Introduction to (Conformal) Geometric Algebra and its efficient implementation using Gaalop

Dietmar Hildenbrand
TU Darmstadt, Germany
dhilden@gris.informatik.tu-darmstadt.de

Alyn Rockwood
AMCS, Kaust
alyn.rockwood@kaust.edu.sa

ABSTRACT

. In this tutorial, we first give an introduction to (Conformal) Geometric Algebra. Gaalop combines the geometric intuitiveness of Geometric Algebra with a high performance of the implementation. We introduce the principles of Gaalop and CLUScript as its input language and show how to use them based on some simple examples.

Keywords. Conformal Geometric Algebra, Runtime Performance

1 CONFORMAL GEOMETRIC ALGEBRA

Conformal Geometric Algebra (CGA) as one specific geometric algebra which is very well suited for engineering applications is a new way of expressing most geometry focused mathematical problems. It deals naturally with intersections and transformations of planes, lines, spheres, circles, points and point pairs, but is also good at representing mechanics and dynamics. In Linear Algebra one would have to differentiate a plane-sphere intersection into three distinct cases, namely circle intersection, point intersection and no intersection. In Conformal Geometric Algebra the intersection itself is formulated as one operation on the plane (P) and the sphere (S) respectively.

$$R = S \wedge P$$

The three different cases of Linear Algebra are implicitly contained in the one result (R) of Conformal Geometric Algebra, being more compact and better readable. Similar observations can be made in other applications of geometry related mathematics. Applied to computer programs, CGA therefore has a high potential for improving code readability and to shorten production cycles. It has also been proven, that if implemented right, Geometric Algebra has at least similar performance, but sometimes even better performance, than conventional approaches [1].

An element of Conformal Geometric Algebra is referred to as multivector. A multivector consists of a linear combination of so called blades. Blades define the basis of CGA and are combinations of the vectors e_1, e_2, e_3, e_0 and e_∞. All possible blades are listed in table 1.

index	blade	grade
1	1	0
2	e_1	1
3	e_2	1
4	e_3	1
5	e_∞	1
6	e_0	1
7	$e_1 \wedge e_2$	2
8	$e_1 \wedge e_3$	2
9	$e_1 \wedge e_\infty$	2
10	$e_1 \wedge e_0$	2
11	$e_2 \wedge e_3$	2
12	$e_2 \wedge e_\infty$	2
13	$e_2 \wedge e_0$	2
14	$e_3 \wedge e_\infty$	2
15	$e_3 \wedge e_0$	2
16	$e_\infty \wedge e_0$	2

index	blade	grade
17	$e_1 \wedge e_2 \wedge e_3$	3
18	$e_1 \wedge e_2 \wedge e_\infty$	3
19	$e_1 \wedge e_2 \wedge e_0$	3
20	$e_1 \wedge e_3 \wedge e_\infty$	3
21	$e_1 \wedge e_3 \wedge e_0$	3
22	$e_1 \wedge e_\infty \wedge e_0$	3
23	$e_2 \wedge e_3 \wedge e_\infty$	3
24	$e_2 \wedge e_3 \wedge e_0$	3
25	$e_2 \wedge e_\infty \wedge e_0$	3
26	$e_3 \wedge e_\infty \wedge e_0$	3
27	$e_1 \wedge e_2 \wedge e_3 \wedge e_\infty$	4
28	$e_1 \wedge e_2 \wedge e_3 \wedge e_0$	4
29	$e_1 \wedge e_2 \wedge e_\infty \wedge e_0$	4
30	$e_1 \wedge e_3 \wedge e_\infty \wedge e_0$	4
31	$e_2 \wedge e_3 \wedge e_\infty \wedge e_0$	4
32	$e_1 \wedge e_2 \wedge e_3 \wedge e_\infty \wedge e_0$	5

Table 1: The 32 blades that compose a multivector of the 5D Conformal Geometric Algebra

2 GAALOP

The Geometric Algebra Algorithms Optimizer (Gaalop) [2] was developed by TU Darmstadt (Germany) and is a powerful tool for optimizing algorithms, expressed in Conformal Geometric Algebra. It generates non CGA-specific code from code defined in a CGA-specific language and symbolically optimizes the algorithm on-the-fly, invoking a Computer Algebra System (CAS). In this context, CGA can be seen as a higher level mathematical language that is being transformed into simple arithmetic mathematical language by Gaalop. Philosophically spoken, Gaalop could be defined as a math compiler.

Gaalop uses CLUScript as an input language and for instance C++ as output language. Conformal Geometric Algebra can not be expressed in terms of regular mathematical syntax. CGA-specific operators like the outer product $\wedge$, inner product . and geometric product $*$ require the design of a language like especially designed integrated development environments for CLUScript. The especially designed integrated development environments for CLUScript are called CLUCalc (old) and CLUViz (new). In words of the author Dr. Christian Perwass [4, 3].

> CLUCalc/CLUViz is a freely (for non-commercial use) available software tool for 3D visualizations and scientific calculations that was conceived and written by Dr. Christian Perwass. CLUCalc interprets a script

language called CLUScript, which has been designed to make mathematical calculations and visualisations very intuitive.

Indeed, CLUScript is a very intuitive language and we have found CLUCalc to be an advanced tool for developing and testing Geometric Algebra algorithms. It is easy to use, installs and runs smoothly on Microsoft Windows platforms.

Gaalop is also able to generate Field Programmable Array (FPGA), LaTex and CLUScript code. It is currently restricted to the very powerful 5D Conformal Geometric Algebra, but will be extended in the future also to higher algebras.

3 EXAMPLE

As an example the following CLUScript code computes the intersection circle C of two spheres $S1$ and $S2$ with center points $P1$ and $P2$ and radii $r1$ and $r2$.

```
P1 = x1*e1 +x2*e2 +x3*e3
+1/2*(x1*x1+x2*x2+x3*x3)*einf +e_0;

P2 = y1*e1 +y2*e2 +y3*e3
+1/2*(y1*y1+y2*y2+y3*y3)*einf +e_0;

S1 = P1 - 1/2 * r1*r1 * einf;
S2 = P2 - 1/2 * r2*r2 * einf;

?C = S1 ^ S2;
```

A question mark in CLUScript at the beginning of a line indicates the variables that have to be evaluateted and optimized.

The resulting C code generated by Gaalop for the intersection circle C is as follows and only depends on the variables $x1$, $x2$, $x3$, $y1$, $y2$, $y3$, $r1$ and $r2$:

```
C[7] = x1*y2-x2*y1;
C[8] = x1*y3-x3*y1;

C[9] = -0.5*y1*x1*x1-0.5*y1*x2*x2-0.5*y1*x3*x3+0.5*y1*r1*r1
       +0.5*x1*y1*y1+0.5*x1*y2*y2+0.5*x1*y3*y3-0.5*x1*r2*r2;

C[10] = -y1+x1;
C[11] = -x3*y2+x2*y3;

C[12] = -0.5*y2*x1*x1-0.5*y2*x2*x2-0.5*y2*x3*x3+0.5*y2*r1*r1
        +0.5*x2*y1*y1+0.5*x2*y2*y2+0.5*x2*y3*y3-0.5*x2*r2*r2;

C[13] = -y2+x2;

C[14] =-0.5*y3*x1*x1-0.5*y3*x2*x2-0.5*y3*x3*x3+0.5*y3*r1*r1
        +0.5*x3*y1*y1+0.5*x3*y2*y2+0.5*x3*y3*y3-0.5*x3*r2*r2;

C[15] = -y3+x3; C[16] = -0.5*y3*y3+0.5*x3*x3+0.5*x2*x2+0.5*r2*r2
        -0.5*y1*y1-0.5*y2*y2+0.5*x1*x1-0.5*r1*r1;
```

Gaalop always computes optimized 32-dimensional multivectors. Since a circle is described with the help of a bivector, only the blades 7 to 16 (see table 1) are used. As you can see, all the corresponding coefficients of this multivector are very simple expressions with basic arithmetic operations.

4 AGENDA

In this tutorial, we first of all give an introduction to (Conformal) Geometric Algebra and present some applications. In a second part, we introduce the principles of Gaalop and CLUScript as its input language and show how to use it based on some simple examples.

REFERENCES

[1] Dietmar Hildenbrand, Daniel Fontijne, Yusheng Wang, Marc Alexa, and Leo Dorst. Competitive runtime performance for inverse kinematics algorithms using conformal geometric algebra. In *Eurographics conference Vienna*, 2006.

[2] Dietmar Hildenbrand, Joachim Pitt, and Andreas Koch. *Gaalop - High Performance Parallel Computing based on Conformal Geometric Algebra*, volume 1, pages 350–358. Springer, May 2010.

[3] Christian Perwass. *Geometric Algebra with Applications in Engineering*. Springer, 2009.

[4] Christian Perwass. The CLU home page. Available at http://www.clucalc.info, 2010.