IUST
Iran University of Science and Technology

Iranian Journal of Electrical and Electronic Engineering

Journal Homepage: ijeee.iust.ac.ir

Clifford Algebra's Geometric Product Properties in Image-Processing and its Efficient Implementation

A. Sadr*(C.A.) and N. Orouji*

Abstract: Clifford Algebra (CA) is an effective substitute for classic algebra as the modern generation of mathematics. However, massive computational loads of CA-based algorithms have hindered its practical usage in the past decades. Nowadays, due to magnificent developments in computational architectures and systems, CA framework plays a vital role in the intuitive description of many scientific issues. Geometric Product is the most important CA operator, which created a novel perspective on image processing problems. In this work, Geometric Product and its properties are discussed precisely, and it is used for image partitioning as a straightforward instance. Efficient implementation of CA operators needs a specialized structure, therefore a hardware architecture is proposed that achieves 25x speed-up in comparison to the software approach.

Keywords: Clifford Algebra, Image Processing Visualization, Geometric Product, Hardware Implementation.

1 Introduction

CLIFFORD Algebra (CA) first introduced by W. K. Clifford in the 19th century in order to unify exterior algebra and quaternions introduced by Grassmann and Hamilton, respectively [1]. This unification made CA framework a powerful tool, and its intuitive representations solve the problems in a novel way. Nowadays, researchers deploy CA framework in computer vision to improve their prospect, visualize the problem, and suggest the best solutions [2]. Recently, image processing has become an active topic in computer vision research fields e.g. color image edge detection [3], image segmentation, and registration [4]. Intuitive color representation and comprehensible operators in CA framework present a whole new world for interaction with image-related problems. Quaternion atomic function [5], Clifford Fourier transform [6, 7], filter-based approaches [8], and image analysis in the frequency domain [9] are amongst powerful solutions in this framework.

Iranian Journal of Electrical and Electronic Engineering, 2019.
Paper first received 17 July 2018 and accepted 26 August 2018.
* The authors are with the Department of Electrical Engineering, Iran University of Science and Technology (IUST), Tehran, Iran.
E-mails: sadr@iust.ac.ir and niloofar_orouji@yahoo.com.
Corresponding Author: A. Sadr.

Despite undeniable avails and various applications, Clifford Algebra has a massive computational load. Along with significant improvements of computational softwares and systems, novel architectures for efficient implementation of CA framework have been proposed. Clucalc dedicated software [10], Gaigen software library [11], Gaalop pre-complier [12], various co-processors [4, 13-16], and specialized hardware architectures [17] are amongst these novel implementation approaches. Software-based approaches use general-purpose CPUs to execute CA operators. This might be applicable in case of simple simulations and educational purposes, however, it cannot provide enough computational resources for more complicated algorithms, such as edge detection. Therefore, hardware implementations that provide more computational capacity, are superior solutions. Almost all CA co-processors, namely S-CliffoSor [18], CliffordALU [13], ConformalALU [4], and GA co-processor [16], exploited popular FPGA platform and dedicated architectures to perform expected operations [19].

Geometric product (GP) is the most important operator of the CA framework, which performs general geometric operations e.g. reflection, rotation, and translation [20]. In this work, properties and benefits of GP in image processing is discussed, and a hardware architecture for its efficient execution is proposed. This paper is organized as follows: Clifford Algebra three-

dimensional space and Geometric Product will be discussed in Section 2. Performing GP on color pixels, mathematical descriptions, and proposed hardware architecture will be the main topics of Section 3. Results of applying GP on color images and consumed computational resources are presented in Section 4. Discussion and conclusion have been made in Section 5.

2 Clifford Algebra Three-Dimensional Vector Space

Clifford algebra is an immense space of multidimensional entities and multitude operators, therefore it's very important to choose best fitting space and operations according to problem requirements. In image processing problems, the first issue is color image information descriptions, and usually, the simplest solution is the RBG or CMY color system. According to these best-known description systems, the best choice for expressing color information is CA three-dimensional space with $\{\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3\}$ basis vectors [21]. This space is denoted by $\mathbb{R}_{3,0}$, which means three basis vectors are squared to 1 and none of them is squared to -1 [4]. Each CA n-dimensional space consists of 2^n elements, which in this case $\mathbb{R}_{3,0}$ has 2^3 elements:

$$\{1, \mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3, \mathbf{e}_{12}, \mathbf{e}_{23}, \mathbf{e}_{31}, \mathbf{e}_{123}\}$$

In this notation, 1 stands for all scalar values of $\mathbb{R}_{3,0}$ space and usually shows the amount of black color in each pixel. The coefficients of $\mathbf{e}_1$, $\mathbf{e}_2$ and $\mathbf{e}_3$ basis vectors describe amounts of red, green and blue colors, respectively and represents the RGB color system. Subsequently, $\mathbf{e}_{12}$, $\mathbf{e}_{23}$ and $\mathbf{e}_{31}$ are called bivectors and their coefficients, respectively express the amounts of yellow, cyan, and magenta colors. These three bivectors are the representative of the CMY color system. Finally, the $\mathbf{e}_{123}$ element of $\mathbb{R}_{3,0}$ is called trivector and expresses the amount of white color. Geometrical representation of these elements is depicted in Fig. 1 using GAviewer 0.85 software [22]. Bivectors (oriented areas) and trivectors (oriented volumes) are obtained by performing wedge products (^) on basis vectors:

$$\mathbf{e}_i \wedge \mathbf{e}_j = \mathbf{e}_{ij} \ (i \neq j) \quad (1)$$

$$\mathbf{e}_i \wedge \mathbf{e}_j \wedge \mathbf{e}_k = \mathbf{e}_{ijk} \ (i \neq j \neq k) \quad (2)$$

2.1 Multivectors and Color Representation

Linear combination of $\mathbb{R}_{3,0}$ vector space elements results in a multivector. General formation of a multivector is expressed in (3).

$$\mathbf{M} = \mathbf{P} + r\mathbf{e}_1 + g\mathbf{e}_2 + b\mathbf{e}_3 + y\mathbf{e}_{12} + c\mathbf{e}_{23} + m\mathbf{e}_{31} + Q\mathbf{e}_{123} \quad (3)$$

Mostly, more than half of coefficients in (3) are zeros, for instance in RGB color representation only r, g, and b coefficients are non-zero. As an example, the gray

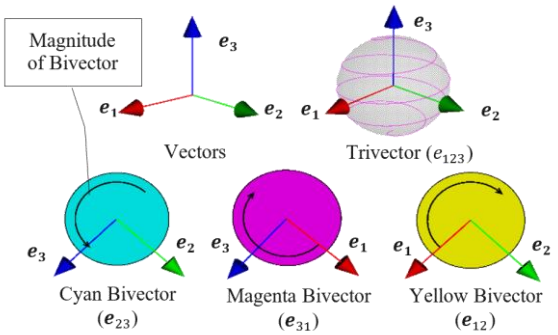

Fig. 1 Demonstration of $\mathbb{R}_{3,0}$ elements.

color consists of equal amounts of red, green, and blue, therefore it can be expressed by:

$$\boldsymbol{\mu} = \frac{1}{\sqrt{3}}(\mathbf{e}_1 + \mathbf{e}_2 + \mathbf{e}_3) \quad (4)$$

where the $1/\sqrt{3}$ coefficient is the magnitude of multivector $\boldsymbol{\mu}$.

Considering an $M \times N$ image, each pixel information in RGB format is expressed in (5). In a similar way, color information in the CMY system is defined based on bivectors and shown in (6).

$$\mathbf{I}(x,y) = r(x,y)\mathbf{e}_1 + g(x,y)\mathbf{e}_2 + b(x,y)\mathbf{e}_3 \quad (1 \le x \le N \text{ and } 1 \le y \le M) \quad (5)$$

$$\mathbf{I}(x,y) = y(x,y)\mathbf{e}_{12} + c(x,y)\mathbf{e}_{23} + m(x,y)\mathbf{e}_{31} \quad (1 \le x \le N \text{ and } 1 \le y \le M) \quad (6)$$

2.2 Geometric Product

Consider two multivectors namely $\mathbf{u}$ and $\mathbf{v}$. Addition (or subtraction) of these multivectors is equivalent to each element's corresponding coefficient addition (or subtraction), however, the multiplication case is different. Geometric product (GP) performs multiplication of two multivectors, using an inner product and a wedge product.

$$\mathbf{uv} = \mathbf{u}.\mathbf{v} + \mathbf{u} \wedge \mathbf{v} \quad (7)$$

In (7), $\mathbf{u}.\mathbf{v}$ denotes the inner product of two multivectors which is peer to peer multiplication of coefficients. The $\mathbf{u}\wedge\mathbf{v}$ term is the wedge product and is similar to the outer product in classic geometries [23]. If $\mathbf{u}$ and $\boldsymbol{v}$ denote two multivectors in the RGB system, then GP of these two will consist of a scalar term and several bivectors. Depending on the nature of $\mathbf{u}$ and $\mathbf{v}$, these two parts of GP can be interpreted differently, which will be discussed completely in the next section.

3 Geometric Product of Color Images and Different Multivectors: Properties and Implementation

It is shown that GP of every single pixel of a color

image and **μ** multivector (4), results in image intensity and achromatic information [3]. The scalar part of GP is the projection of each pixel's multivector on the **μ** multivector, thus it represents the value or intensity map of the image. On the other hand, pixels with lower chromaticity are closer to the gray multivector, therefore they will have less bivector magnitude. In [3], the bivector part's magnitude is exploited as a mask for full recognition of achromatic areas, however, we will prove that the avails of the bivector part are more than a mask.

3.1 GP's Bivector Component as a Map for Image Partitioning

According to properties of GP, the closer multivectors' GP will result in the less bivector magnitude, which is depicted in Fig. 2. In Fig. 2 the yellow and purple arrows represent two general multivectors, and the cyan disk is the bivector part of GP between these two multivectors (an oriented area). As mentioned before, the swirl length shows the magnitude of the resultant bivector, the closer multivectors have the less swirl length. This exclusivity can be exploited for partitioning images into desired color regions. As if we multiply the image multivectors in a specific color multivector through GP, the lower magnitude of the bivector part will exhibit more closeness to the specified color.

In addition, pixels with low intensity have a low bivector magnitude and are not necessarily close to the specified color, therefore the scalar part of GP is used to discriminate them. Based on both scalar and bivector information, the pixel with low bivector magnitude and low scalar coefficient is neglected, while the pixel with low bivector magnitude and high scalar coefficient is definitely close to the specified color.

3.2 The Proposed Hardware Architecture for GP Efficient Implementation

Considering **I**(*x*,*y*) defined in (5) as a color pixel multivector, and a specific color multivector called **γ**, the general GP of these two multivectors in RGB color system according to (7) will be expressed as (8); where γ_1, γ_2, and γ_3 are coefficients of multivector **γ**, and $\sqrt{\gamma_1^2+\gamma_2^2+\gamma_3^2}$ is its norm. According to (8) the scalar part and the bivector part's magnitude is defined as (9) and (10), respectively.

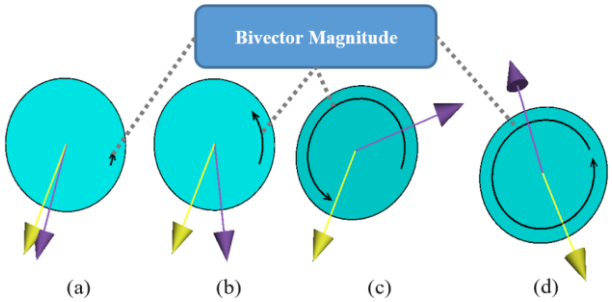

Fig. 2 Magnitude of GP's Bivector part in different multivectors' positions drawn by GAviewer: a) & b) Two close multivectors, representative of two close colors, will result in low bivector magnitude; c) & d) Two completely different colors result in high bivector magnitude.

The proposed hardware architecture, based on (9) and (10) is depicted in Fig. 3. The architecture consists of two main units: Scalar Unit and Bivector Unit. For more generality and flexibility, each one has its individual control unit which is supervised by the main control unit, which steers data flows and checks data validations on different input and output buses.

Scalar Unit data flow is straightforward to understand: corresponding coefficients are multiplied in the first stage and results are added through ADD1 and ADD2, then the final result is divided by the **γ**'s norm which is calculated by the Norm Unit shown in Fig. 4. The 16×32bit dual-port RAM is used to store the third multiplication results, and its output and input addresses are controlled by Scalar Control Unit. Bivector Unit, calculates the magnitude of GP's bivector part according to (10), in a similar way as Scalar Unit. It

$$\begin{aligned}\mathrm{GP}\left(\mathbf{I}(x,y),\boldsymbol{\gamma}\right)&=\left(\frac{1}{\sqrt{\gamma_1^2+\gamma_2^2+\gamma_3^2}}\right)\left[r(x,y)\mathbf{e}_1+g(x,y)\mathbf{e}_2+b(x,y)\mathbf{e}_3\right]\left[\gamma_1\mathbf{e}_1+\gamma_2\mathbf{e}_2+\gamma_3\mathbf{e}_3\right]\\&=\left(\frac{1}{\sqrt{\gamma_1^2+\gamma_2^2+\gamma_3^2}}\right)\Big[\left(r(x,y)\gamma_1+g(x,y)\gamma_2+b(x,y)\gamma_3\right)+\left(r(x,y)\gamma_2-g(x,y)\gamma_1\right)\mathbf{e}_{12}\\&\quad+\left(g(x,y)\gamma_3-b(x,y)\gamma_2\right)\mathbf{e}_{23}+\left(b(x,y)\gamma_1-r(x,y)\gamma_3\right)\mathbf{e}_{31}\Big]\end{aligned}\tag{8}$$

$$\mathrm{Sca}(x,y)=\left(\frac{1}{\sqrt{\gamma_1^2+\gamma_2^2+\gamma_3^2}}\right)\left[\left(r(x,y)\gamma_1+g(x,y)\gamma_2+b(x,y)\gamma_3\right)\right]\tag{9}$$

$$\begin{aligned}\left|\mathrm{Biv}(x,y)\right|&=\left(\frac{1}{\sqrt{\gamma_1^2+\gamma_2^2+\gamma_3^2}}\right)\times\\&\left[\left(r(x,y)\gamma_2-g(x,y)\gamma_1\right)^2+\left(g(x,y)\gamma_3-b(x,y)\gamma_2\right)^2+\left(b(x,y)\gamma_1-r(x,y)\gamma_3\right)^2\right]^{\frac{1}{2}}\end{aligned}\tag{10}$$

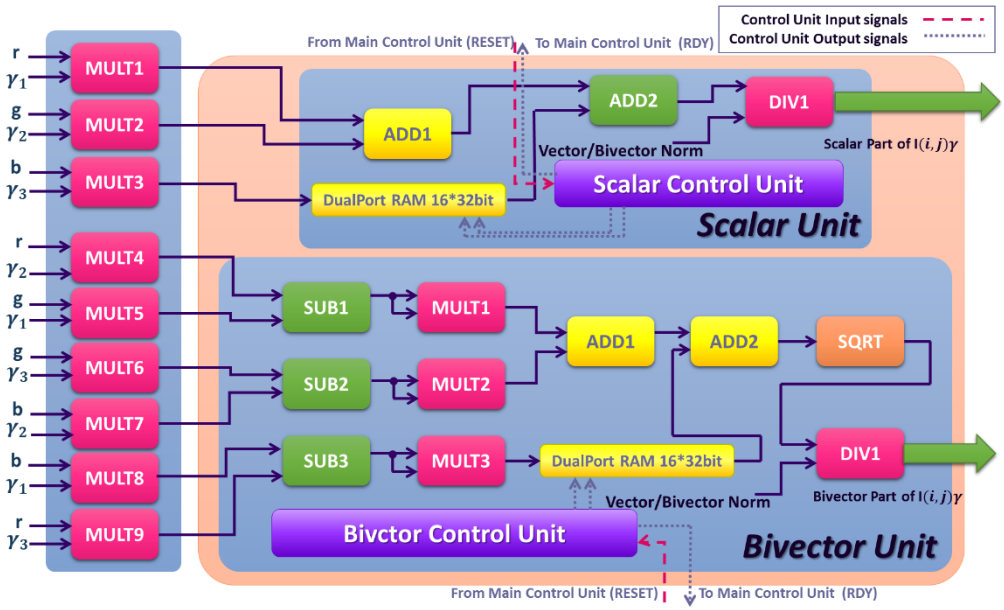

Fig. 3 The proposed hardware architecture for GP efficient implementation.

should be noted that the proposed architecture is able to calculate GP for multivectors in form of (6) with no further changes.

4 Simulations and Implementation's Specifications

In this section, the results of GP between a color image and a specific color vector demonstrates its ability to categorize an image into different color regions. On the other side, run times of the proposed hardware architecture is compared to C++ software runtimes. Finally, last subsections exhibit resource usage of this architecture, and a comparison to GA co-processor.

4.1 GP Between Color Images and Specific Multivectors

As explained in Section 3, two close multivectors will result in low GP's bivector magnitude. To demonstrate this assertion, we will operate GP between the well-known tulip image and the yellow color multivector, and analyze the results. The $\boldsymbol{\gamma}$ multivector defines the mentioned yellow color as:

$$\boldsymbol{\gamma} = 0.83(0.9\mathbf{e}_1+0.8\mathbf{e}_2) \tag{11}$$

where γ_1=0.9, γ_2=0.8, and $\dfrac{1}{\sqrt{\gamma_1^2+\gamma_2^2}} = 0.83$. The scalar part and bivector part's magnitude is obtained through the proposed hardware, and is shown in Figs. 5(b) and 5(c), respectively. In these images the lower feature will result in the darker areas, as an instance the center of each tulip's color is the same as the chosen one, therefore these areas in Fig. 5(c) are darker than the others which shows their lower bivector part's magnitude. There are many dark regions in Fig. 5(c) which their colors are not close to the $\boldsymbol{\gamma}$ multivector. The reason behind their low magnitude is their low intensity. The scalar part shown in Fig. 5(b) contains intensity information, therefore it is the best mean to discriminate between low intensity regions and pixels close to the specified color. The final map of yellow color regions in the tulip image is shown in Fig. 5(d).

4.2 Run Times and Resource Usage

In order to attain GP's software run time, a routine is written in C++ and executed on a general-purpose processor, Intel Core™ i5- 2430M. The GP operation is executed 5×10^6 times to achieve the average run times. The proposed architecture is implemented on FPGA Virtex-5 XC5VFX200T and simulated using ISE Design Suite 13.1. All hardware run times are calculated based on the final design frequency which is 368.64 MHz and averaged using 5×10^6 experiments. Software and hardware execution speeds are listed in Fig. 6.

According to Fig. 6, the proposed hardware executes the GP operation almost 25x faster than the software approach.

It should be noted that the latency of each multiplier core in the first stage (Fig. 3) is considered on the corresponding unit. Computational cores, e.g. ADD and MULT cores, are implemented using LogiCore IP Floating-Point Operator v5.0. Resource usage and latency of each core and the proposed architecture are listed in Tables 1 and 2, respectively.

4.3 Performance Comparison With GA Co-Processor

One of the hardware implementations of CA framework in image processing field, is GA co-processor [15, 16]. Same as the proposed hardware, the co-processor exploits single precision numbers in IEEE 754 format. The architecture consists of three six-

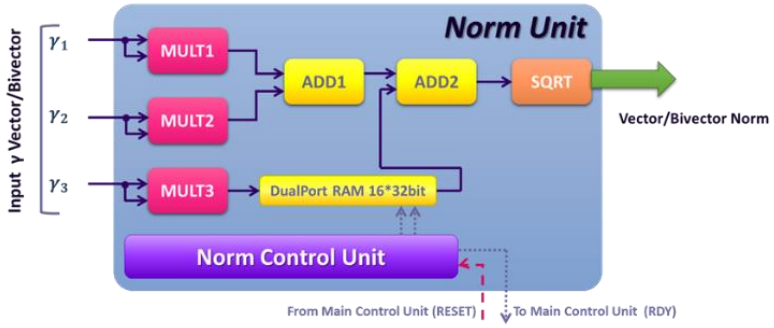

Fig. 4 The Norm Unit architecture.

(a) (b) (c) (d)

Fig. 5 GP between the tulip image and yellow multivector: a) The original image, b) Scalar part of GP, c) Bivector part of G, and d) Yellow regions of the image.

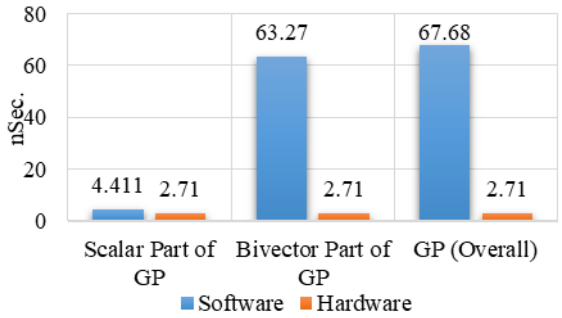

Fig. 6 Average run times for a single GP operation.

Table 1 Implementation method, resource usage, and latency of each computational core.

Computational Cores	Implementation Method	Resource Usage				Latency for First Operation [Cycles]	Frequency [MHz]
		Slice Registers	Slice LUTs	LUT FF Pairs	DSP48E		
ADD	Logic gates	549	427	643	-----	12	373.134
SUB	Logic gates	549	427	643	-----	12	373.134
MULT	Two slices of DSP48E	188	99	221	2	8	373.134
DIV	Logic gates	1352	758	1429	-----	28	380.517
SQRT	Logic gates	808	491	901	-----	28	380.517

Table 2 Resource usage and latency of the proposed hardware architecture.

Units	Resource Usage				DualPort RAM 16×32bit	Latency for First Operation [Cycles]
	Slice Registers	Slice LUTs	LUT FF Pairs	DSP48E		
Scalar Unit	1899	1295	2241	8	1	39
Bivector Unit	5483	3667	6289	20	1	88
Norm Unit	2470	1642	2848	6	1	59
Total	9949	6610	11384	34	3	88

stage pipelined adder, two five-stage pipelined multiplier and various control units to supervise its state machine. The co-processor has three different versions. In full multivector version, the co-processor operates GP regardless of whether the coefficients are zero or not. Obviously it takes more processing cycles and resources. As an optimization for image processing purposes, the single core version is presented, which is eliminated unnecessary operations to increase the design's speed. Also, the dual core version is presented for faster execution.

The speedups of the proposed hardware in comparison

Table 3 Comparison of required clock cycles and run times.

GP Implementation	Image Size						Freq. [MHz]	Average Speedup (Compared to 125MHz version)	Average Speedup (Compared to 368.64MHz version)
	128×128		256×256		512×512				
	Cycles	Run Time [ms]	Cycles	Run Time [ms]	Cycles	Run Time [ms]			
Full Multivector [16]	1.42×10^6	11.4	5.97×10^6	46	2.29×10^7	184	125	87.9	271.5
ASIC with Single Core [15]	1.24×10^6	9.5	4.97×10^6	38.3	1.99×10^7	153	130	73.1	226
ASIC with Two Cores [15]	6.37×10^5	4.9	2.55×10^6	19.6	1.02×10^7	78.6	130	37.6	116
This Work	1.63×10^4	0.13	6.55×10^4	0.52	2.62×10^5	2.1	125	----	----
This Work	1.63×10^4	0.04	6.55×10^4	0.17	2.62×10^5	0.71	368.64	----	----

Table 4 Resource usages comparison between different implementations.

Implementation	Target Device	No. of Cells
GA Co-processor [15]	ASIC Prototype (Single Core)	35,355
This Work	Xilinx XC5VFX200T	87,368

to different versions of GA co-processor are presented in Table 3. The results show that the proposed hardware is 37.6 times faster than the dual core version at almost the same frequency. The Table 4 exhibits the resource usage of the design in term of used cells. The resource usage of the proposed hardware is only 2.5x more than single core version, which is sensible in comparison to the achieved speedups.

5 Discussion and Conclusion

Clifford algebra is one of the most powerful mathematical tools in visualizing problems in many research fields such as computer vision and image processing. The $\mathbb{R}_{3,0}$ vector space of this framework is the best fit for mapping color description systems into geometric entities, such as vectors, bivectors (oriented areas), and trivectors (oriented volumes). Besides, the geometric product is an effective CA operator, which performs many geometrical operations, e.g. projection and reflection. Therefore, the combination of images, specific color multivectors, and GP leads to novel approaches of image processing, especially in image partitioning. The GP between image and specific colors multivectors are subjected in this work, and the resultant images were analyzed precisely. Finally, a specialized architecture is suggested for its efficient implementation, which executes the GP operations 25x faster than the software approach.

References

[1] G. Sommer, *Geometric computing with Clifford algebras*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2001.

[2] E. Bayro-Corrochano, "Geometric algebra for the twenty-first century cybernetics," in *Geometric Algebra Applications Vol. I*, Cham: Springer International Publishing, pp. 1–17, 2019.

[3] P. Carré, P. Denis, and C. Fernandez-Maloigne, "Spatial color image processing using Clifford algebras: application to color active contour," *Signal, Image and Video Processing*, Vol. 8, No. 7, pp. 1357–1372, Oct. 2014.

[4] S. Franchini, A. Gentile, F. Sorbello, G. Vassallo, and S. Vitabile, "ConformalALU: A conformal geometric algebra coprocessor for medical image processing," *IEEE Transactions on Computers*, Vol. 64, No. 4, pp. 955–970, Apr. 2015.

[5] L. Dorst and J. Lasenby, Eds., *Guide to geometric algebra in practice*. London: Springer London, 2011.

[6] T. A. Ell and S. J. Sangwine, "Hypercomplex Fourier transforms of color images," *IEEE Transactions on Image Processing*, Vol. 16, No. 1, pp. 22–35, Jan. 2007.

[7] T. A. Ell and S. J. Sangwine, "Hypercomplex Wiener-Khintchine theorem with application to color image correlation," in *Proceedings 2000 International Conference on Image Processing (Cat. No.00CH37101)*, Vol.2, pp. 792–795, 2000.

[8] S. J. Sangwine and T. A. Ell, "Mathematical approaches to linear vector filtering of colour images," in *First European Conference on Colour in Graphics, Imaging and Vision (CGIV 2002)*, pp. 348–351, 2002.

[9] P. Denis, P. Carre, and C. Fernandez-Maloigne, "Spatial and spectral quaternionic approaches for colour images," *Computer Vision and Image Understanding*, Vol. 107, No. 1–2, pp. 74–87, Jul. 2007.

[10] C. Perwass, "CLUCalc: Interactive visualization," 2010. [Online]. Available: http://www.clucalc.info. [Accessed: 05-Mar-2018].

[11] D. Fontijne, "Gaigen 2: A geometric algebra implementation generator," in *Proceedings of the 5th international conference on Generative programming and component engineering*, pp. 141–150, 2006.

[12] D. Hildenbrand, J. Pitt, and A. Koch, "Gaalop-high performance parallel computing based on conformal geometric algebra," *Geometric Algebra Computing in Engineering and Computer Science*, pp. 477–494, 2010.

[13] S. Franchini, A. Gentile, F. Sorbello, G. Vassallo, and S. Vitabile, "Design space exploration of parallel embedded architectures for native Clifford algebra operations," *IEEE Design and Test of Computers*, Vol. 29, No. 3, pp. 60–69, 2012.

[14] C. Perwass, C. Gebken, and G. Sommer, "Implementation of a Clifford algebra co-processor design on a field programmable gate array," pp. 561–575, 2003.

[15] B. Mishra, P. Wilson, and R. Wilcock, "A geometric algebra co-processor for color edge detection," *Electronics*, Vol. 4, No. 1, pp. 94–117, Jan. 2015.

[16] B. Mishra, M. Kochery, P. Wilson, and R. Wilcock, "A novel signal processing coprocessor for n-dimensional geometric algebra applications," *Circuits and Systems*, Vol. 5, No. 11, pp. 274–291, 2014.

[17] F. Stock, A. Koch, and D. Hildenbrand, "FPGA-accelerated color edge detection using a Geometric-Algebra-to-Verilog compiler," in *International Symposium on System on Chip (SoC)*, 2013.

[18] S. Franchini, A. Gentile, M. Grimaudo, C. A. Hung, S. Impastato, F. Sorbello, G. Vassallo, and S. Vitabile, "A sliced coprocessor for native Clifford algebra operations," in *10th Euromicro Conference on Digital System Design Architectures, Methods and Tools (DSD 2007)*, pp. 436–439, 2007.

[19] S. Franchini, A. Gentile, F. Sorbello, G. Vassallo, and S. Vitabile, "Embedded coprocessors for native execution of geometric algebra operations," *Advances in Applied Clifford Algebras*, Vol. 27, No. 1, pp. 559–580, 2017.

[20] D. Hestenes, "Clifford algebra to geometric calculus. A unified language for mathematics and physics," *American Journal of Physics*, Vol. 53, No. 5, p. 510, 1985.

[21] S. Roy, A. Mitra, and S. K. Setua, "Color & grayscale image representation using multivector," in *Proceedings of the 2015 Third International Conference on Computer, Communication, Control and Information Technology (C3IT)*, pp. 1–6, 2015.

[22] L. Dorst, D. Fontijne, and S. Mann, "GAViewer," 2005. [Online]. Available: http://www.geometricalgebra.net/gaviewer_download.html. [Accessed: 05-Mar-2018].

[23] L. Dorst, D. Fontijne, and S. Mann, *Geometric algebra for computer science*. Morgan Kaufmann, Burlington, 2007.

A. Sadr received the Ph.D. degree in Instrumentation from Department of instrumentation & Analytical Science (DIAS), University of Manchester Institute of Science and Technology (UMIST), Manchester, England in 2002. He is currently an Associate Professor in Electronic Engineering Department of Iran University of Science and Technology (IUST). His research interest include non-destructive evaluation, medical instrumentation, and microprocessor & microcontroller- based systems design.

N. Orouji received her B.Sc. degree in Electronic Engineering from K.N. Toosi University of Technology, Tehran, Iran in 2014, and M.Sc. degree in Digital Electronic Systems from Iran University of Science and Technology (IUST), Tehran, Iran in 2017. She is currently pursuing the Ph.D. degree in Electronic Engineering in Iran University of Science and Technology. Her research interests are specialized architecture design and novel co-processors.

© 2019 by the authors. Licensee IUST, Tehran, Iran. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution-NonCommercial 4.0 International (CC BY-NC 4.0) license (https://creativecommons.org/licenses/by-nc/4.0/).