SOC for image processing using SIFT accelerator

Akshatha V Murthy
Department of Electronics and Communication Engineering
PES University,
Bangalore, Karnataka, India
Email: avila.murthy@gmail.com

B Rajeshwari
Department of Electronics and Communication Engineering
PES University,
Bangalore, Karnataka, India
Email: rajeshwari@pes.edu

Bajarangbali B
Department of Electronics and Communication Engineering
PES University,
Bangalore, Karnataka, India
Email: bajarangreddy@pes.edu

***Abstract*—In this paper, we have implemented System on Chip (SoC) for SIFT algorithm by integrating designed and library IPs. In this algorithm, we have verified for feature transforms like rotation, intensity increase and decrease. The key points are detected for the given image and compared with that of transformed images. The key point matching achieved is up to 99.8%. It is also verified that Scale Invariant Feature Transform (SIFT) algorithm works for any image format. The algorithm is implemented in Xilinx model composer and then it is converted to Xilinx system generator block for Field Programable Gate Arrays (FPGA) implementation. FPGA used is Zynq-7000 xc7z020clg484-1.**

Keywords— SIFT, IP integration, key points detection, FPGA, Xilinx

I. Introduction

In image processing, the main and important step is to find the features of the test image and use this for object detection. These features should be stable i.e. they should be invariant to position of the object, illumination and scale. In these conditions, SIFT suits better.

SIFT algorithm was introduced by D. Lowe [15]. Algorithms like Speed Up Robust Features (SURF) and fast SIFT were introduced but performance speed achieved was less for real time application. Graphic Processing Unit (GPU) based systems using SIFT algorithm was introduced but they required more hardware and consumed too much power. Many hardware designs were introduced for reducing the hardware [1]. Authors proposed hardware accelerator architecture for SIFT which consists of key point identification and feature descriptor generation for Video Graphics Array (VGA) based images but, they claim that this method is used for key feature extraction of less than 890 points. This algorithm is used in our hardware implementation and achieved higher number of key point detection. [2] introduces Principle Component Analysis (PCA) SIFT where authors have reduced the calculation cost by offloading hardware with gaussian kernel calculation. [16] introduces pipelined architecture for SIFT algorithm. They claim that this architecture can be applied for real time and is effective. [17] introduces an algorithm for key point matching between images which is implemented in our paper. In [18], authors have implemented object recognition with invariance to the rotation, scale changes and illumination changes. There had been many algorithms introduced previously for image processing and finding features, but SIFT algorithm is proved to be the suitable among the ones existing because, as the name suggests, it is invariant to scale changes, rotation and illumination. This would form the very important aspect when it comes for image processing.

In this paper, we have implemented high performance hardware by calculating different sigma and gaussian kernels on dedicated hardware unlike [2]. IPs are designed and integrated according to the [1]. Hardware IPs are generated for three octaves, (i.e. DoG (Difference of Gaussians) and gaussian filters), key point detection, magnitude, orientation calculation and feature descriptors using Xilinx model composer and are converted to Xilinx system generator to implement on Zynq FPGA. Before this work, SIFT was never implemented using Xilinx model composer as it is the newly introduced topic.

II. System Architecture

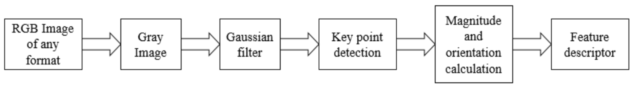

Fig. 1. SIFT block diagram

A. Gaussian and DoG pyramid

Fig. 1 shows SIFT block diagram. The input image is in RGB format. Its size is 512 x 512. It is converted to gray scale image before applying to gaussian filter using (1).

$$(0.2989 \times R) + (0.5870 \times G) + (0.1140 \times B) \qquad (1)$$

The gray image is filtered using gaussian filter to blur edges and reduce contrast. We convolve the image by gaussian filter kernel. Equation (2) shows the equation for gaussian kernel calculation and (3) shows equation for image convolution where I(i, j) denotes the image, G(x, y, σ) denotes the gaussian kernel, L(i, j, σ) denotes the gaussian filtered image, k is the multiplicative factor which is √2 and σ is the variable scaling factor which gives the amount of blur given by (4), where, k_1 gives the number of images per octave and k_2 gives the number of octaves. Octaves are a set of gaussian images or gaussian pyramid. There are 3 octaves and 4 images per octave in our algorithm, hence k_1 varies from 0 to 3 and k_2 varies from 0 to 2. For 2nd octave, the size of the original image is reduced by half and then gaussian filtering is done on this image. Similarly, for 3rd octave, the size of 2nd octave images are reduced by half. DoG images are obtained from gaussian images by subtracting corresponding images in the gaussian pyramid using (5), where D (i, j, σ) denotes the DoG image.

$$G(x.\, y, \sigma) = \frac{1}{2\pi(k\sigma)^2} \times e^{\frac{-(x^2+y^2)}{2(k\sigma)^2}} \qquad (2)$$

$$L(i, j, \sigma) = I(i, j) * G(x, y, \sigma) \qquad (3)$$

$$\sigma = (k)^{k_1 + 2k_2} \times 1.6 \qquad (4)$$

$$D(i, j, \sigma) = L(i, j, k\sigma) - L(i, j, \sigma) \qquad (5)$$

B. *Key point detection*

Three DoG images obtained in each octave are used to detect key points [4]. Every pixel in the DoG image is compared with the 8 pixels around it, 9 corresponding pixels in above DoG image and 9 corresponding pixels in below DoG, i.e. each pixel is compared with 26 (8 + 9 + 9) pixels. This is marked as key point if it is minima or maxima among these 26 pixels.

C. *Calculation of magnitude and orientation for histogram*

Magnitude and orientation are calculated for all pixels. This step retains only those key points which are invariant to location, scale and rotation. The gradient magnitude and orientation are calculated using (6) and (7) respectively on the gaussian filtered images, where m (i, j) is the magnitude and θ (i, j) is the orientation.

$$m(i, j) = \sqrt{[L(i+1, j) - L(i-1, j)]^2 + [L(i, j+1) - L(i, j-1)]^2} \quad (6)$$

$$\theta(i, j) = \tan^{-1}\left[\frac{L(i, j+1) - L(i, j-1)}{L(i+1, j) - L(i-1, j)}\right] \quad (7)$$

The 360 degree of a circle is divided into 36 bins where each bin covers 10 degrees. Here, bins represent intensity (0 to 255). The gradient magnitudes are assigned to these bins and the result is called gradient histogram of orientation.

D. *Generation of feature descriptors*

For each key point, descriptor is calculated by dividing the region around every key point into 16 x 16 region which is in turn divided into 4 x 4 sub regions, hence there are 16 histograms for each key point. Now, the 360 degree of the circle is divided into 8 bins with each bin covering 45 degrees. Here, bins represent angles. The gradient histogram of orientation is calculated and assigned for each sub region. Hence, there are 16 x 8 = 128 values for each key point. This provides invariance to illumination.

III. IMPLEMENTATION

A. *Gaussian filter implementation*

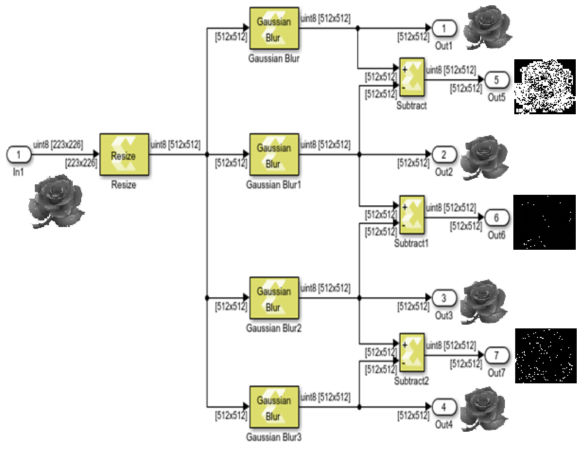

Fig. 2. IP integration of an octave containing three DoGs

The input image is fed to gaussian filter block which consists of multiplier, adder and shifter. The gaussian kernels are also calculated on FPGA. The series of gaussian filtering is called octave. The value of sigma used is 1.6 and value of constant k is 2. The model is implemented in Xilinx model composer. Fig. 2 shows octave implementation of first octave. Other two octaves are implemented similarly but the only difference is the image size is reduced by half for second octave and the image size of second octave is reduced by half for third octave.

B. *Key point detection implementation*

Fig. 3 shows model composer implementation for key point detection, where Min block gives minimum and Max block gives maximum of 3 DoG comparisons among 26 pixels as explained in section II B.

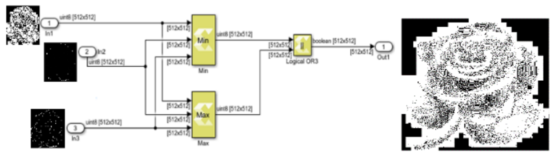

Fig. 3. IP integration of key point detection

C. *Calculation of magnitude and orientation implementation*

Fig. 4 shows model composer implementation for calculation of magnitude and orientation.

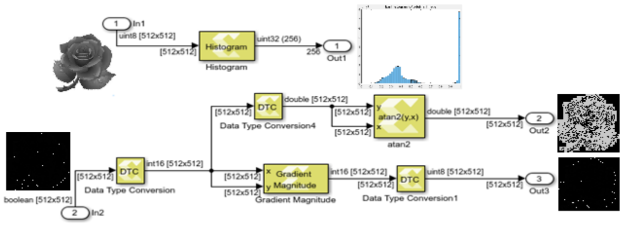

Fig. 4. IP integration of Histogram, Calculation of magnitude and orientation implementation

D. *Calculation of magnitude and orientation implementation*

Fig. 5 shows model composer implementation for Feature descriptor.

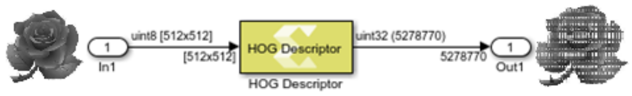

Fig. 5. IP integration of HOG (Histogram of gradients) descriptor

E. *IP integration*

Fig. 7 shows the model composer IP integration of three octaves, key point detector, magnitude and orientation calculation and feature descriptor. Fig. 8 shows Xilinx system generator IP integration for the same.

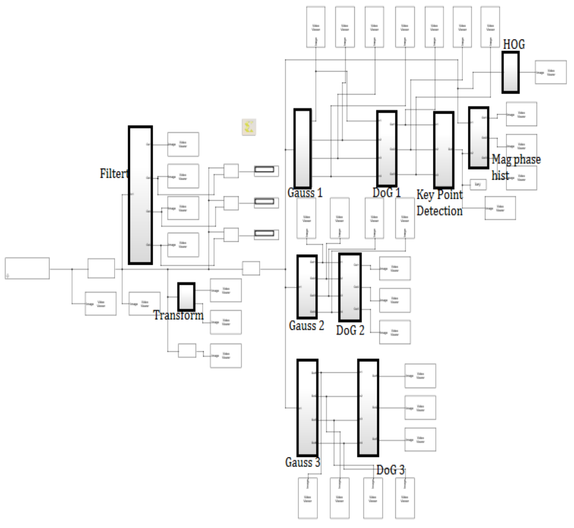

Fig. 6. SIFT IP integration

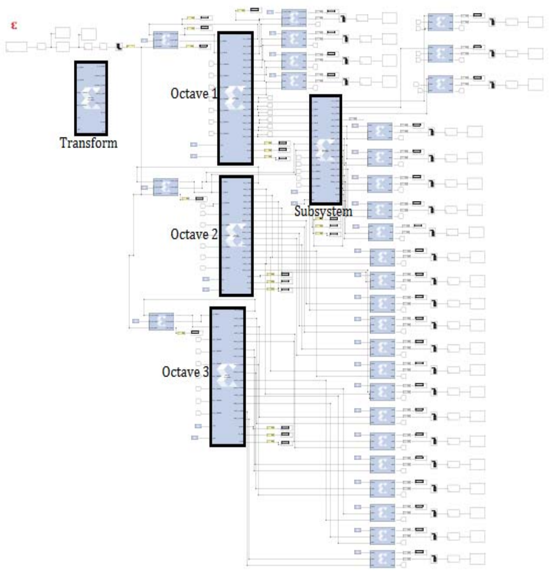

Fig. 7. Xilinx system generator IP integration

IV. RESULTS AND ANALYSIS

To prove the invariance of SIFT algorithm, some transformations like rotation and intensity variations are applied on the image and whole SIFT algorithm is applied on them. Fig. 9 shows the model composer model for transform block. Key point matching is done by comparing corresponding key points at a location obtained on the original image and that of transformed image. If there is a match, the pixel is made as 1, otherwise 0. TABLE I shows the key point matching for all the three transformations.

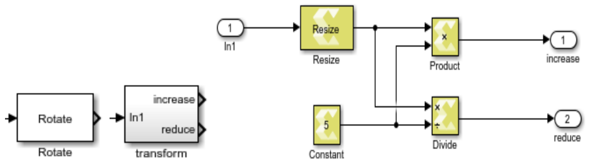

Fig. 8. IP integration of Transform block

SIFT algorithm can be applied to various image formats as shown in TABLE II. Run time analysis of SIFT algorithm is compared with [1] as shown in TABLE III. Number of key points detected is compared with previous works as shown in TABLE IV.

TABLE I. KEY POINT MATCHING

Parameters	Original image	Intensity reduced	Intensity increased	Rotation
Time for finding key points (seconds)	0.84	0.509	0.805	0.839
Total number of key points	1082	234	2890	1091
Total number of unmatched key points	-	5	17	6
Percentage unmatch in key points (%)	-	0.4621	1.5712	0.5545
Percentage match in key points (%)	-	99.5379	98.4288	99.4455

TABLE II. COMPARISON WITH VARIOUS KINDS OF IMAGES

Image format	Image size	Number of key points	Time for key point detection (sec)
jpg	223 x 226	1082	0.808
png	640 x 445	2132	0.856
bmp	225 x 225	1338	0.694
tif	251 x 201	2411	0.913

TABLE III. TIME COMPARISON ON DIFFERENT FPGA BOARD

Parameters	Our Implementation		[1]
	Zynq – 7000 (sec)	*Kintex plus (sec)*	***DE2-70 FPGA BOARD NIOSII CPU***
Gaussian pyramid construction	19.014	13.125	2278.27
DoG space construction	13.125	11.724	3.62
Key point detection	4.350	3.305	9.08
Gradient and orientation assignment	6.562	4.271	57.56
Descriptors generation	6.697	4.289	311.80
Entire SIFT operation	**49.748**	**36.714**	**2660.66**

TABLE IV. COMPARISON OF NUMBER OF KEY POINTS DETECTED

Reference	Time (sec)	Key points	Image size	FPGA
[1]	2.87	890	640 x 480	NiosII CPU on DE2-70 FPGA Board
[3]	87.37	1523	1920 x 1080	Zynq-7000 AP SoC ZC702
Proposed	0.84	1082	512 x 512	Zynq-7000 xc7z020clg484-1

V. CONCLUSION

SIFT implemented on FPGA using model composer, improved the performance as the time consumed in our implementation is approximately about 2% of that of the method used by [1] as shown in TABLE III. Also, including kernels whole SIFT is implemented on FPGA unlike [2]. Hence, speed up is achieved. Key point matched for variance with rotation is 99.4455, intensity reduced is 99.5379 and intensity increased is 98.4288 as shown in TABLE I. SIFT can be applied to any image formats. More number of key points are detected in the proposed work as shown in TABLE IV.

REFERENCES

[1] Feng-Cheng Huang, Shi-Yu Huang, Ji-Wei Ker and Yung-Chang Chen, “High-Performance SIFT Hardware Accelerator for Real-Time Image Feature Extraction”, IEEE transactions on circuits and systems for video technology, vol. 22, no. 3, march 2012.

[2] Shih-An Li, Wei-Yen Wang, Wei-Zheng Pan, Chen-Chien Hsu, Cheng-Kai Lu, “FPGA-Based Hardware Design for Scale- Invariant Feature Transform”, IEEE Access, 2017.

[3] J.Q. Peng, Y.H. Liu, C.Y. Lyu, Y.H. Li, W.G. Zhou, K. Fan, “FPGA-Based Parallel Hardware Architecture for SIFT Algorithm”, Proceedings of the 2016 IEEE International Conference on Real-time Computing and Robotics, June 6-9, 2016, Angkor Wat, Cambodia.

[4] Leonardo Chang, Jose Hernandez-Palancar, L. Enrique Sucar, Miguel Arias-Estrada, “FPGA-based detection of SIFT interest keypoints”, Springer-Verlag, 30 May 2012.

[5] Irene Amerini, Lamberto Ballan, Roberto Caldelli, Alberto Del Bimbo, Giuseppe Serra, “A SIFT-Based Forensic Method for Copy–Move Attack Detection and Transformation Recovery”, IEEE transactions on information forensics and security, VOL. 6, NO. 3, September 2011.

[6] “Chapter 6: SIFT (Scale Invariant Feature Transform) based Face Recognition”.

[7] “Model Composer User Guide - Vivado Design Suite”, UG1262 (v2018.3) December 5, 2018.

[8] “Tutorial: Model-Based Design Using Model Composer”, UG1259 (v2018.3) December 5, 2018.

[9] Shawki Areibi, “Tutorial - Using Xilinx System Generator 14.6 for Co-Simulation on Digilent NEXYS3 (Spartan-6) Board”, August 15, 2017.

[10] Ives Rey Otero, “Anatomy of the SIFT method”, Dissertation, Submitted on 9 Nov 2015.

[11] Emmett Kuhn, “Grayscale Conversion of a Color Image Using Simulink and Xilinx Blocks”, An Application Note, November 19, 2010.

[12] Tom Ganley, “FPGA Co-Simulation of Gaussian Filter Algorithm”, Application Note, November 19, 2010.

[13] “System Generator for DSP - Reference Guide”, UG638 (v11.4) December 2, 2009.

[14] Gi-woong Shin, Jong-tae Sung, Young-Hyoung Kim and Yong-hwan Lee, “Hardware Design of Feature Point Extraction using SIFT Algorithm “, Advanced Science and Technology Letters, CIA 2014.

[15] David G. Lowe, “Distinctive Image Features from Scale-Invariant Keypoints”, 2004.

[16] Chenyu Xu, En Zhu, Tiange Su, Xiaoran Li, “An Optimized Structure on FPGA of Key Point Detection in SIFT Algorithm”, MATEC Web of Conferences 56, 02003 (2016).

[17] Rajkumar N.Satare, Prof. S. R. Khot, “Image matching with SIFT feature”, Proceedings of the Second International Conference on Inventive Systems and Control (ICISC 2018).

[18] Shivakanth, Archana Mane, “Object Recognition using SIFT”, IJISET - International Journal of Innovative Science, Engineering & Technology, Vol. 1 Issue 4, June 2014.

[19] K. Anusudha, G. Bharadwaja Reddy, “Implementation of Image Edge Detection on FPGA using XSG”, 2016 International Conference on Circuit, Power and Computing Technologies [ICCPCT].

[20] Dovari. Swaraj, Dr.G.L.Madhumati, “FPGA Implementation of SIFT Algorithm Using Xilinx System Generator”, International Journal of Emerging Trends in Electrical and Electronics, Vol. 10, Issue. 10, Oct. 2014.

[21] Syama K Nair, Ragimol, “Modified SIFT Algorithm for Image Feature Detection”, International Journal of Science and Research (IJSR), Volume 5 Issue 7, July 2016.

[22] Mohammed Abdulhalim Alareqi, Mateur Khalid, Rachid Elgouri, “Real Time Hardware Co-Simulation for Image Processing Algorithms Using Xilinx System Generator”, Article in International Journal on Electrical Engineering and Informatics · December 2015.

[23] Deepesh Prakash Guragain, Pramod Ghimire, Kapil Budhathoki, “Implementation of FPGA Based Image Processing Algorithm Using Xilinx System Generator”, International Research Journal of Engineering and Technology (IRJET), Volume: 05, Issue: 01, Jan-2018.