Foundations of Geometric Algebra Computing

Dietmar Hildenbrand

Technische Universitaet Darmstadt

Abstract. Geometric Algebra has the power to lead easily from the geometric intuition of solving an engineering application to its efficient implementation on current and future computing platforms. It is easy to develop new algorithms in areas such as computer graphics, robotics, computer animation and computer simulation. Owing to its geometric intuitiveness, compactness and simplicity, algorithms based on Geometric Algebra can lead to enhanced quality, a reduction in development time and solutions that are more easily understandable and maintainable. Often, a clear structure and greater elegance result in lower runtime performance. However, based on our computing technology, Geometric Algebra implementations can even be faster and more robust than conventional ones.

We present an example on how easy it is to describe algorithms in Geometric Algebra and introduce our technology for the integration of Geometric Algebra into standard programming languages. We really do hope that this technology can support the widespread use of Geometric Algebra Computing technology in many engineering fields [1].

Keywords: Geometric Algebra, Precompiler, C++, OpenCL

INTRODUCTION

Hermann G. Grassmann's *Ausdehnungslehre* of 1862 laid the foundations for Geometric Algebra as a mathematical language combining geometry and algebra. 150 years later, the Geometric Algebra Computing technology is intended to lay the foundations for the widespread use of this mathematical system on various computing platforms.

Seventeen years after Grassmann's first more philosophically written *Ausdehnungslehre* of 1844, he admitted in the preface of his mathematical version of 1862, "I remain completely confident that the labor I have expended on the science presented here and which has demanded a significant part of my life, as well as the most strenuous application of my powers, will not be lost. It is true that I am aware that the form which I have given the science is imperfect and must be imperfect. But I know and feel obliged to state (though I run the risk of seeming arrogant) that even if this work should again remain unused for another seventeen years or even longer, without entering into the actual development of science, still that time will come when it will be brought forth from the dust of oblivion and when ideas now dormant will bring forth fruit." And he went on to say, "there will come a time when these ideas, perhaps in a new form, will arise anew and will enter into a living communication with contemporary developments." The form that we give to Geometric Algebra with our computing technology, described in this paper, will hopefully lead to the widespread use of Geometric Algebra in many engineering fields.

FIGURE 1. Horizon of an observer on a beach.

[1] Note that we capitalize the term "Geometric Algebra Computing" to emphasize that it is being used with a specific meaning for a technology to integrate Geometric Algebra as a domain-specific language into standard programming languages. Similarly, we have capitalized "Geometric Algebra" and "Conformal Geometric Algebra".

Without going into too much details, the following example shows how easy it is to compute with geometric objects and how compact algorithms in Geometric Algebra can be formulated. It uses Conformal Geometric Algebra [1], an algebra embedding the 3D Euclidean space in a 5D algebra with two additional basis vectors(e_0 represents the origin and e_∞ represents infinity). Given a sphere S describing the Earth and a viewpoint P of an observer on a beach (see Fig. 1), we wish to find an algebraic expression for the horizon as seen by the observer, provided there is no occlusion of any sort other than the Earth itself in the scene.

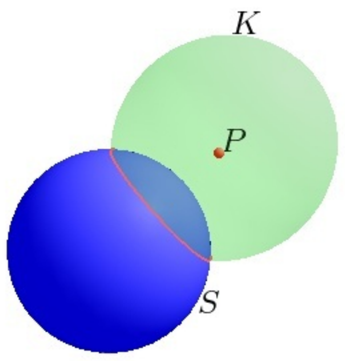


FIGURE 2. Calculation of the intersection circle (horizon)

First of all, we compute the distance from the viewpoint to the horizon. The square of this distance d can easily be computed based on the inner product of the Earth and the viewpoint:

$$d^2 = -2(S \cdot P). \tag{1}$$

Given this squared distance, we may construct another sphere K around P with that distance as its radius ($r^2 = d^2$):

$$K = P - \frac{1}{2} d^2 e_\infty \tag{2}$$

or

$$K = P + (S \cdot P) e_\infty. \tag{3}$$

The circle representing the horizon may then be calculated by calculating the intersection (via the outer product) of the two spheres. Fig. 2 illustrates the calculation:

$$C = S \wedge K \tag{4}$$

or

$$C = S \wedge (P + (S \cdot P) e_\infty). \tag{5}$$

This is a very compact description of the horizon circle C seen from the viewpoint P on the Earth S.

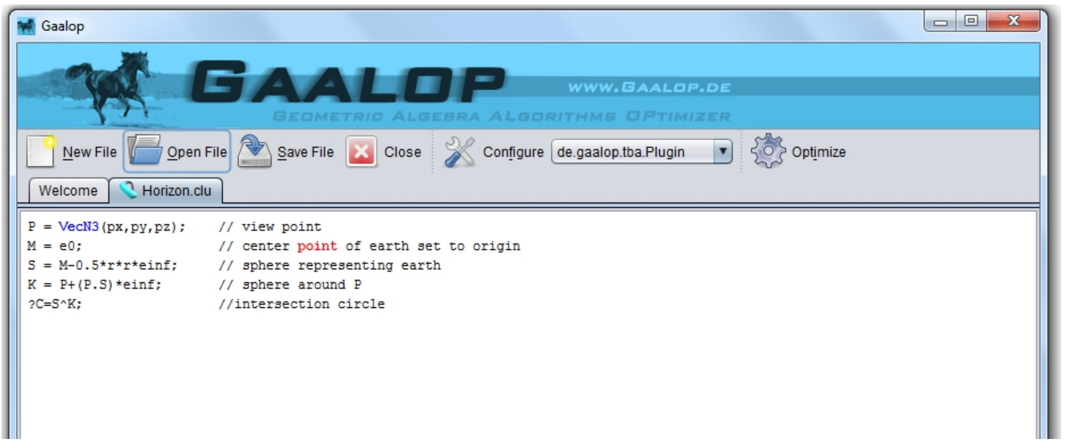
THE GEOMETRIC ALGEBRA ALGORITHMS OPTIMIZER GAALOP

The paper [2] was the basis for the development of our Gaalop (Geometric algebra algorithms optimizer) compiler, using the CLUCalc language [3] as the input language (see [4]). Gaalop is our main tool for the efficient implementation of Geometric Algebra algorithms.

Listing 1: Gaalop input for the horizon example.

```
P = VecN3(px,py,pz);       // view point
M = e0;                    // center point of earth set to origin
S = M-0.5*r*r*einf;        // sphere representing Earth(center M, radius r)
K = P+(P.S)*einf;          // sphere around P
?C=S^K;                    // intersection circle
```

The CLUCalc code for the computation of the horizon example is shown in Listing 1. Note that we did not need any coordinates for the formulation of the horizon example. Only if we have to make concrete computations do we have to use coordinates. The CLUCalc macro VecN3 computes the 5D representations of the 3D point (px,py,pz). The variables **px**, **py**, **pz** and the radius **r** are free variables. This script can be used directly as an input to Gaalop. The free variables are handled in Gaalop as symbolic variables. Fig. 3 shows a screenshot of the Gaalop input for the



```
P = VecN3(px,py,pz);     // view point
M = e0;                  // center point of earth set to origin
S = M-0.5*r*r*einf;      // sphere representing earth
K = P+(P.S)*einf;        // sphere around P
?C=S^K;                  //intersection circle
```

FIGURE 3. The CLUCalc input code of the horizon example as requested by Gaalop

horizon example. First of all, we can see that the CLUCalc code in Listing 1 is exactly what Gaalop expects as input. A question mark at the beginning of a line indicates a multivector variable that has to be explicitly computed by Gaalop. This means that Gaalop is able to optimize not only single statements, but a number of Geometric Algebra statements written in the CLUCalc language. In Listing 1, the expressions for P, M, S, and K are used only by Gaalop, in order to compute an optimized result for the horizon circle C (see the question mark in the last line of the listing).

A multivector is the 2^n-dimensional algebraic element of a n-dimensional Geometric Algebra (32-dimensional for Conformal Geometric Algebra). Gaalop writes the content of the multivectors in C++ syntax in an output file. More exactly, it computes the coefficients of the multivector and assigns them to an array representing this multivector. Listing 2 shows the result for the optimized circle C as an array with entries 8, 9, 11, 12, 13, 14, and 15 (all the other entries are zero).

Listing 2: Gaalop output for the horizon example.

```
C[8] = 0.5f * px * r * r;   // e1^einf
C[9] = - px;                // e1^e0
C[11] = 0.5f * py * r * r;  // e2^einf
C[12] = - py;               // e2^e0
C[13] = 0.5f * pz * r * r;  // e3^einf
C[14] = - pz;               // e3^e0
C[15] = - r * r;            // einf^e0
```

This shows that only very simple arithmetic expressions are needed. This is why a high runtime performance of implementations with Gaalop can be achieved.

PRECOMPILER FOR GEOMETRC ALGEBRA

In order to simplify the use of the Geometric Algebra Computing technology, we have developed Gaalop GPC [5], a precompiler, which integrates Gaalop [6] into standard programming languages such as C++, OpenCL, and CUDA. Figure 4 outlines the concept for the C++ programming language. With Gaalop GPC, we are able to enhance ordinary C++ code with Geometric Algebra code and automatically generate optimized C++ code.

A precompiler is an elegant way of extending the features of a programming language. For Geometric Algebra Computing, it is of high interest to use both the power of high-performance languages such as C++/OpenCL/CUDA and the elegance of expression of a domain-specific language such as CLUCalc. We have therefore embedded

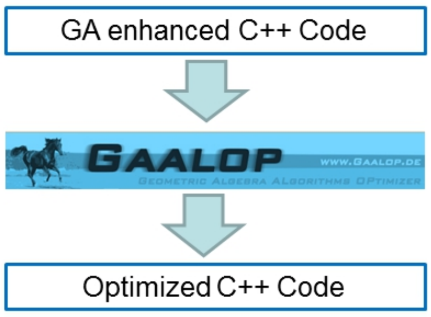


FIGURE 4. Gaalop GPC for C++.

CLUCalc code into C++/OpenCL and CUDA code, and compile it by utilizing the precompiler concept and the fast optimizations and code generation features of Gaalop.

Gaalop GPC enhances standard programs with:

- embedding of Geometric Algebra code using multivectors;
- functionality to interact with multivectors.

The horizon example is implemented in C++ using Gaalop GPC as follows. Listing 3 computes and visualizes the observer point, the spheres *S* and *K*, and the horizon circle.

Listing 3: Code for horizon example in Gaalop GPC for C++ with visualization.

```
void horizon() {
  #pragma clucalc begin
     :Black;
     :P = VecN3(1,1,0);
      r = 1;
     :Blue;
     :S = e0-0.5*r*r*einf;
     :Color(0,1,0,0.2);
     :K = P+(P.S)*einf;
     :Red;
     :C = S^K;
  #pragma clucalc end
}
```

This shows how easy it is to integrate Geometric Algebra into standard programming languages. We hope that this Geometric Algebra Computing technology will lead to a widespread use in many engineering areas.

REFERENCES

1. D. Hestenes, “Old Wine in New Bottles: A New Algebraic Framework for Computational Geometry,” in *Geometric Algebra with Applications in Science and Engineering*, edited by E. Bayro-Corrochano, and G. Sobczyk, Birkhäuser, 2001, ISBN 0-8176-4199-8.
2. D. Hildenbrand, D. Fontijne, Y. Wang, M. Alexa, and L. Dorst, “Competitive runtime performance for inverse kinematics algorithms using conformal geometric algebra,” in *Eurographics Conference Vienna*, 2006.
3. C. Perwass, The CLU home page, Available at http://www.clucalc.info (2010).
4. D. Hildenbrand, J. Pitt, and A. Koch, “Gaalop – High Performance Parallel Computing based on Conformal Geometric Algebra,” in *Geometric Algebra Computing in Engineering and Computer Science*, edited by E. Bayro-Corrochano, and G. Scheuermann, Springer, May 2010.
5. P. Charrier, and D. Hildenbrand, “Geometric Algebra Enhanced Precompiler for C++ and OpenCL,” in *AGACSE conference La Rochelle*, 2012.
6. D. Hildenbrand, P. Charrier, C. Steinmetz, and J. Pitt, Gaalop home page, Available at `http://www.gaalop.de` (2012).