FPGA synthesis of an stereo image matching architecture for autonomous mobile robots.

G. Doménech-Asensi, R.Ruiz-Merino, J. Zapata, J.A. López-Alcantud
Dpto. de Electrónica y Tecnología de Computadoras
Universidad Politécnica de Cartagena
Cartagena, Spain
gines.domenech@upct.es

J.A. Díaz-Madrid
Departamento de Integración
Centro Universitario de la Defensa CUD-UPCT
Santiago de la Ribera (Murcia), Spain

***Abstract*—This paper describes a hardware proposal to speed up the process of image matching in stereo vision systems like those employed by autonomous mobile robots. This proposal combines a classical window-based matching approach with a previous stage, where key points are selected from each image of the stereo pair. In this first step the key point extraction method is based on the SIFT algorithm. Thus, in the second step, the window-based matching is only applied to the set of selected key points, instead of to the whole images. For images with a 1% of key points, this method speeds up the matching four orders of magnitude. This proposal is, on the one hand, a better parallelizable architecture than the original SIFT, and on the other, a faster technique than a full image windows matching approach. The architecture has been implemented on a lower power Virtex 6 FPGA and it achieves a image matching speed above 30 fps.**

Keywords—, Computer Vision, Stereo Vision; SIFT; SAD; FPGA;.

I. INTRODUCTION

The use of stereo vision technique for mobile robots navigation is increasing in the last years. This technique is an alternative to laser or sonar for unstructured environments or to artificial landmarks in known ones. Window-based image matching algorithms have been commonly used for this task [1], [2]. The more employed ones are the sum of absolute differences (SAD) [3], the sum of squared differences (SSD) and the normalized cross correlation (NCC) [4]. There are in the literature several implementations of image matching algorithms both based on software [5], [6] and hardware developments [7-11]. However, the computational cost of window-based image matching algorithms is very high, since it is proportional to the image size.

On the other hand, power consumption of mobile-robot vision-systems is a critical issue when power supply is a scarce commodity, as it happens for small robots with tiny batteries, for robots powered by solar cells or even for unmanned aerial vehicles (UAVs) [12]. This implies that the use of powerful software algorithms running of general purpose CPUs is discouraged for such applications, because they can demand an unaffordable power consumption. An alternative to cut down on power dissipation in stereo vision systems is to combine a simplification of such algorithms with the use of specific hardware architectures.

The image matching process can be significantly simplified if the window-based algorithm is not computed over all the pixels in both stereo images. So, if there is a previous stage, in which only selected points are chosen from both images of the stereo pair are matched, then the computational complexity of the algorithm can be considerably reduced. It is commonly assumed that two matching pixels can be found on the common epipolar line to both stereo images [7]. This restricts the number of target pixels analysed in the second image to those placed on this line and at a maximum predefined distance from the pixel in the first image. However, small misalignments among both cameras due to vibrations or other causes, can void this assumption. In this regard, in [13-16] alternative software implementations to speed up the image matching are proposed. They combine the capability of SIFT algorithm [17] to extract selected points (or key points) from an image, with a window-based algorithm (SAD and NCC), to perform the matching between two images of a stereo vision system. Thus, the window-based matching is only performed over the set of key points (KP) previously extracted, instead of being applied over the whole image. For a typical image, where the maximum number of key points is 1% of the image size, this represents a speed up of four orders of magnitude.

The SIFT algorithm is composed of two major stages of computation: key point extraction and descriptor generation. While the first stage requires most of the workload of the whole algorithm, the second one requires complex arithmetic operations. This has led to different synthesis approaches to obtain designs which progressively reach the real time operation. Due to its inherent parallel nature, FPGAs are the natural choice for a real time implementation of the SIFT algorithm. However, many of the mentioned implementations use a combination of FPGA for the first stage with an embedded processor for the second stage. This is because while the first stage of the algorithm is easily parallelizable, the nature of the arithmetic operations required by the second stage makes the use of software implementations more suitable.

Regarding real time hardware implementations (30 frames per second), in [18, 19] two hardware/software co-designs which use embedded FPGA processors and able to generate descriptors for QVGA and VGA images respectively, are described. A pure hardware FPGA based hardware implementations, also working with VGA images at real time is presented in [20]. Two CMOS implementations are described in [21, 22], which reach the same frame rate for VGA and HD1080 images respectively. Finally, in [23-25], three pure hardware FPGA implementations able to generate descriptors at a faster rate (above 50 fps) are described. However, these fast implementations require some simplifications to the original algorithm in order to achieve such processing rates.

Thus, replacing the second stage of the SIFT algorithm by a windows based matching approach eliminates the complexity of arithmetic operations. This combination of the first stage of the SIFT algorithm, to extract characteristic points from images, with a windows-based technique to perform matching between them is, on the one hand, a better parallelizable architecture than the original SIFT, and on the other, a faster technique than a full image windows matching approach. So, in this work, a full hardware implementation of this methodology is proposed, in order to improve both the processing speed and the power dissipation. The rest of the paper is organized as follows. Section II describes the system architecture, where key point extraction and region matching implementations are detailed. Results are described in Section III. Finally, conclusions are drawn in Section IV.

II. SYSTEM ARCHITECTURE

Fig. 1 shows the general architecture of the proposed system, where the main modules and the sizes of the digital buses used to interconnect them are detailed. As usual in every stereo vision system, the two cameras are placed side by side on an horizontal axis. Images taken from the left and the right cameras are stored in the respective memories. At the same time, these images are also processed by a key point extraction module. This module searches for KPs in both left and right input images at a frequency of one pixel per clock cycle. Every time a pixel is identified as a KP, its index is stored in the respective left or right memory or KP list. The index of a pixel defines its position in the image. For a QVGA image, the index is 0 for the pixel in the upper row and left column and 320x240-1 for the pixel in the lower row and right column. Once all the key points have been extracted from both images, the region matching process begins. During this process, all the combinations between the KPs of the left image with the KPs of the right image are analyzed. The matching of two regions is evaluated using the SAD algorithm. The two main processes, key point extraction and matching are detailed in the following subsections.

A. Key point extraction

As it has been explained, the first stage of SIFT algorithm is used for the KP extraction. Fig 2 summarizes this part of the process. An input image G_0 is successively convolved with a gaussian function K_i of variance σ_i^2 to create a Gaussian Scale space (GSS). Then, each pair of adjacent gaussian images G_{i+1} and G_i are subtracted to obtain a Difference-of-Gaussians (DoG) space.

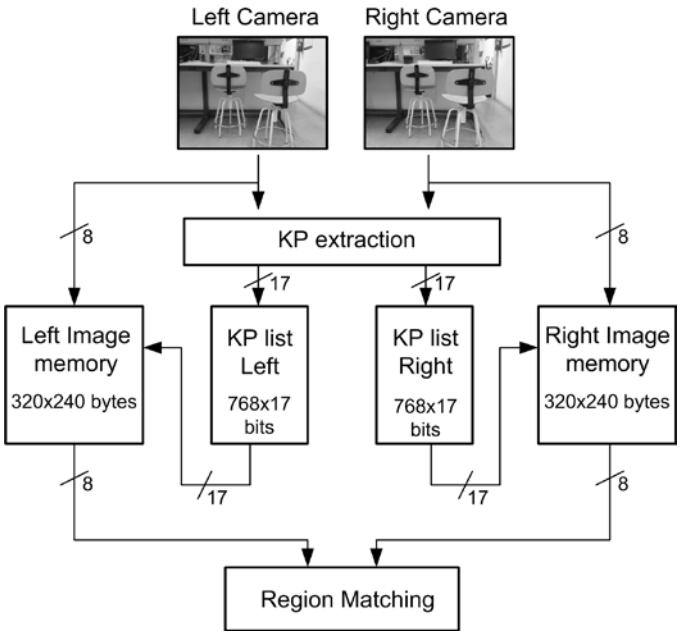

Fig. 1. General architecture of the proposed system for QVGA images.

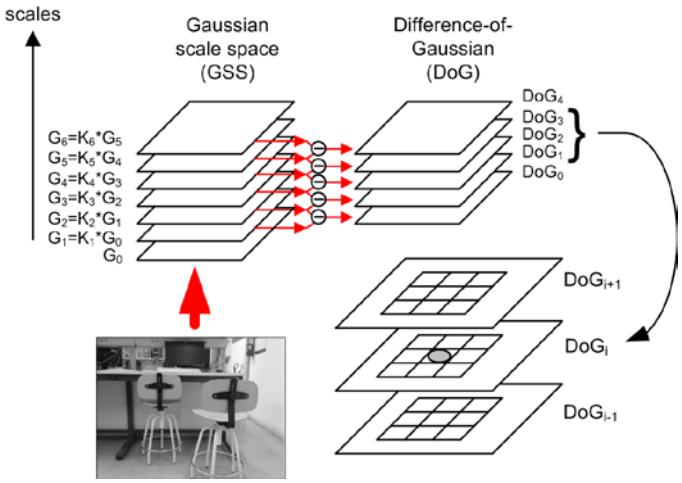

Fig. 2. Key Point extraction process in the SIFT algorithm.

According to [17] the proper values of σ_i, vary from 1.6 for the first convolution to 5.08 for the last one. Once the GSS and the DoG spaces have been created, every pixel in these spaces is identified by its *x,y* coordinates and its value of σ.

The next step is the identification of extrema points in the DoG space. This is achieved comparing each pixel of a given DoG with its 26 neighbors, 8 in the same DoG, 9 in the upper one and 9 in the lower one. If the pixel value is the maximum or the minimum value of all these pixel values, then this pixel is considered a KP candidate. As shown in Fig. 2, for a Gaussian pyramid with six Gaussian images, there are five DoGs and the extrema identification can be done over DoGs 1 to 3. Some of the KP candidates will be points with low contrast with respect to its neighbors. Thus, a threshold value is introduced for the determination of KPs. Hence, the absolute difference between a KP candidate and each one of its 26 neighbors must be greater than this threshold in order to consider the candidate pixel a real KP.

Among the different proposals which can be used to synthesize this part of the SIFT algorithm, in this work several techniques which reduce the amount of computational resources have been used [26]. The first technique consists on the use of recursive filtering from the first image instead of absolute filtering. As shown in Fig. 2, each gaussian image G_i in the GSS is obtained from its precedent gaussian G_{i-1}. This is called recursive filtering. This implies an increase in the latency of the operations, because the gaussian images would need their precedent images to be generated first. On the other hand, all gaussian images could be created from the original image (absolute filtering). This would allow to simultaneously generate all the Gaussian images in the GSS. Taking into account that

$$\sigma_{12}^2 = \sigma_1^2 + \sigma_2^2 \qquad (1)$$

it can be noted that, if σ_1, and σ_2 are the values of σ used to generate two adjacent gaussian images (i.e. G_1 and G_2), their values are lower than the value ofσ_{12}, the one required to generate the G_2 from the original image. Thus, the use of recursive filtering requires smaller values of σ. Given that the gaussian mask size is proportional to the value of σ [14], this means that also smaller mask sizes are required, as shown in Table I. This allows the use of fewer hardware registers and also fewer processing units. A second technique consists on exploiting the separability of gaussian filters [27]. This property allows to decompose a two-dimensional NxN Gaussian filtering into two cascaded one-dimensional Gaussian filters. So, the use of this property reduces the number of multipliers from N*N to N+N, and also allows the reuse of intermediate results.

To effectively synthesize the recursive filtering stage, a pipelined schema has been implemented. Fig. 3 shows the structure of a pipeline structure which exploits parallelism between the successive filtering stages. All the filtering operations, (Gauss), Difference of Gaussians operations (DoG) and neighborhood evaluation to detect extrema points (Neigh) generate a valid pixel value per clock cycle. The first filtering operation (Gauss1) performed on the left image requires 1619 cycles to output the first valid pixel, and then 320x240 additional cycles to complete the initial image filtering and obtain G_1. The second filtering operation (Gauss 2) over this same image starts 1,294 cycles later, when the top four lines of G_1 are completed. As it is shown in the figure, each consecutive filtering operation, from Gauss 2 to Gauss 6, begins with an increasing delay, because the respective mask sizes are bigger (Table I). This means that more lines are needed to be generated in G_i before the next Gaussian operation starts. This same pipeline structure allows that the first DoG operation (DoG1) begins when the first pixel of G_2 is available.

TABLE I. ABSOLUTE FILTERING VS RECURSIVE FILTERING

Scale	Absolute		Recursive	
	σ	*Mask size*	σ	*Mask size*
6	5.08	31	3.0900	15
5	4.03	25	2.4525	13
4	3.20	19	1.9466	11
3	2.54	15	1.5450	9
2	2.02	13	1.2263	7
1	1.6	9	1.6	9

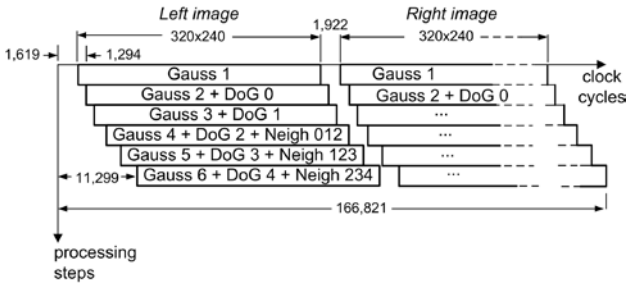

Fig. 3. Chronogram of the pipelined first stage of the SIFT algorithm.

Then the successive DoG operations begin successively and, when the first pixel of the DoG2 is available, the comparison between pixels in DoG1 and its neighbors in DoG0 and DoG2 begins. This KP extraction process ends after 88,099 clock cycles. Compared to the alternative of using absolute filtering, the recursive filtering implies a certain latency in the pipeline structure which, however, is negligible compared to the throughput achieved.

This same processing is applied to the image captured by the right camera. However, due to the structure of the pipeline implemented, this operation begins after 1,922 cycles once the Gauss1 operation on the left image is finished. The whole processing of both images takes 166,821 cycles (1,668 ms with a 100 MHz system clock).

B. Region Matching

Region matching begins once the KP extraction process of the right image has finished. Every KP in the right image is matched with every KP in the left image. This yields a total number of NKP^2 operations. The matching algorithm used has been the Sum of Absolute Differences (SAD), which is given by:

$$SAD = \sum_{i=1}^{N}\sum_{j=1}^{M}\left|I_{i,j} - R_{i,j}\right| \qquad (2)$$

where $I_{i,j}$ is the intensity of a given pixel of the analyzed block of image and $R_{i,j}$ is the intensity of a given pixel in the reference block, as shown in Fig 4. A null value of SAD means a perfect match between the image block and the reference block.

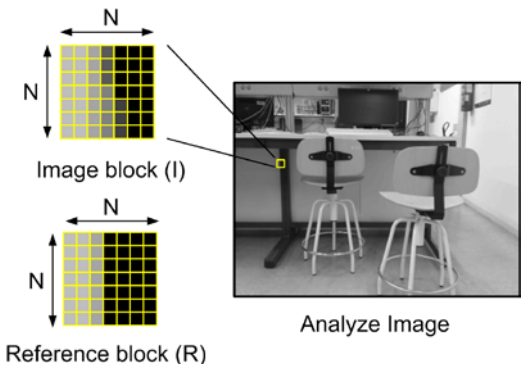

Fig. 4. Analyzed image (I) and reference image or block (R).

The size of the reference block can vary, usually from 3x3 pixels to greater values. The bigger the size is, the more accurate results will be obtained, but also, a higher hardware complexity and/or a longer processing time will be required. In this work several sizes have been tested as it will be discussed in next section, being the 7x7 the smallest one. The procedure is the following: The first KP index is read from the KP list of the left image (Fig. 1), and then, the pixels which compose the block reference centered in that KP are read from the left image memory. This will be the reference block *R*. *R* is then matched against the region centered in the first KP of the right image and the corresponding value of SAD is obtained. This procedure is repeated for all the KPs in the right image. When this process ends, the second KP from the left image is taken as reference block index and so on. For a 7x7 window size, the matching operation requires 7x7 cycles to read a given KP window from the left image memory and three more cycles due to intermediate buffers, which yields 52 cycles. 52 cycles are also required to read each KP image block from the right image memory. Thus, the match of all KPs in the left image with all the KPs in the right image would require 52 x NKP^2 cycles. For QVGA size images, and taking into account that the number of KPs typically represents the 1% of the pixels in the image [14], NKP would be 768 pixels.

This means a maximum of 30,670,848 cycles to perform the matching operation between both images. For a 100 MHz FPGA, this is equivalent to 306.71 ms, which yields a processing rate of approximately 3 frames per second (fps), far away from real time requirements. Hence, the parallelization of this stage is required, in order to perform several matching operations simultaneously. In this case, the required parallelism degree for the matching stage in order to achieve processing rates above 30 fps is 10.

III. Results

The proposed hardware architecture has been prototyped on a lower power Virtex 6 FPGA (XC6VLX75TL). The image size employed for the stereo pair was QVGA (320x240). The Gaussian pyramid was built using the values of σ and mask size shown in Table I for recursive filtering operations. The threshold to discard key point candidates with low contrast was set after considering different issues. In our tests with different set of images, the best results were obtained for threshold values within the range 0.03 to 0.06. On the other hand, the recommended value for the threshold value employed in the extrema detection is 0.03 [17]. Finally, the first significant digit of most of these values in a binary representation is the fifth one. So, in order to obtain an efficient hardware implementation the threshold was set to 2^{-5} (0.03125).

A fixed point data format has been used to perform all arithmetic operations. It uses 8 bits for the integer part. The minimum number of bits for the decimal part was defined according to the size of the threshold mentioned previously. So a five decimal bit format was finally implemented.

Different tests were carried out to check the hardware proposed. Fig. 5 shows a pair of stereo images captured from an structured indoor environment. The key points are displayed as dots in each one of the images. The number of key points extracted from the left and from the right image has been 89 and 105 respectively. For a 7x7 pixel size window, the number of matches has been 37. Fig. 6 shows the matches between both left and right images. These matches were obtained after analyzing the SAD values for every combination of KPs from both stereo images.

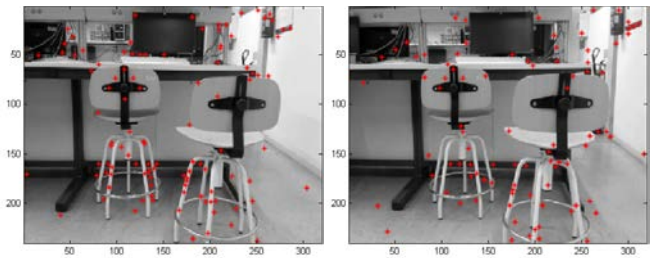

Fig. 5. Key points extracted from feft and.right images

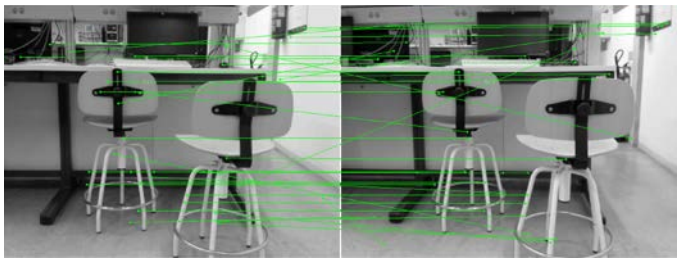

Fig. 6. Key points matched between both images.

TABLE II. NUMBER OF KEYPOINTS AND MATCHES FOR DIFFERENT SIZES OF THE MATCHING WINDOW

Window size	Number of KP		Matches
	Even	Odd	
7x7	89	105	37
9x9	87	102	39
11x11	84	100	36
13x13	84	100	41
15x15	84	100	41

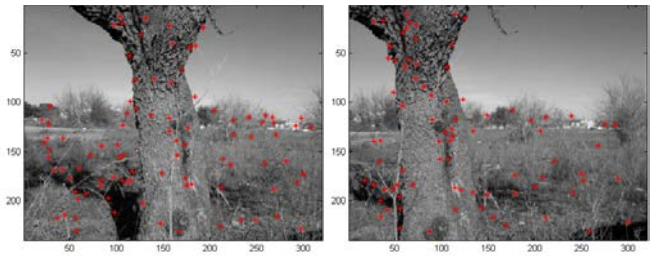

Fig. 7. Key points extracted from feft and.right images

Fig. 8. Key points matched between both images.

Table II details the number of KP extracted for left and right images as well as the number of matches for different sizes of the matching window for the images shown in Fig. 5. The number of KPs detected decreases for increasing window sizes because the pixels between the image border and the mask radius are discarded.

Unstructured environments have also been tested. Fig. 7 shows a pair of stereo images captured from an outdoor unstructured environment. The matching between both images is shown in Fig. 8. The number of KPs extracted from the left and from the right image has been 87 and 96 respectively. For a 7x7 pixel size window, 23 matches were found. The number of KP extracted for even and odd images as well as the number of matches for different sizes of the matching window is shown in Table III. Table IV details the device utilization. The FPGA is the smallest of the low power Virtex 6 series. The second column shows the values when no parallelization is used. The third column shows the utilization when a parallelization x10 is used for the matching stage. It can be noted that the parallelization mainly requires slice registers and LUTS resources, while complex units as extra DSPs are not needed. This is because they are used only to compute the multiplications of the Gaussian filtering of the first stage.

TABLE III. NUMBER OF KEYPOINTS AND MATCHES FOR DIFFERENT SIZES OF THE MATCHING WINDOW

Window size	Number of KP		Matches
	Even	Odd	
7x7	87	96	23
9x9	84	94	22
11x11	83	94	25
13x13	82	94	25

15x15	81	93	25

TABLE IV. DEVICE UTILIZATION SUMMARY (XC6VLX75TL)

Logic Utilization	**No parallel**	**Parallel x 10**	**Available**
Slice Registers	4782 (5%)	5223 (5%)	93120
Slice LUTs	6953 (14%)	8042 (17%)	46560
fully used LUT-FF pairs	1732 (17%)	2002 (20%)	10086
Block RAM/FIFO	112 (71%)	112 (71%)	156
BUFG/BUFGCTRL/	1 (3%)	1 (3%)	32
DSP48E1s	128 (44%)	128 (44%)	288

IV. CONCLUSION

In this work, a stereo matching hardware architecture for autonomous mobile robots, has been described. The architecture combines the ability of the SIFT algorithm to extract characteristic points from images with a window-based matching algorithms.

The solution proposed is, one the one hand, a better parallelizable architecture than the original SIFT, and on the other, a faster technique than a full image windows matching approach. The architecture has been synthesized on a XC6VLX75TL FPGA. This is the lower power device of the Virtex 6 series. It is a very suitable FPGA to be employed by small mobile robots which require low power dissipation. The use of the resources of the FPGA allows for further parallelization of the matching process. This would allow to increase the number of key points or even the image size in future developments.

The hardware implementation has been tested for an structured indoor environment and for an unstructured outdoor one. In both tests, the matching has exhibit good results in comparison with software testbenches.

ACKNOWLEDGMENT

This work has been funded by Spanish government project TEC2015-66878-C3-2-R (MINECO/FEDER, UE).

REFERENCES

[1] Heiko Hirschmu, ller and Daniel Scharstein,, “Evaluation of Stereo Matching Costs on Images with Radiometric Differences”, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 9, pp 1852-1599, September 2009.

[2] R. Klette *et al*., "Performance of Correspondence Algorithms in Vision-Based Driver Assistance Using an Online Image Sequence Database," *IEEE Transactions on Vehicular Technology*, vol. 60, no. 5, pp. 2012-2026, June 2011.

[3] T. Kanade, H. Kano, S. Kimura, A. Yoshida and K. Oda, "Development of a video-rate stereo machine," *Proc. IEEE/RSJ International Conference on Intelligent Robots and Systems. Human Robot Interaction and Cooperative Robots*, Pittsburgh, PA, 1995, pp. 95-100.

[4] M. J. Hannah, “Computer matching of areas in stereo images.” Ph.D. dissertation, Stanford, CA, USA, 1974, aAI7427032

[5] A. Broggi, M. Buzzoni, M. Felisa and P. Zani, "Stereo obstacle detection in challenging environments: The VIAC experience," *IEEE/RSJ International Conference on Intelligent Robots and Systems*, San Francisco, CA, 2011, pp. 1599-1604.

[6] G. Saygili, L. v. d. Maaten and E. A. Hendriks, "Stereo Similarity Metric Fusion Using Stereo Confidence," *22nd International Conference on Pattern Recognition*, Stockholm, 2014, pp. 2161-2166.

[7] K. Ambrosch, W. Kubinger, M. Humenberger and A. Steininger, "Hardware implementation of an SAD based stereo vision algorithm," *2007 IEEE Conference on Computer Vision and Pattern Recognition*, Minneapolis, MN, 2007, pp. 1-6

[8] K. Häublein, M. Reichenbach and D. Fey, "Fast and generic hardware architecture for stereo block matching applications on embedded systems," *2014 International Conference on ReConFigurable Computing and FPGAs (ReConFig14)*, Cancun, 2014, pp. 1-6.

[9] N. Isakova, S. Başak and A. C. Sönmez, "FPGA design and implementation of a real-time stereo vision system,"2*012 International Symposium on Innovations in Intelligent Systems and Applications*, Trabzon, 2012, pp. 1-5.

[10] Ngo Huy Tan, N. H. Hamid, P. Sebastian and Yap Vooi Voon, "Resource minimization in a real-time depth-map processing system on FPGA," *TENCON 2011 - 2011 IEEE Region 10 Conference*, Bali, 2011, pp. 706-710.

[11] P. M. Santos and J. Canas Ferreira, "FPGA-based real-time disparity computation and object location," *NORCHIP 2010*, Tampere, 2010, pp. 1-4.

[12] P. Sadeghi-Tehran, C. Clarke and P. Angelov, "A real-time approach for autonomous detection and tracking of moving objects from UAV," *IEEE Symposium on Evolving and Autonomous Learning Systems (EALS)*, Orlando, FL, 2014, pp. 43-49.

[13] Xia, Xin and Gai, Shaoyan, "An Improved Dense Matching Algorithm for Face Based on Region Growing", in "Computer Engineering and Networking: Proceedings of the 2013 International Conference on Computer Engineering and Network (CENet2013)", 2014, Springer International Publishing, Cham, pp. 625-633,

[14] W. Li, H. Yao, R. Ji, P. Xu, X. Liu and D. Zhao, "Robust Stereo Matching Combining SIFT Descriptor with NCC under MRF Framework," *2010 First International Conference on Pervasive Computing, Signal Processing and Applications*, Harbin, 2010, pp. 1018-1021

[15] H. Yang, M. Yu and S. Zhang, "Wide baseline stereo matching based on scale invariant feature transformation with hybrid geometric constraints," in *IET Computer Vision*, vol. 8, no. 6, pp. 611-619, 12 2014.

[16] C. Wang, J. Chen and D. He, "A Sub-pixel registration method of UAV oblique images," *2016 IEEE International Geoscience and Remote Sensing Symposium (IGARSS)*, Beijing, 2016, pp. 7358-7361.

[17] D. Lowe, "Distintive image features from scale-invariant key-points", *Int. J. Compt. Vis.*, vol. 60, no. 2, pp 91-110, November 2004.

[18] V. Bonato et al, "A Parallel Hardware Architecture for Scale and Rotation Invariant Feature Detection," IEEE Trans. on Circ. and Syst. for Video Techn., vol.18, no.12, pp. 1703-1712, Dec. 2008.

[19] L. Yao et al, "An architecture of optimised SIFT feature detection for an FPGA implementation of an image matcher," Intl. Conf. on Field-Programmable Technology, 2009. FPT 2009, pp.30-37, 9-11 Dec. 2009.

[20] M. Qasaimeh, A. Sagahyroon and T. Shanableh, "FPGA-Based Parallel Hardware Architecture for Real-Time Image Classification," in IEEE Trans. on Computational Imaging, vol. 1, no. 1, pp. 56-70, March 2015

[21] F.C. Huang, et al, "High-Performance SIFT Hardware Accelerator for Real-Time Image Feature Extraction," IEEE Trans. on Circ. and Syst. for Video Tech., vol.22, no.3, pp. 340-351, Mar. 2012.

[22] L. C. Chiu et al, "Fast SIFT Design for Real-Time Visual Feature Extraction," in IEEE Trans. on Image Processing, vol. 22, no. 8, pp. 3158-3167, Aug. 2013.

[23] K. Mizuno et al., "Fast and low-memory-bandwidth architecture of SIFT descriptor generation with scalability on speed and accuracy for VGA video," in Proc. IEEE FPL, 2010, pp. 608–611.

[24] W. Deng et al, "An efficient hardware architecture of the optimised SIFT descriptor generation," in Proc IEEE FPL, 2012, pp. 345-352.

[25] J. Jiang, X. Li and G. Zhang, "SIFT Hardware Implementation for Real-Time Image Feature Extraction," in IEEE Transactions on Circuits and Systems for Video Technology, vol. 24, no. 7, pp. 1209-1220, July 2014.

[26] G. Doménech-Asensi, .J. Garrigós, P. López, V. Brea and D. Cabello " Real time architectures for the Scale Invariant Feature Transform algorithm ," *15th International Workshop on Cellular Nanoscale Networks and their Applications*, August 23-25, 2016, Dresden, Germany , pp. 75-76.

[27] R. Jain, R. Kasturi, and B. G. Schunck, *Machine Vision*. McGraw-Hill, Inc., 1995.