IIETA International Information and Engineering Technology Association
Advancing the World of Information and Engineering

Traitement du Signal
Vol. 36, No. 5, October, 2019, pp. 377-382
Journal homepage: http://iieta.org/journals/ts

GA-Based Optimization of SURF Algorithm and Realization Based on Vivado-HLS

Hüseyin Özdemir, Refik Sever, Övünç Polat*

Faculty of Engineering, Department of Electrical and Electronics Engineering, Akdeniz University, Antalya, Turkey

Corresponding Author Email: ovuncpolat@akdeniz.edu.tr

https://doi.org/10.18280/ts.360501

Received: 5 June 2019
Accepted: 18 August 2019

Keywords:
speeded-up robust features, high-level synthesis, genetic algorithm, optimization, character recognition

ABSTRACT

The aim of this study is realization of SURF algorithm based on Vivado-HLS tool for FPGA platform. The SURF algorithm is a method which is used in image processing that is not affected by feature changes such as size, color and contrast. Genetic algorithm is used to determine the optimum values of the parameters affecting the success of the SURF algorithm in this work. It has been observed that when the parameter values determined using the optimization algorithm is used, the success rate significantly increases. The proposed method has been tested for character recognition application with fixed font which is independent from rotation and character size. In this work, the tests were carried out with images formed from numbers. Reference and test pictures for each number were created. The test images consist of images of different sizes and rotations. Optimal parameter values were used in HLS after being determined by proposed approach. Using the proposed optimized SURF structure, high success rates were obtained.

1. INTRODUCTION

The SURF (Speeded-Up Robust Feature) algorithm tries to determine the best features of an object that is desired to be detected on the image and can quickly resolve the result. The SURF algorithm provides the advantage of being invariant against both changes and rotations [1-2].

In this study, the SURF algorithm was implemented with the Vivado-HLS tool. Using HLS, algorithms can be implemented by using C, C++, *etc* languages without using hardware description languages (verilog, VHDL, *etc.*). The SURF algorithm was implemented with the C-language in this work.

Bay et al. [2] aims to develop a detector that does not compromise performance, can perform fast calculations, find enough distinguishing features and reduce the descriptor size and complexity. The SURF detector is based on the Hessian matrix. Integral images were used to reduce the calculation time. For this reason, it is expressed as a Fast-Hessian detector.

In the literature, the SURF algorithm is implemented with C code using HLS by Faliagkas [3], and "float" and "integer" are selected as the data types used when defining the registers. The system has been optimized using the directives available in HLS. The system is designed to work with Microblaze.

In this study, fixed-point data type was chosen for the identification of the caches. Since the SURF algorithm supports rotation instability up to ± 15 degrees by default, it is aimed that the system can work faster by removing the structure related to the rotation in the algorithm. In addition, the threshold value and octave - 1 values used in the SURF algorithm are optimized with the genetic algorithm and used with the values that have the greatest success so that the success rate is more in the SURF algorithm. The parameters obtained here are aimed at establishing the best success rate using HLS. Cai et al. [4] have presented a SURF algorithm for hardware implementation, which combined several optimization techniques in literature and three approaches (word length reduction, sampling radius reduction and low bits abandon). The computation operations of the simplified and optimized SURF were reduced by 50% compared with the unoptimized SURF algorithm. In referred study, the optimized design of SURF has been realized by using TSMC 65 nm process. Oliveira et al. [5] have presented a new approach to retarget images based on both genetic algorithm and a high-performance scale and rotation-invariant interest point descriptor.

The digit recognition application was chosen to determine the appropriate parameters and to test the success of the proposed system and the images created from the digits were used in this study. It is aimed to determine the digit images of different sizes and different rotations. There are different studies in the literature on font recognition and character recognition [6-8]. The method proposed by Zahedi and Eslami [6] can recognize automatic Persian / Arabic fonts based on SIFT (Scale Invariant Feature Transform). Since the SIFT algorithm supports scale and rotation invariance, it is not affected by such factors. The SIFT algorithm was used by Jamjuntr and Dejdumrong [7] to identify the Thai font, and a success rate of 97.37% was achieved in the tests performed. Ouyang et al. [9] compare 3 robust feature detection methods: SIFT, Principal Component Analysis (PCA)-SIFT and SURF. The performances of these three methods were compared in terms of scale changes, rotation, illumination changes, blur and affine transformations.

Next section gives used methods and proposed approach for implementation of SURF algorithm based on Vivado-HLS. Simulations and obtained results are given in the section 3.

2. METHOD

2.1 SURF algorithm

SURF is an algorithm used for feature mappings. It was developed based on the SIFT algorithm. SIFT detects the key points existing in an image and calculates the attributes by using the descriptors through these points. The most important advantage of this algorithm is that it is not influenced too much by the fact that when the key points are encountered, the picture is rotated in different directions, the size of the picture is different, or the intensity of light in the picture varies [10]. The SURF algorithm can work faster and more effectively than SIFT. The SIFT algorithm detects the discriminative points on the image and extracts the identifiers for each discriminant point. These distinctive points extracted are independent of features such as rotation, scale etc.

The SURF algorithm tries to localize the points of interest on the image [11]. The high performance and accuracy of the SURF algorithm can be attributed to the use of integral images [12]. Furthermore, the SURF detector is efficient and yields accurate results because of the use of the Hessian matrix.

The SURF algorithm implements the scale-space by applying the box filters in the original image to the increasing size. The reasons for this performance improvement are; It permits multiple layers of the scale-space pyramid to be processed at the same time, and the processing of the sub-samples of the view is not performed. In other words, for different scale situations, the scale of the filters is changed without changing the scale of the view.

The scale space is divided by the number of octaves at which the octave belongs to the sequence of reaction maps computed with certain filters. The creation of the scale space starts with a 9x9 filter that computes the determinant response of the image for the smallest scale [11].

Each octave is divided into 4 intervals. Subsequent layers are obtained by scaling the filters up while preserving the same filter-placement ratio. As the filter size increases, the values of the associated Gaussian scale increases. The descriptors are calculated for specific pixels of the scene. Filters in different structures and sizes specify which rows and columns of pixels should be processed. After calculating the four inputs of the Hessian matrix, normalization is applied for each scale-space position. This scale normalization of the determinant response map is obtained by dividing these inputs by the filter area [11].

2.2 High-level synthesis (HLS)

With Vivado-HLS developed by Xilinx [13], code generated with C, C ++ or SystemC can be converted into RTL source code or packages called IP (Intellectual Property) Core.

The first step of the implementation is the creation of the algorithm with a supported programming language (C, C ++, SystemC). The generated code is added to "HLS Tools". In addition, the system directives are used to improve performance or reduce the amount of hardware used. The HLS directives are used to generate the optimized RTL code.

Once the modules implemented in HLS are tested and synthesized, they can be packaged as "ip-core" and transferred to the Vivado environment. It can be used as a library file in FPGA projects to be created in the Vivado environment.

2.3 Genetic Algorithm (GA)

Genetic Algorithms are one of the leading artificial intelligence optimization techniques used today. Genetic algorithm is a search method that exemplifies the mechanism of evolution in nature. Genetic algorithms produce solutions that are constantly healing based on the rule of life of the best in nature [14, 15].

It is aimed to optimize the sizes of threshold values and box filters used in OPENSURF [16, 17] algorithm by using GA in the study. In proposed approach, GA tries to maximize the number of matching points of interest by changing threshold and box filter sizes. The optimum values of five parameters in the GA and SURF algorithms were determined. These parameters are threshold value and octave-1 (4 values). In the optimization stage, the population size was 50, the selection function was "stochastic uniform" and the mutation rate was "0.01". The settings made to limit the optimization are the lower and upper limits selected for the threshold value 0.0001 - 0.01, the selected upper and lower limits for the octave-1 values {4 5 10 15} - {10 15 27 27}.

2.4 Implementation of the SURF algorithm

The OPENSURF library is an implementation of the SURF algorithm [16-19]. Realized by Christopher Evans in C ++ languages [16]. It has been adapted to the Matlab program by Kroon [17]. Fan et al. [19] propose the high performance hardware architecture on FPGA to implement SURF algorithm. The aimed hardware architecture is consistent with the OpenSURF. Chen et al. [20] proposed an implementation and acceleration architecture for OpenSURF algorithm on FPGA platform.

In the study, software analysis of the library was done first. After making the required arrangements, the modules required for HLS have been identified. In order to determine the most suitable bit lengths of the registers, the smallest bit lengths that can be generated by various simulations are determined by software. The goal is to save performance and minimize memory usage by storing the data in the smallest possible memory area with minimal loss.

Then, the functions determined in software as modules were adapted to HLS and synthesized by making necessary arrangements. In order to optimize the results of the synthesis, the ones that are available in the HLS are used.

In the stage of using the OPENSURF Library, analyzes were made with reference to the version of the library adapted by Kroon [17]. The algorithm basically consists of two steps. Firstly, the reference image to be taken of the reference descriptor vectors must be determined. This can be any ROI on an object or image specified in the image. The specified image is subjected to the operations of the SURF algorithm to obtain descriptor vectors according to the determined parameters. The resulting vectors are stored and then used in search operations.

Secondly, similar operations are repeated for the image to be searched for the object whose vectors are detected. It is also passed through the operations of the SURF algorithm to obtain descriptive vectors. If the object being searched can be predicted in which region on the image, only the processing of that part of the image will yield better results in terms of speed. If there is no prediction, it will be necessary to compare the reference vectors with the descriptor vectors of the entire image [18].

The points of interest found are based on the similarities between the obtained vectors and the reference vectors. The most similar is the sort that will be in the first place. That is,

according to the SURF algorithm, the coordinates of the two pixels that are most similar between the two images are determined. The number of these pixels varies according to the number of points of interest on the reference image and the image being searched. Also, there may not be any points of interest on the image.

Fixed-point data types were changed as data loss was minimized in the study, and the system was integrated into the HLS as modules in the final state. In the synthesis process with HLS, "Zedboard Zynq Evaluation and Development Kit (xc7z020clg484-1)" was used and the system frequency was selected as 100 MHz.

The mode of operation used was the Upright SURF (U-SURF) rotation control mode, which maintains stability for image rotations of up to ± 15 degrees. Also, since there are not too many scale changes, one was chosen as the octave value. The octave value can be increased for images with a larger scale change, so that the points of interest can be detected more successfully. However, increasing the octave value increases the processing load and therefore causes the system to run slower.

3. SIMULATION RESULTS

The performance of the proposed optimum SURF algorithm has been tested for digit recognition application independently of character size and rotation. For the reference and test images, figures of 45x80 were created. 10 images have been created as a reference. The images were created using figures of different sizes and rotations so as not to be the same as the reference image for use in the test data set. As a result, 10 pictures for reference and 80 pictures for use in the test data set have been created.

While the 45x80 image is used as the reference image, the test image is 45x800, which is created by merging ten images in 45x80 sizes. The purpose here is to find each digit in the image generated by selecting from the test data set. Thus, it is aimed to determine the corresponding digit from each picture selected from the test data set using the SURF algorithm.

The success rate can be determined because it is known which of the reference digit and the corresponding figure is in which region of the randomly generated test image. That is, points of interest are determined for the figure used in the reference picture. Likewise, points of interest for the test picture are determined. The points of interest are sorted by comparison with each other and the most similar ones are identified. The distribution of points of interest most similar to the reference points of interest in the test image refers to the test success. If the entire distribution of points of interest is in the region of the figure sought, the success rate is expressed as 100%. GA optimization algorithm is used to increase the success rate obtained by default parameter values in the study. The optimization algorithm is run for two different situations, over 50% of the number of points of interest detected using the optimization algorithm and over 70%. As a result of optimization for these two different situations, it has been observed that the success rate is significantly increased compared to the default parameters. The parameter values determined by the optimization algorithm are used in HLS. Reference and test images were used for the tests made at HLS in 45x80 dimensions.

For the structure of the SURF algorithm, a reference picture is needed. The points of interest on the reference images are stored in the memory and compared with the points of interest on the test images. Numbers are used for the reference dataset. Each figure was created as a 45x80 image with font size 20 and writing style "Calibri" (rotation 0 degrees). The images used for the reference are shown in Figure 1.

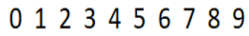

Figure 1. Reference digit images

A test data set was created to observe how the parameter changes change the algorithm result. In addition, this data set has been created to measure the success of the algorithm and to be used during optimization. Optimization tool tries to determine optimal parameters by using reference images and test images. For the test data set, each number should have a font size of 18 and 22, font format "calibri", rotation of +10 degrees, 0 degrees and -10 degrees, font size of 20, font format "calibri" degree and -10 degrees in 45x80 dimensions as the picture was created. The important thing here is that the images used as references are not included in the test data set. As a result, one reference image and six test images were created for each figure (10 reference images, 60 test images). Figure 2 shows the test images created for the all number zero. (the same image as the reference used in the test data set is not used).

Figure 2. Example test data for 0 digit

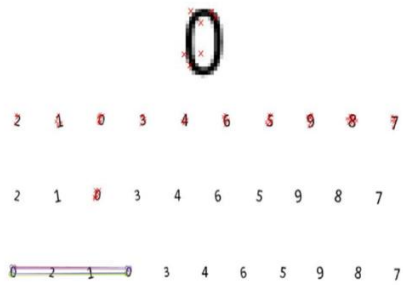

Figure 3. Test results for 0 digit

In order to determine the success of the optimum parameter values determined using the reference and test data in this study, a different validation data set was also created. This data set contains 10 images in a similar way as in the test data set. Twenty validation images were created, randomly selected from the dataset, resulting in two images for each digit. By using the optimization algorithm, it is aimed to maximize the number of correct matches with each reference image, the test image including ten digits in different sizes and rotations. To find this number, at least 50% of the points of interest in the reference image at the end of each test are examined to see if it is in the relevant region in the test image. If there are as many points of interest as possible in the area concerned, the success rate is increased by 1. For a total of 60 test images, if there is at least 50% interest in the respective region of each view, the success counter will be 60 and the test success will be 100%. In Figure 3, one of the tests for the number zero is shown. At the top, the points of interest for the reference number zero are shown. In the second row, the points of interest in one of the test images created in 45x800 dimensions are shown. In the third row, the points of interest in the reference image and the

points of interest in the test image are shown. In the fourth row, the reference image and matching points of interest in the test image are shown matched with the drawing.

In the OPENSURF algorithm, the octave-1 values are set to 9 - 15 - 21 - 27 by default and the threshold value is set to 0.0001. Here the threshold value represents which of the candidate points of interest will be considered as points of interest. The octave values represent the box filter dimensions to be used. That is, when octave-1 default values are used, box filters are performed in sizes 9x9, 15x15, 21x21, 27x27. With the optimization algorithm, it is aimed to determine the effect of these filters on the success rate of the changes in dimensions.

The obtained test results shown in Table 1 are shown. Here, the success of the test and the overall test are shown separately for each figure. According to this, the number of tests that were 50% (compared to the number of points of interest) was found to be 18 in a total of 60 tests. The 50% threshold value indicates that the related digit is at least 50% correct, since the percentages in these digits represent the success of finding the correct number. Table 2 shows the cases in which the success rate is higher than 70%. According to this, the number of tests with at least 70% accuracy in the 60 tests was 12. As a result, the success rate is 20%. It is aimed to determine more appropriate values of octave - 1 and threshold values in order to increase success rates. For this purpose, the GA optimization algorithm was first run to increase the number of tests that were more successful than 50% (based on the number of points of interest). The octave-1 values were found to be 4, 9, 18, 23 and the threshold value was 0.007. Table 3 shows the test results. Accordingly, the number of tests that were more successful than the 50% tested in the 60 tests was 59. As a result, the success rate was calculated as 98.3%.

The number of tests that were more successful than 70% was 54. The success rate is obtained as 90%. In order to verify the parameter values determined by the optimization algorithm and to determine the success of the system, tests were carried out using 45x800 images (selected from images of 45x80 size) generated by random selection from the test data set. The results obtained for a total of 200 validation data are shown in Table 4. Accordingly, it has been observed that the parameter values determined by the optimization algorithm significantly increase the success rate of the system.

Table 5 and Table 6 show the accuracy rates of the optimal parameter values determined by the GA for HLS tests. In the tests performed here, 45x80 size reference image and 45x80 size test image are used. For example, if the number zero is accepted as the reference point size 20, the rotation 0 is selected as the first test with point size 22, the rotation 10, the second test with point 18, and the rotation 0. According to the test results, the accuracy rate obtained with the default parameter values is 56% while the success rate obtained using the parameter values calculated with optimization is 81%.

Table 1. Test accuracy results for the unoptimized case (for more than 50% of detected points of interest)

When the parameters are selected as Octav=[9,15,21,27] and threshold=0.0001								
Numbers	The Number of Feature in Reference Image	The success of found numbers that were in test images. (for each number separately %)						The number of tests which are more success from percent 50
		Test - 1	Test - 2	Test - 3	Test - 4	Test - 5	Test - 6	
0	8	25.00	62.50	50.00	37.50	37.50	25.00	1
1	2	100.00	50.00	0.00	0.00	0.00	50.00	1
2	6	33.33	50.00	33.33	50.00	50.00	83.33	1
3	5	0.00	60.00	40.00	20.00	20.00	60.00	2
4	4	100.00	50.00	75.00	25.00	25.00	75.00	3
5	4	75.00	50.00	50.00	25.00	25.00	50.00	1
6	3	66.67	66.67	33.33	33.33	33.33	66.67	3
7	4	50.00	50.00	75.00	75.00	100.00	100.00	4
8	3	0.00	33.33	100.00	33.33	33.33	0.00	1
9	2	100.00	50.00	50.00	50.00	50.00	50.00	1
18 of the 60 tests were more successful than the 50%. Result=18/60=>30%								18

Table 2. Test accuracy results for the unoptimized case (for more than 70% of detected points of interest)

When the parameters are selected as Octav=[9,15,21,27] and threshold=0.0001								
Numbers	The Number of Feature in Reference Image	The success of found numbers that were in test images. (for each number separately %)						The number of tests which are more success from percent 70
		Test - 1	Test - 2	Test - 3	Test - 4	Test - 5	Test - 6	
0	8	25.00	62.50	50.00	37.50	37.50	25.00	0
1	2	100.00	50.00	0.00	0.00	0.00	50.00	1
2	6	33.33	50.00	33.33	50.00	50.00	83.33	1
3	5	0.00	60.00	40.00	20.00	20.00	60.00	0
4	4	100.00	50.00	75.00	25.00	25.00	75.00	3
5	4	75.00	50.00	50.00	25.00	25.00	50.00	1
6	3	66.67	66.67	33.33	33.33	33.33	66.67	0
7	4	50.00	50.00	75.00	75.00	100.00	100.00	4
8	3	0.00	33.33	100.00	33.33	33.33	0.00	1
9	2	100.00	50.00	50.00	50.00	50.00	50.00	1
12 of the 60 tests were more successful than the 70%. Result=12/60=>20%								12

Table 3. Test results for optimized case by using GA (for more than 50% of detected points of interest)

When the parameters are selected as octav=[4,9,18,23] and threshold=0.007 (By using optimization algorithm)								
Numbers	The Number of Feature in Reference Image	The success of found numbers that were in test images. (for each number separately %)						The number of tests which are more success from percent 50
		Test - 1	Test - 2	Test - 3	Test - 4	Test - 5	Test - 6	
0	4	100.00	100.00	100.00	100.00	100.00	100.00	6
1	5	100.00	80.00	100.00	80.00	80.00	60.00	6
2	6	83.33	100.00	66.67	83.33	83.33	83.33	6
3	7	57.14	42.86	85.71	57.14	57.14	85.71	5
4	5	80.00	80.00	100.00	100.00	100.00	100.00	6
5	7	71.43	71.43	85.71	71.43	71.43	71.43	6
6	5	80.00	80.00	100.00	100.00	100.00	100.00	6
7	4	100.00	100.00	75.00	100.00	100.00	100.00	6
8	10	90.00	80.00	100.00	100.00	100.00	100.00	6
9	7	100.00	71.43	85.71	100.00	100.00	85.71	6
59 of the 60 tests were more successful than the 50%. Result=59/60=>98.3%								59

Table 4. Accuracy results for validation data

	The number of tests which are more success from percent 50	**The number of tests which are more success from percent 70**
When the parameters are selected as Octav=[9,15,21,27] and threshold=0.0001	30%	19%
When the parameters are selected as Octav=[4,9,18,23] and threshold=0.007 (By using optimization algorithm)	96%	84%

Table 5. HLS test success results for unoptimized case

When the parameters are selected as Octav=[9,15,21,27] and threshold=0.0001							
Numbers	The Number of Features in Reference Image	TEST – 1			TEST – 2		
		The Number of Features in Test Image	The number of features which are success	The Success Rate %	The Number of Features in Test Image	The number of features which are success	The Success Rate %
0	8	4	3	37.5	5	3	37.5
1	2	2	1	50	7	0	0
2	6	3	3	50	5	4	66.67
3	7	5	4	57.14	5	2	28.57
4	5	3	3	60	9	3	60
5	4	5	4	100	4	4	100
6	3	4	2	66.67	2	2	66.67
7	6	7	4	66.67	3	3	50
8	5	5	3	60	4	3	60
9	3	3	1	33.33	4	2	66.67
				58%			54%

Table 6. HLS test success results for GA optimized case

When the parameters are selected as Octav=[4,9,18,23] and threshold=0.007							
Numbers	The Number of Features in Reference Image	TEST – 1			TEST – 2		
		The Number of Features in Test Image	The number of features which are success	The Success Rate %	The Number of Features in Test Image	The number of features which are success	The Success Rate %
0	4	5	4	100	3	2	50
1	5	3	3	60	5	5	100
2	6	3	3	50	6	4	66.67
3	8	6	6	75	10	7	87.5
4	5	5	5	100	4	4	80
5	7	7	7	100	8	7	100
6	5	9	5	100	4	4	80
7	4	4	4	100	3	3	75
8	10	10	8	80	11	8	80
9	7	8	5	71.43	4	4	57.14
				84%			78%

4. CONCLUSIONS

In this work, it is aimed to construct the optimum performance SURF algorithm with HLS. Modules implemented with HLS in the study are primarily optimized as hardware using directives between HLS tools. The success of the SURF algorithm is tested on the data set generated by the numbers. In this context, tests were done using images in 45x80 dimensions. It is aimed to determine optimal values by optimizing the parameters used in algorithm with genetic algorithm. For this, the octave-1 values (representing the filter sizes used in the SURF algorithm) and the threshold value are optimized. Optimize algorithm has been determined optimal values by giving possible range of values and these values are used in SURF algorithm. Using the values of the optimized parameters, the success rate is significantly increased. Proposed approach can be used for different applications such as object recognition, handwriting character recognition.

REFERENCES

[1] Lei, F., Wang, W. (2014). A fast method for image mosaic based on SURF. 9th IEEE Conference on Industrial Electronics and Applications, Hangzhou, pp. 79-82. https://doi.org/10.1109/ICIEA.2014.6931135

[2] Bay, H., Tuytelaars, T., Van Gool, L. (2006) SURF: Speeded up robust features. European Conference on Computer Vision, ECCV 2006: Computer Vision – ECCV 2006, Springer, Berlin, Heidelberg, 3951: 404-417. https://doi.org/10.1007/11744023_32

[3] Faliagkas, K. (2013). High level synthesis of the OpenSURF algorithm. Student thesis, National Technical University of Athens, Greece.

[4] Cai, S., Liu, L., Yin, S., Zhou, R.Y., Zhang, W.L., Wei, S.J. (2014). Optimization of speeded-up robust feature algorithm for hardware implementation. Science China Information Sciences, 57(4): 1-15. https://doi.org/10.1007/s11432-013-4946-y

[5] Oliveira, S.A.F., Neto, A.R.R., Bezerra, F.N. (2016). A novel genetic algorithms and SURF-based approach for image retargeting. Expert Systems with Applications, 44: 332-343. https://doi.org/10.1016/j.eswa.2015.09.015

[6] Zahedi, M., Eslami, S. (2011). Farsi/Arabic optical font recognition using SIFT features. Procedia Computer Science, 3: 1055-1059. https://doi.org/10.1016/j.procs.2010. 12.173

[7] Jamjuntr, P., Dejdumrong, N. (2012). Thai font type recognition using SIFT. 2012 Ninth International Conference on Computer Graphics, Imaging and Visualization, Hsinchu, Taiwan, pp. 57-60. https://doi.org/10.1109/CGIV.2012.23

[8] Ahmad, R., Afzal, M.Z., Rashid, S.F., Liwicki, M., Breuel, T. (2015). Scale and rotation invariant OCR for pashto cursive script using MDLSTM network. 13th Int Conference on Document Analysis and Recognition (ICDAR), Tunis, Tunisia, pp. 1101-1105. https://doi.org/10.1109/ICDAR.2015.7333931

[9] Ouyang, N.J., Li, W.T., Wei, W., Pan, Q. (2013). A comparison of SIFT, PCA-SIFT and SURF. International Journal of Image Processing, 28(1): 58-64.

[10] Lowe, D.G. (2004). Distinctive image features from scale-invariant keypoints. International Journal of Computer Vision, 60(2): 91-110. https://doi.org/10.1023/B:VISI.0000029664.99615.94

[11] Bouris D., Nikitakis, A., Papaefstathiouand, I. (2010). Fast and efficient FPGA-based feature detection employing the SURF algorithm. 18th IEEE Annual International Symposium on Field-Programmable Custom Computing Machines, Charlotte, NC, USA. https://doi.org/10.1109/FCCM.2010.11

[12] Viola, P., Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Kauai, HI, USA. https://doi.org/10.1109/CVPR.2001.990517

[13] Xilinx. (2013). Vivado design suite user guide on high level synthesis. UG902 (v2013.2).

[14] Goldberg, D.E. (1989). Genetic algorithm in search, optimization, and machine learning. Addison-Wesley.

[15] Yeniay, Ö. (2001). An overview of genetic algorithms. Anadolu University Journal of Science and Technology, 2(1): 37-49.

[16] Evans, C. (2009). Notes on the OpenSURF library. Technical Report, University of Bristol.

[17] Kroon, D.J. (2010). OpenSURF. http://ch.mathworks.com/matlabcentral/fileexchange/28300-opensurf--including-image-warp, accessed on 12 May 2019.

[18] H.Özdemir, (2018). Realization of SURF algorithm based on vivado HLS for FPGA platform. Akdeniz University, Master Thesis, February.

[19] Fan, X., Wu, C., Cao, W., Zhou, X., Wang, S., Wang, L. (2013). Implementation of high performance hardware architecture of OpenSURF algorithm on FPGA; Proceedings of the 2013 International Conference on Field-Programmable Technology, Kyoto, Japan, pp. 152-159. https://doi.org/10.1109/FPT.2013.6718346

[20] Chen, C., Yong, H., Zhong, S., Yan, L. (2015). A real-time FPGA-based architecture for OpenSURF. Ninth International Symposium on Multispectral Image Processing and Pattern Recognition (MIPPR2015), Enshi, China. https://doi.org/10.1117/12.2205633