Distinctive image features from scale invariant keypoint. Int J Comput Vis 60(2):91–110
32. Mikolajczyk K, Schmid C (2004) Scale & affine invariant interest point detectors. Int J Comput Vis 60(1):63–86
33. Mikolajczyk K, Schmid C (2005) A performance evaluation of local descriptors. IEEE Trans Pattern Anal Mach Intell 27(10):1615–1630
34. Ouyang P, Yin S, Liu L, Zhang Y, Zhao W, Wei S (2018) A fast and power efficient hardware architecture for visual feature detection in Affine-SIFT. IEEE Trans Circuits Syst I Regul Pap 65(10):3362–3375
35. Qasaimeh M, Saleh H, Mohammad B, Tekeste T, Ismail M (2019) 'A novel SIFT architecture and ASIC implementation for real time SOC application.' Analog Integr Circuits Signal Process 99(2):325–338

36. Rosten E, Drummond T (2006) Machine learning for high-speed corner detection. In: Computer Vision–ECCV 2006: 9th European Conference on Computer Vision, Graz, May 7-13, 2006. Proceedings, Part I 9. Springer Berlin Heidelberg, pp 430–443
37. Sanchez HAL, George AD (2021) "A streaming hardware architecture for real-time SIFT feature extraction," 2021 International Conference on Field-Programmable Technology (ICFPT), 1-9, https://doi.org/10.1109/ICFPT52863.2021.9609932
38. Smith SM, Brady JM (1997) SUSAN—A new approach to low level image processing. Int J Comput Vis 23(1):45–78
39. Suzuki T, Ikenaga T (2012) "SIFT-based low complexity jeypoint extraction and its real-time hardware implementation for full-HD video," in Proceedings of The 2012 Asia Pacific Signal and Information Processing Association Annual Summit and Conference, pp.1–6
40. Vourvoulakis J, Kalomiros J, Lygouras J (2017) "A complete processor for SIFT feature matching in video sequences," IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS), pp. 95–100
41. Vourvoulankis J, Kalomiros J, Lygouras J (2016) Fully pipelined FPGA-based architecture for real-time SIFT extraction. Microprocess Microsyst 40:53–73
42. Wang J, She M, Nahavandi S, Kouzani A (2010) A review of vision-based gait recognition methods for human identification. Int Conf Digit Image Comput: Tech Appl 2010:320–327
43. Yum J, Lee CH, Kim JS, Lee HJ (2016) A novel hardware architecture with reduced internal memory for real-time extraction of SIFT in an HD video. IEEE Trans Circuits Syst Video Technol 26(10):1943–1954
44. Yum J, Lee C-H, Park J, Kim J-S, Lee H-J (2018) A Hardware Architecture for the Affine-Invariant Extension of SIFT. IEEE Trans Circuits Syst Video Technol 28(11):3251–3261
45. Zhou H, Yuan Y, Shi C (2009) Object tracking using SIFT features and mean shift. Comput Vis Image Underst 113(3):345–352

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Springer Nature or its licensor (e.g. a society or other partner) holds exclusive rights to this article under a publishing agreement with the author(s) or other rightsholder(s); author self-archiving of the accepted manuscript version of this article is solely governed by the terms of such publishing agreement and applicable law.