International Journal Of Electronics Engineering (Issn: 0973-7383)
Volume 10 • Issue 2 pp. 233-240 June 2018-Dec 2018 www.csjournals.com

FPGA Implementation of Integral Image generator in SURF detector

Y. Mamillu[1], Y. Sri Chakrapani[2] , Dr. M. Kamaraju[3]
[1] M.Tech student, [2] Assoc Prof., [3] Professor,
Dept. of E.C.E , Gudlavalleru Engineering College, Gudlavalleru , India,
mamillu.yaragani4a6@gmail.com, srichakrapani@gmail.com, madduraju@yahoo.com

Abstract: Integral image generation improves the speed by reducing no. of computations like additions and multiplications. in computer vision applications such as image feature detectors, There are different algorithms for image feature detection, such as SURF, SIFT, HOG, Harris-Laplace Feature detection, FAST etc. Integral Image generation is used in SURF detector, which detects salient points from image and computes descriptors of their surroundings that are invariant to scale, rotation and illumination changes, hence it can be used in many of applications. The proposed Integral image generator in SURF detector uses Recursive addition equations for 320x240 image. Which improves the speed and reduce the Hard ware.. This Integral Image Generator is implemented in Virtex7 FPGA using verilog HDL.

Keywords: FAST ,Feature, HOG, SIFT, SURF.

I. Introduction

Feature detection is a basic low level image processing operation. It is usually performed as the first operation on an image, and examines every pixel to see if there is a feature present at that pixel.

Many computer vision algorithms uses feature points detection as a initial step, there are different feature detectors have been developed, Different algorithms are proposed for feature point detection Wenjie Chent.al [1] proposed parallel and pipeline architecture for image feature detection. Stephens [2] proposed a second moment matrix method for corner and edge detection that provides rotational invariance but lacks scale invariance. T.Lundeber [3] proposed the conception of automatic scale selection to extract scale-invariant feature points. He uses the Hessian matrix determinant and the Laplace operator to detect block like feature points. Schmid [4] proposed Harris Laplace and Hessian Laplace detectors to improve the automatic scale selection method for feature point detection. Lowe [5] proposed the SIFT algorithm used to find the maxima in the scale space and extracting the location information of the feature points.

To develop the SIFT algorithm, Gool [6, 7] proposed a feature point detection, description and extraction algorithm, Speeded up robust features(SURF). SURF algorithm reduces the number of feature points detection and the length of the feature point's descriptor, which significantly improves the speed of image feature detection, matching and extraction. The main applications of SURF algorithm is target tracking, image mosaics, video segmentation, intelligent transportation, and image registration.

FPGA based image features Detection algorithm improves the speed of image features Detection, Cornelis [9] proposed FPGA based real time image feature detection and matching system based on SIFT algorithm improve the speed. Pedre [10] proposed a FPGA based image fearure extraction system using SURF algorithm. Oruklu [11] proposed a FPGA based traffic sign detection system. Hofstee [12] proposed FPGA base image feature detection system for image analytics. It uses a Harris Laplace variant of scale invariant feature detection.

In SURF detector Integral image generation is essential component. Integral image generation improves execution speed for computing box filters. Employment of the integral image eliminates computationally expensive multiplications for box filter calculation, reducing it to three addition operations. This allows all box filters to be computed at a constant speed, irrespective of their size.

II. SURF algorithm

The SURF workflow includes feature point detection, feature point description and matching. In this paper, we focus on the feature point detection. The block diagram is shown in fig1, more details could be found on [5].

Feature point detection

There are four steps for feature point detection: (S1) integral image generation, (S2) calculating the approximated determinant of the Hessian matrix, (S3) scale space construction, and (S4) localization of feature point by detection the maxima from the blob response map.

International Journal Of Electronics Engineering (Issn: 0973-7383)
Volume 10 • Issue 2 pp. 233-240 June 2018-Dec 2018 www.csjournals.com

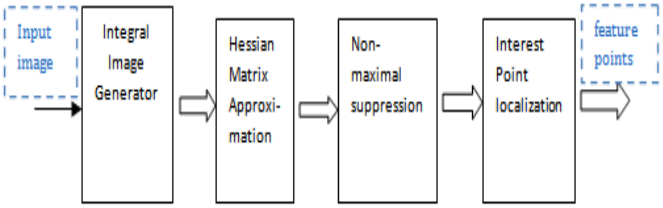

Fig1: Block diagram of SURF algorithm

S1: Integral image Generation: In SURF algorithm, the Integral Image generator is introduced to achieve fast computation of the box filter convolution used later in stages.
The integral image (II) is generated from the origin image (I). The Integral image value at location (i, j) is II(i, j) which represents the sum of all pixels in the input image I within a rectangular region formed by the origin (0, 0) and (i, j), i.e.,

$$II(i, j) = \sum_{x=0}^{i} \sum_{y=0}^{j} I(x, y) \qquad (1)$$

Once the integral image generation completed, it takes only three addition operations to calculate the sum of the intensities over any upright, rectangular area (abcd) (see Equation 2). Hence, the calculation time is independent of its size.

$$I=II(a) + II(b) - II(c) - II(d) \qquad (2)$$

S2: Calculation of Approximated determinant of the Hessian matrix: The feature point detection part of a SURF algorithm is mainly based on the Hessian matrix determinant, i.e. It depends on local maximum of the Hessian matrix determinant. When the determinant is a local maxima, a blob like structure is detected. The blob like structure means a small area that is lighter intensity or darker intensity than its surrounding pixels.
In general for the conventional Hessian matrix calculation, the convolution of the Gaussian second order derivatives $\frac{\partial^2}{\partial x^2} g(s), \frac{\partial^2}{\partial y^2} g(s), \frac{\partial^2}{\partial x \partial y} g(s)$ should be calculated with the image I at the point (i, j) at the scale s. This is too complex to calculate Gaussian second order derivatives due to more no of computations. Therefore, SURF algorithm uses box filters to approximation of the second order Gaussian derivatives. By using the integral images generator discussed in the previous section, these approximations are evaluated at a very low computational cost.

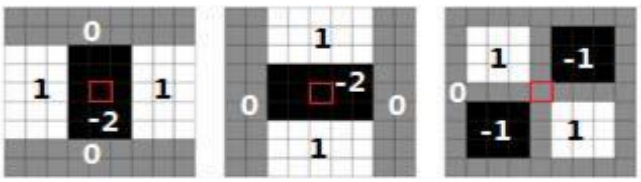

Figure2. 9×9 box filters: Dxx, Dyy, Dxy

Figure 2 illustrates 9×9 box filters, which are used as approximations of a Gaussian second order derivatives with scale s =1.2 and represent the lowest scale for computing the blob response maps. Box filters can be denoted by Dxx, Dyy, Dxy (which are used as a approximations of Gaussian second order derivatives $\frac{\partial^2}{\partial x^2} g(s), \frac{\partial^2}{\partial x \partial y} g(s), \frac{\partial^2}{\partial y^2} g(s)$ respectively). The weights applied to the rectangular regions are 0 is for the gray area, 1 is for the white area, and -1/-2 is for the black area to ensure computational efficiency.
These yields

$$Det(H_{approxmation}) = Dxx\ Dyy - (W\ Dxy)^2 \qquad (3)$$

Here, W is the relative weight of the box filter responses, which is used to balance the expression for the calculation of Hessian's determinant. The weight is set to 0.9 in practice.
The det(Happroximation) represents the blob response in the image at location (i, j). The det(Happroximation) is a positive value which represents maxima. These responses are stored in a blob response map for different values of scale.
S3: Scale space construction: The scale space construction means that the feature points can be found at different scales in SURF algorithm. This scale invariance was derived from SIFT algorithm. In SIFT algorithm,

the scale spaces is implemented as an image pyramid. In SIFT the images are repeatedly smoothed with a Gaussian filter and then sub sampled to achieve high level of the pyramid. The filter remains unchanged, and the image size is changed for each scale level repeatedly. In contrast, in SURF algorithm, due to the usage of integral image generator, box filters of any size can be applied to the original image with same speed.

Table I FILTER SIZE

octave	Layer 1	Layer 2	Layer 3	Layer 4
1	9	15	21	27
2	15	27	39	51
3	27	51	75	99

In Scale space construction the scale space is divided into different octaves. An octave represents a series of filter response maps, the responses are obtained by convolving the same input image with a filter of different size. In general, each Octave contains 4 layers. The construction of the scale space starts with filter of size 9x9, which calculates the blob response of the image for the smallest scale s = 1.2. In the first octave, the size of the layers increases by 6 pixels per layer. The second octave increases by 12 pixels, and the 3rd octave increases by 24 pixels, etc. Refer to Table 1.

For a layer of size M×M , the scale can be calculated with s = 1.2×9/M . Perform the convolution of different layers with the original image and compute the response at each location with Equation 3; then, response map with different scales(s) will be obtained, thus constituting a 3-dimensional space (i, j, s).

S4: Feature point localization: After completion of Hessian matrix determinant calculation, the interest feature points with a strong response will be localized by using feature point localization.

This is includes 3 steps.

(1) Threshold: In threshold set a threshold value to filter out the feature points point with a weak response. The value of the threshold is balanced in between the feature point number and the strength of the feature point.

(2) Non-maximum suppression: In 3 Dimensional (i, j, s) scale space, non-maximum suppression is performed in each 3x3x3 local area.

(3) Interpolation calculation is used to obtain accurate feature point. Because the sample step is 2n (n=1, 2...), it is necessary to localize the accurate position by interpolating on the 3-dimensional (i, j, s) space. The offset is achieved by fitting using a 3-dimensional quadratic equation fitting method.

Using the Hessian determinant, a Taylor expansion is performed around the feature points.

$$H(i) = H + \frac{\partial H^T}{\partial i} i + \frac{1}{2} i^T \frac{\partial^2 H}{\partial i^2} i \qquad (4)$$

$$\text{then } \hat{i} = (i, j, s)$$

$$i = -\frac{\partial^2 H^{-1}}{\partial i^2} \frac{\partial H}{\partial i} \qquad (5)$$

If in the offset on i, the j and s directions are less than 0.5, then this point is a proper feature point. The accurate location of this feature point is

$$\text{Location(accu)} = \text{Location(coarse)} + \text{Steplength} * \text{off set} \quad (6)$$

So far, we have obtained the location and the scale information of the feature points (x, y, s).

III. Existing method of Integral image

In existing method the integral image generated by using serial addition of image and integral image pixels is shown in equation(1) and diagram as shown in below. The input of the integral image module is the source image (intensity), and the output is the integral image (integral value). Refer to Figure 3, where there are 4 adjacent pixels A(i−1, j −1), B(i−1, j), C(i, j−1), and D(i, j). Let II(A), IΣ(B), and II(C) denote the integral values of A, B, and C; then,

$$II(D) = II(A) + (II(B) - II(A)) + (II(C) - II(A)) + I(D)$$

$$= II(B) + II(C) - II(A) + I(D)$$

Or

$$II(i, j) = II(i-1, j) + II(i, j-1) - II(i-1, j-1) + I(i, j) \qquad (7)$$

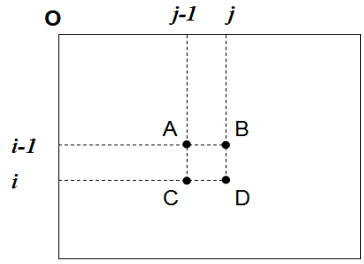

Figure 3: Calculate IΣ(D) with its upper-left neighbors

Based on Equation 7, we can construct the data path of the integral image, as illustrated in Figure 4. We need a FIFO with the length of a row to store the integral value of row i − 1. Moreover, two registers are needed to store II(i, j−1) and II(i−1,j−1).

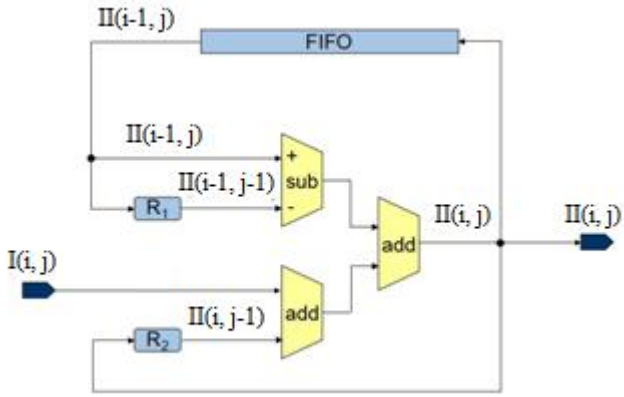

Figure 4: Data path of integral image

For 2x2 image we calculate by using serial addition 5 adders are required.

II[1][1] = I[1][1];
II[2][0] = I[2][0]+II[1][1];
II[1][2] = I[1][2]+II[1][1];
II[2][2] = II[1][2] + II[2][1] - II[1][1] + I[2][2];

For image size of MxN the adders recuried to design is3MN.

IV. Proposed method of integral image

In existing method the no. of additions and computations are more. To reduce the delay and the no. of adders, the proposed method uses single row parallel addition using recursive addition method.

The recursive equations presented below:

$$S(i, j) = I(i, j) + S(i, j - 1) \quad (7)$$
$$II(i, j) = II(i - 1, j) + S(i, j) \quad (8)$$

Where I(x, y) is the input pixel value at image location (i, j), S(i, j) is the cumulative row sum value at image location (i, j) and II(i, j) is the integral image value at image location (i, j). These equations reduce the number of additions involved to 2MN.

Based on the equitation's 7&8 the data path for integral image generator is shown in fig. 5.

The data path for proposed method is shown in figure 5.

International Journal Of Electronics Engineering (Issn: 0973-7383)
Volume 10 • Issue 2 pp. 233-240 June 2018-Dec 2018 www.csjournals.com

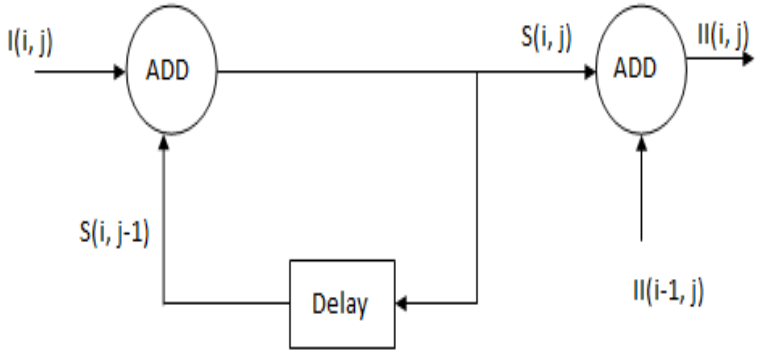

Figure 5: Data Flow Graph of recursive equations for a single row of the input image.

For recursive addition methode to calculate the integral image for each pixel two adders are required.

$$S(1, 1) = i(1, 1) + S(1,0)$$
$$ii(1, 1) = ii(0,1) + S(1, 1)$$
$$S(2, 1) = i(2, 1) + S(2,0)$$
$$ii(2, 1) = ii(1,1) + S(2, 1)$$

For Image of size MxN the adders required to compute integral image is 2MN. Where M- row size, N- column size.

V. Synthesis and Simulation results

In the design of Integral image generator, the source image is converted in to text file using Matlab and loaded in to a memory or a register to further processing using Xilinx ISe14.7. The source image in test format is shown in figure 6.

Figure 6 : Source image

Figure 7 shows the text file generated by Matlab for the input source image. The text values is in the form of Hexadecimal format each pixel represents 8 bit.
Figure 8 shows the text file loaded in to a memory/Register using Xilinx ISE 14.7. in this each pixel value is loaded in to a each memory location is shown in fig8.

International Journal Of Electronics Engineering (Issn: 0973-7383)
Volume 10 • Issue 2 pp. 233-240 June 2018-Dec 2018 www.csjournals.com

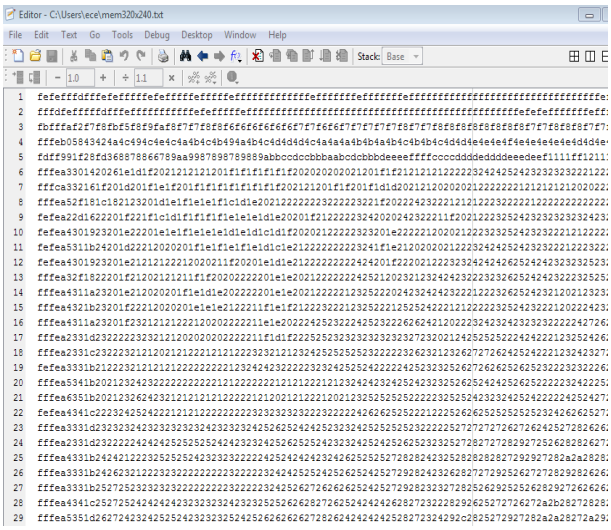

Figure 7 : Text file Generated by Matlab

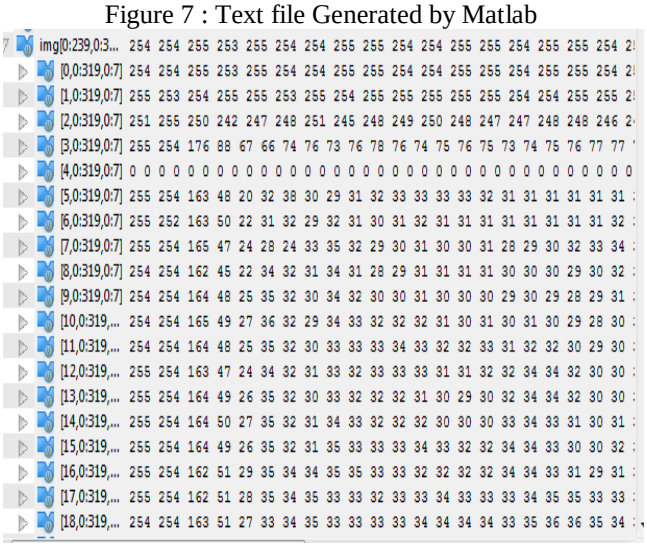

Figure 8: Text file Loaded In to a Register or memory

The input to the integral image generator is source image. The RTL for integral image is shown in below figure 9 and simulation results are shown in figure 10.

In RTl a be the input in which each pixel of image is loaded sequentially to compute the integral image. b, d are the inputs to display the integral image. r & c are inputs represents row and column size. and z be the output of integral image.

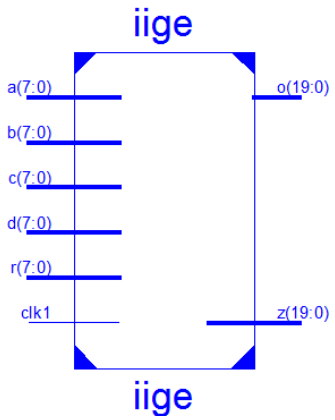

Figure 9: RTL for Integral image generation

Figure 10 shows the integral image generated for the input source image using Xilinx ISE 14.7. The integral image value is displayed for the Pixel II(10,10) in output z.

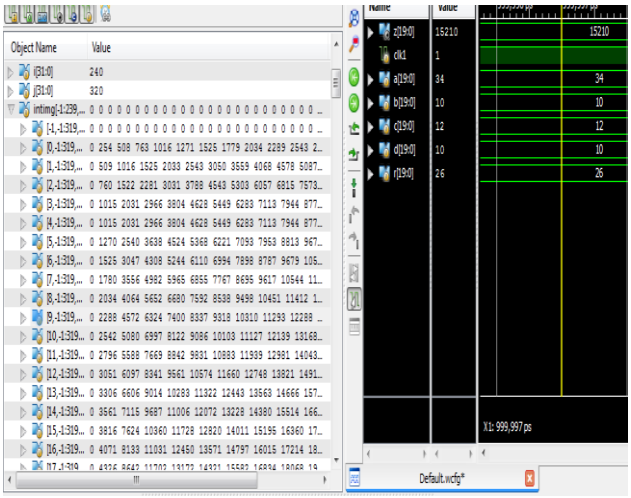

Figure 10: Simulation results for integral image

Device utilization summary of Integral image generator implementation is shown in figure 11.It shows LUTs, IOBs, Registers utilize to design FPGA.

Device Utilization Summary (estimated values)			[-]
Logic Utilization	**Used**	**Available**	**Utilization**
Number of Slice Registers	20	866400	0%
Number of Slice LUTs	64534	433200	14%
Number of fully used LUT-FF pairs	20	64534	0%
Number of bonded IOBs	60	720	8%
Number of BUFG/BUFGCTRLs	1	32	3%

Figure 11: Device utilization Summary

Comparison of proposed work with existing work:

Comparison of different parameters between existing and proposed methods for 320x240 image is shown in table 2.

Table 2: existing Vs Proposed methods

S.No	Parameter	Serial addition	Recursive
1	Delay(ns)	7.217	5.751
2	adders	7	4
3	Xor's	5974	3176
4	LUT's	73683	64534
5	Memory(KB)	1788988	1500796

Comparison of different parameters between existing and proposed methods for different images is shown in table 3.

Image	Serial addition of IIG		Recursive addition of IIG	
	No. of XORs	Delay(ns)	No. of XORs	Delay(ns)
320x240	5944	7.350	3176	5.819
256x192	5272	7.179	2952	5.544
128x96	3110	6.783	1866	5.148
64x64	1980	6.182	1188	4.727
32x32	850	6.028	510	4.312
16x16	360	5.874	216	4.188
8x8	150	5.347	90	3.721

VI. Conclusion & Future scope

In this paper, Integral image generator in SURF detector is implemented by using recursive equations which reduce the average delay up to 24.933% and reduces the 43.58% of hard ware when compared to the serial addition implementation of integral image generation.
Integral image generator is implemented in this paper for single row of image. In future integral image generator with parallel operation techniques for more no. of rows of a image can reduce more delay.

VII. References

[1] Wenjie Chen, Shuaishuai Ding, Zhilei Chai, Daojing He, Weihua Zhang, Guanhua Zhang, Qiwei Peng and Wang Luo's "FPGA-Based Parallel Implementation of SURF Algorithm" IEEE 22nd International Conference on Parallel and Distributed Systems (ICPADS), pp. 308 - 315,Year: 2016.

[2] C. Harris and M. Stephens, "A combined corner and edge detector.," vol. 15, p. 50, Citeseer, 1988.

[3] T. Lindeberg, "Feature detection with automatic scale selection, "International journal of computer vision, vol. 30, no. 2, pp. 79–116, 1998.

[4] K. Mikolajczyk and C. Schmid, "Indexing based on scale Invariant interest points," vol. 1, pp. 525–531, IEEE, 2001.

[5] D. G. Lowe, "Distinctive image features from scale-invariant key points," International journal of computer vision, vol. 60, no. 2, pp. 91–110, 2004.

[6] H. Bay, T. Tuytelaars, and L. Van Gool, "Surf: Speeded up robust features," pp. 404–417, Springer, 2006.

[7] H. Bay, A. Ess, T. Tuytelaars, and L. Van Gool, "Speeded-up robust features (surf)," Computer vision and image understanding, vol. 110, no. 3, pp. 346–359, 2008.

[8] J. Chao, R. Huitl, E. Steinbach, and D. Schroeder, "A novel rate control framework for sift/surf feature preservation in h. 264/avc video compression," IEEE Transactions on Circuits.

[9] N. Cornelis and L. V. Gool, "Fast scale invariant feature detection and matching on programmable graphics hardware," pp. 1–8, IEEE, 2008.

[10] T. Krajnk, J. vb, S. Pedre, P. ek, and L. Peuil, "Fpga-based module for surf extraction," Machine Vision and Applications, vol. 25, no. 3, pp. 787–800, 2014.

[11] Y. Han and E. Oruklu, "Real-time traffic sign recognition based on zynq fpga and arm socs," in IEEE International Conference on Electro/Information Technology, IEEE International Conference on Electro/Information Technology, pp. 373–376, 2014.

[12] H. Y. Chang, I. H. R. Jiang, H. P. Hofstee, D. Jamsek, and G. J. Nam, "Feature detection for image analytics via fpga acceleration," IBM Journal of Research and Development, vol. 59, pp. 8:1–8:10, March 2015