IOSR Journal of Computer Engineering (IOSR-JCE)
e-ISSN: 2278-0661,p-ISSN: 2278-8727
PP 33-38
www.iosrjournals.org

Faster Image Feature Extraction Hardware

Jibu J.V, Sherin Das, Mini Kumari G
Assistant Professor,College of Engineering, Chengannur.Alappuzha, India. jvjibu77@gmail.com
HSST in Computer Science, Department of Higher Secondary Education.Govt of kerala, India
Assistant Professor,College of Engineering, kottarakkara, Kollam, India.

Abstract: *Scale Invariant Feature (SIFT) has many applications in different fields especially in computer vision. Several improved version of SIFT were proposed after David Lowe proposed" Distinctive Image Features from Scale Invariant key points "in 2004[1].Many people tried to improve sift features by adding some features to original SIFT.SURF, n-dimensional SIFT, CSIFT,GSIFT are some of them. One of the major constraint of SIFT in real time is its computational delay. Because of its computational complexity and large time delay its hardware implementation is a challenging task .Several methods were proposed to improve the speed of SIFT. In this paper we are proposing an architecture to improve the speed of key point identification part of sift algorithm. Here we have selected the architecture proposed by Li fan Yao& Hao Feng in 2009[2]. Using our proposed architecture the speed of the feature detection part has improved to 1 ms from 2.7ms when implemented in Altium Nano Board 3000 operating at 20 MHz clock.*

Key words*: Sift, FPGA, Feature Detection, Image Matching,Computer Vision.*

I. Introduction

Different feature detection algorithms always has attraction in the field of computer vision. One of the major constraints of these feature detection algorithms is the computational complexity and huge demand of memory. Time delay is very much higher in finding the descriptor which restricts them in real time applications .Earlier Harris Corner detector is used mainly in image identification .But it is susceptible to changes in image scale. So when David Lowe proposed" Distinctive Image Features from Scale Invariant key points "in 2004,it becomes the centre of attraction in image matching problems. Mainly SIFT algorithm contains two parts ,Key point identification part and Descriptor generation part. key point identification part is relatively simple and fast .But descriptor generation is computationally intensive and require large memory. So researchers have taken interest in this part and several methods have proposed from 2008 to till date. The result is time to implement the SIFT algorithm has improved from nearly 30 ms to 6 ms.

II. ISift Algorithm

The SIFT algorithm consist of four parts .the first one is the formation of Difference of Gaussian Pyramid. Second step is identification of key points. Then we have to calculate the gradient magnitude and Orientation. Finally we have to Generate the SIFT descriptor.

A. Formation of Difference of Gaussian Pyramid.

The first stage of the SIFT algorithm is to construct the Gaussian pyramid. First, we convolve the given source image, denoted as I(x,y), with a Gaussian functionG(x,y,σ), the result is a Gaussian-blurred image, denoted as L(x,y,σ), as shown in Fig(1). The DoG image, denoted as D(x,y,σ), is defined as the difference between two adjacent Gaussian-blurred images, as shown in Fig (2) and is given by

$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$ .....(1)

$D(x, y, \sigma) = L(x, y, k\sigma) - L(x, y, \sigma)$.....(2)

whereG is the Gaussian kernel with a scale of σ is shown as

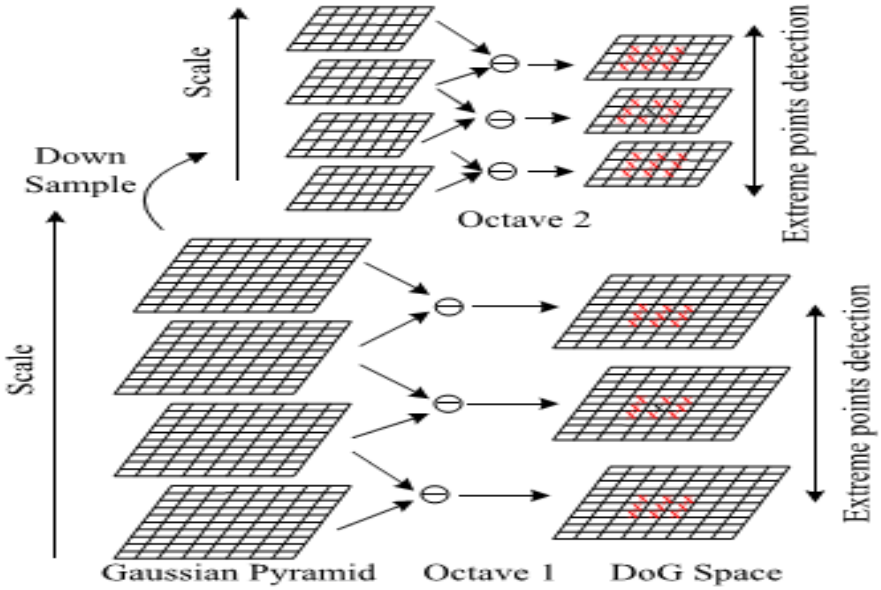
$G(x, y, \sigma) = (1/2\pi\sigma^2)e^{-((x2+y2)/2\sigma2)}$

Which is equivalent to

$$DOG == (k-1)\sigma^2\nabla^2 G \quad \text{.....(3)}$$

The convolution of input image I(x,y) with variable scale Gaussian kernel produces a space of blurred images with amount of blur depends on scale factor (σ).In this way a stack of Gaussian blurred images are produced which constitute one octave. A Gaussian Pyramid is formed by multiple octaves with different image sizes. that is for each successive octave the image is down sampled. no of octaves and scales effects the robustness and complexity. Thus the octave formation is a sequential operation. That is the next scale in each

octave is formed only after the completion of the previous smoothing operation by the Gaussian function. So it is a time consuming part.The process is explained in Fig(1).



Fig(1) Gaussian pyramid formation[1]

For each octave of scale space, the initial image is repeatedly convolved with Gaussians to produce the set of scale space images shown on the left. Adjacent Gaussian images are subtracted to produce the difference-of-Gaussian images on the right. After each octave, the Gaussian image is down-sampled by a factor of 2, and the process repeated. In addition, the difference-of-Gaussian function provides a close approximation to the scale-normalized Laplacian of Gaussian, $\sigma^2\nabla^2 G$, It can be showed that the normalization of the Laplacian with the factor σ^2 is required for true scale invariance. In detailed experimental comparisons, we can prove that the maxima and minima of $\sigma^2\nabla^2 G$ produce the most stable image features compared to a range of other possible image functions, such as the gradient, Hessian, or Harris corner function.

B. Identification of key point.

The maxima and minima of the Difference of Gaussian images are detected by comparing a pixel (marked with X) to its 26 neighbors as shown in fig(2) .This approach gives rise to a lot of Keypoints depending on the image size. Often quality of keypoints is more important than the quantity for reliable object recognition. For ensuring the stability of keypoints,the local extrema are compared with a minimum threshold value. The extrema that have relatively higher minima or maxima value pass the threshold test and are considered, while the weak keypoints having low contrast are discarded, resulting in less but more stable candidates

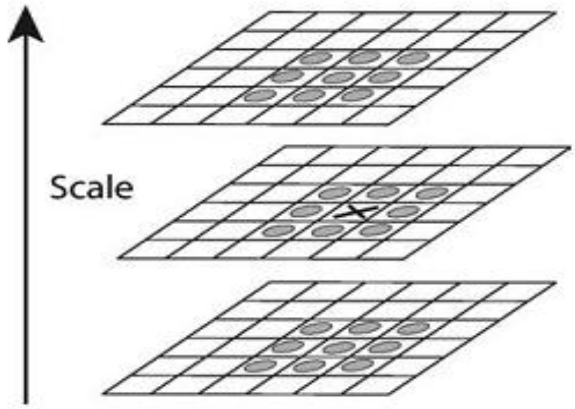


Fig (2) finding feature points[1]

C. Calculate the gradient magnitude and orientation

The magnitude-orientation histogram is used to describe a feature point, which is computed from the gradient magnitude and orientation of the neighbor pixels around the candidate feature point. For a given pixel (x,y) the gradient magnitude and orientation are computed as

$m(x,y)=((L(x+1,y)-L(x-1,y))2+(L(x,y+1)-L(x,y-1))2)1/2$.....(4)

$\Theta(x,y)=\tan-1((L(x,y+1)-L(x,y-1)/(L(x+1,y)-L(x-1,y)))$ .....(5)

Orientation Histogram Generation

The gradient orientation of a circular local region is shown below .The arrows represent the orientation direction and its length represents the gradient magnitude of all the pixels with same orientation. The histogram is represented in 2D coordinate as shown in fig(3).

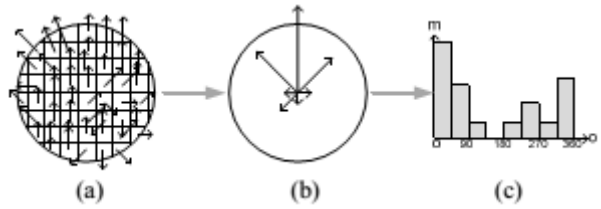


Fig (3) gradient-orientation histogram[2]

Descriptor Generation

Here a different method is used to find the descriptor as shown in fig(4).

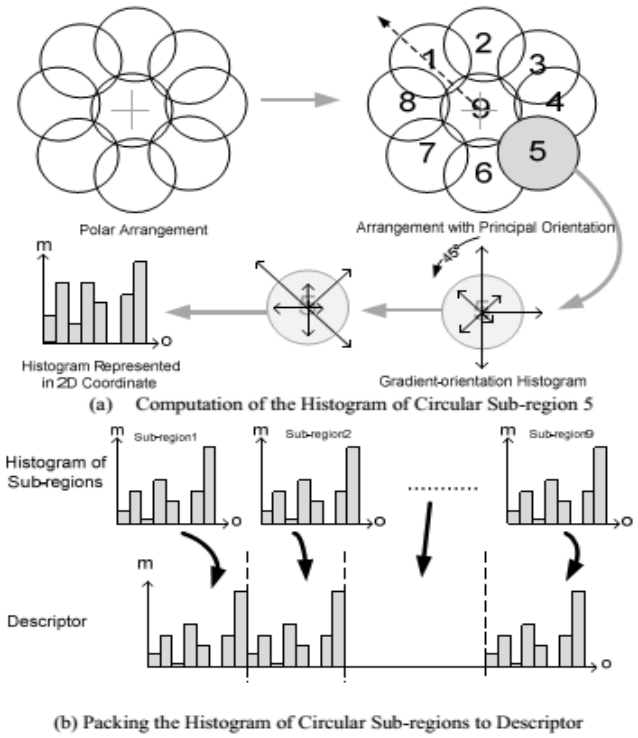


Fig (4) descriptor formation method[2]

The Fig(4) shows the process of generating a feature descriptor as per [2] . Firstly, with polar arrangement, total 9 circular sub-regions of a feature point as shown in Fig (4). are identified, in which the cross in the central sub-region indicates the location of the feature point. Each circular sub region has a diameter of 20 pixels and the overlapping of sub regions is designed for robustness. Secondly, a gradient orientation histogram of all 9 circular sub-regions is built up and then the principle orientation can be identified. Assume that the principle orientation is of the up-left direction pointed by the arrow with the broken line in Fig (4). Each of the 9 circular sub-regions is then assigned a number from 1 to 9 starting from the circular sub-region pointed by the principle orientation, following up with others in a clockwise fashion and ending up with the sub-region in the centre. Thirdly, the gradient-orientation histogram of each circular sub-region is generated and then it is re-ordered with the bin pointed by the principle orientation in the first place in the clockwise fashion. Finally, all

re-ordered histograms are packed together with the order of the circular sub-region number as shown in Fig (4).So the descriptor will be of size 72 instead of 128 as per [3].

III. Proposed Architecture

SIFT descriptor generation is really a time consuming operation which limits its real time application. For real time application it should not take more than 30 ms for 30 frames/sec video The scope of this paper is to implement part of the SIFT software algorithm in hardware on a FPGA. The designed architecture should comprise dedicated hardware units for the computationally intensive part of the SIFT algorithm. The design is synthesizable, adaptable, and generic and demand few FPGA resources. It may be adaptable to different number of octave and scales. One important question in video processing is to decide which algorithm parts should run on software processors and which ones should be implemented in hardware, for instance FPGA or ASIC. The input images are provided by a camera with a jpg format 320X240 with an 8-bit grey scale image and a hardware module which stores the images from the camera on the external RAM. In order to access and store the input images and output data, a pre-developed cache hardware module will be utilized. The output data will be stored on the external RAM memory connected with the FPGA using a multi-port memory controller IC provided by Xilinx.

The main aim of this thesis is to propose an architecture for finding a SIFT descriptor so that it can be used for real time application. The hardware implementation of SIFT algorithm includes four different blocks .The first block form the Scale space and calculate the DoG. The other two blocks run in parallel, to speed up the process. Key point detection part and gradient magnitude and orientation part run in parallel. Next block is the computationally intensive part of SIFT algorithm " Descriptor Generation" part.

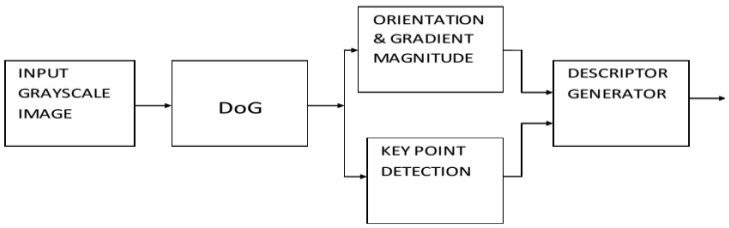
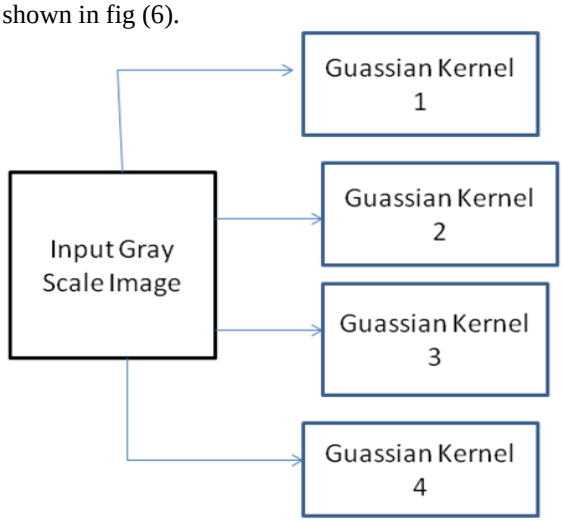


Fig (5) architecture of sift descriptor formation

In this paper we are using two images .First one is the TRAINING image fig(7)and the second one is called TEST image fig(8) .Both of them are gray scale images. Both the images are resized to 320x240 pixels so as to use in the The Altium NanoBoard3000 FPGA. The Gaussian pyramid formation is a sequential task . The convolution of image with the variable scale Gaussian kernel produces a stack of blurred images with the amount of blur dependent on scale factor. First the image is smoothed by the Gaussian Kernel with $\sigma = \sigma 1$,to produce the next element of the pyramid. this is again smoothed by another kernel with $\sigma = \sigma 2$ and so on.The same thing can be obtained by smoothing the original image by a filter with $\sigma = \sigma 1 + \sigma 2$.

Simillarly the second octave is formed by down sampling the first octave .This operation can be implemented directly from the original image by replacing σ with $\sigma/2$.In this way we can directly form the two octaves of gaussian space as shown in fig (6).

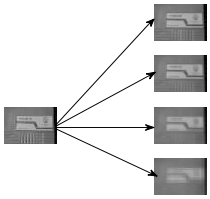


fig(6) gaussian pyramid formation

In this way we can improve the speed of operation of Gaussian space formation at the cost of increased hardware. That is, in the original architecture only one Gaussian kernel is needed but here 8 Gaussian kernel is needed or the value of σ's needed to store is high.But the speed of operation increases .

IV. Result

The training image has 564 no of key points where as test image has 630 feature points .Simulation of SIFT algorithm is done with MATLAB R2012. These results were taken as a reference for the hardware implementation. The simulation result is given below It takes 40ms to run this simulation in MATLAB. Fig(7) shows pyramid formed by training image. All the stages are formed directly from the given image. Similarly fig (8) shows pyramid formed by test image. Fig(9) & Fig(10) shows the SIFT key points obtained for the training and test image. Fig(11) shows the matching of the key points in both images. Fig(12) shows the pyramid formed in the Altium Nano Board 3000. Table 1 gives the FPGA hardware utilization details of original and our architecture. Even though FPGA hardware utilization is large in our architecture its speed of operation has improved to 1 ms from 2.7ms when implemented in Altium Nano Board 3000 operating at 20 Mhz clock.



Fig(7) GaussianPyramid of training Image

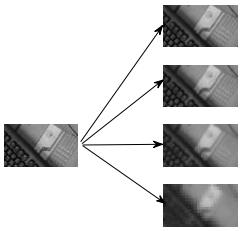
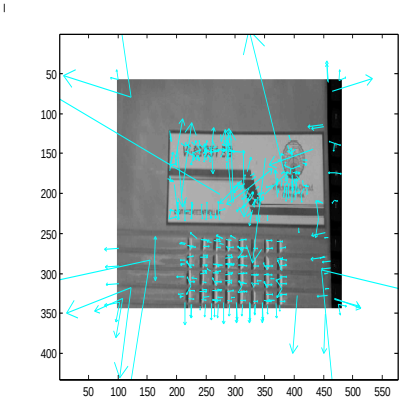
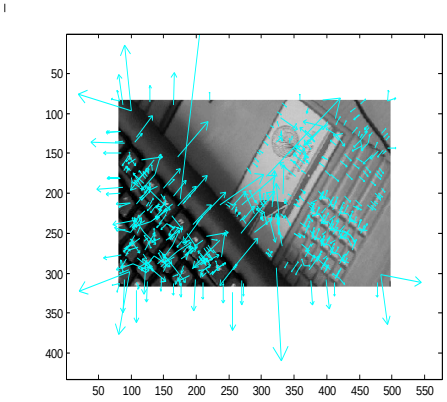


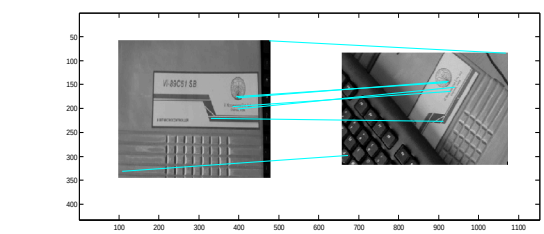
Fig (8) GaussianPyramid of test Image



Fig(9)feature points in training image



Fig(10) feature points in test image



fig(11) image matching

fig (12) fpga output of 1st stage of pyramid

Table I FPGA utilization

Parameter	Value as per original architecture	Value as per our architecture
Number of Slices	4400	7400
Number of Slice flipflops	2079	6028
Number of 4 input LUTs	4578	9746
Number of 2 input LUTs	4600	9852
Number used as RAM	198	340
Number used as ROM	40	78

V. Conclusion

The SIFT algorithm is invariant to image scale and rotation, and are shown to provide robust matching across a substantial range of affine distortion, change in 3D viewpoint, addition of noise, and change in illumination. The features are highly distinctive, in the sense that a single feature can be correctly matched with high probability against a large database of features from many images. It has several application in the following areas. Robot navigation, Panoramic photograph, Object detection and recognition, Sports and Wild life Censes. By this hard ware architecture the speed of operation for the scale space generation is improved from 2.7 ms to 1ms.

References

[1]. Jie Jiang, Xiaoyang Li, and Guangjun SIFT Hardware Implementation for Real-Time Image Feature Extraction IEEE transactions on circuits and systems for video technology, vol. 24, no. 7, july 2014

[2]. Lifan Yao , Hao Feng , Yiqun Zhu , Zhiguo Jiang, Danpei Zhao, Wenquan Feng.An Architecture of Optimised SIFT Feature Detection for an FPGA Implementation of an Image Matcher. 978-1-4244-4377-2/09/ 2009 IEEE .

[3]. D. G. Lowe, "Distinctive image features from scale-invariant key points," Int. J. Comput. Vis., vol. 60, no. 2, pp. 91–110, Jan. 2004.