Visualization of the projective line geometry for geometric algebra

Patrick de Kok
Supervisor: Leo Dorst

July 3, 2012

Abstract

This thesis describes a method for computationally recognizing the geometrical interpretation of the elements of the model of projective line geometry in geometric algebra. These different interpretations are not merely recognizable by the grade of the elements, or the absense or presence of some basis vector, as can be done for conformal geometric algebra.

GAViewer, a graphical calculator for the Euclidean and conformal models of geometric algebra, is extended to implement the projective line geometry and this visualization.

Keywords: Geometric algebra, projective geometry, Plücker coordinates, conformal geometry.

Contents

1 Introduction

Needs to be rewritten. Will be done in the end?

Just as linear algebra, geometric algebra is an algebra over a given vector space $\mathbb{R}^n$. The difference lies in its operations; whereas linear algebra relies on matrix manipulations which cannot always be inverted, geometric algebra's base operation is the geometric product. From this invertible product, one can deduce the inner product, known from linear algebra, and the outer product. The outer product grants the user access to the exterior algebra $\bigwedge \mathbb{R}^n$. The outer product of $\mathbf{A}, \mathbf{B} \in \bigwedge \mathbb{R}^n : \mathbf{A} \wedge \mathbf{B}$ represents the set of elements that are linear combinations of its operands $\{\alpha\mathbf{A} + \beta\mathbf{B} \mid \alpha, \beta \in \mathbb{R}\}$

There are already several models of geometry in use in the geometric algebra community, of which the conformal model is the most popular and widely known [3]. Although usable in many cases, the transformations of the conformal model are a subset of those of the projective model. The projective transformations are an important class of operations within computer vision and computer graphics.

GAViewer is a visualization and computing tool for the 3-dimensional Euclidean and conformal models of geometric algebra, developed by Daniel Fontijne at the University of Amsterdam [4]. It allows the user to perform calculations in geometric algebra, and shows the results, both numerical as well as graphical. Moreover, it allows the user to rotate, translate and zoom the viewport, as well as to manipulate the displayed elements. The variables in which they are stored are automatically updated, and, if desired by the user, other objects that are parameterized by the manipulated object, may be updated dynamically as well.

Recently, Li and Zhang [5] have found a way to model projective geometry, using a 6 dimensional representation space $\mathbb{R}^{3,3}$ with a special metric, which allows three basis vectors to square to -1. Using Plücker coordinates [5,6], lines in 3-dimensional Euclidean space $\mathbb{E}^3$ are represented in the representation space by vectors $\mathbf{v}^2 = \mathbf{v}\mathbf{v} = \mathbf{v}\cdot\mathbf{v} = 0$. As a consequence, there are also vectors in the representation space $\mathbf{w}^2 \neq 0$ which do not represent lines in $\mathbb{E}^3$. Different geometrical objects are generated by the outer product over lines and non-lines. For example, for two intersecting lines ℓ_1, ℓ_2, the outer product $\ell_1 \wedge \ell_2$ represents a pencil, the collection of all lines that are in the same plane as ℓ_1 and ℓ_2, and pass through the same point. Taking the outer product of this pencil with a third intersecting line that does not lie in the same plane, results in a bundle; the set of all lines intersecting in a certain point. This could be used to represent points in our space of lines.

In their article, Li and Zhang have not discussed what each element in their algebra might represent. Barrau [1,2] and Pottmann and Wallner [6,7] have investigated the objects that can be represented in a 6-dimensional space with Plücker coordinates, both using a different algebra from ours. Pottmann and Wallner use linear algebra, while Barrau's algebra has less expressive power.

1.1 Research question

1.2 Document structure

In section 2 of this thesis, the basics of our mathematical toolkit are explained. This includes a superficial introduction to geometric algebra and its homogeneous model, as well as a description of the Plücker space and how it is represented in geometric algebra.

Section 3 describes the problem of recognizing the geometrical interpretation of elements of the Plücker space and parameterizing important features for displaying the element on a screen. In subsection 3.2, methods for achieving this classification and parameterization are given. The implementation of these methods in GAViewer is documented in section 4.

This documents concludes at section 5 with a discussion of this work as well as possible future work, and a bibliography.

2 Projective geometry and geometric algebra

2.1 Geometric algebra

Paraphrase Moos's work?

Include the next notation explanation:

- Bold for Euclidean
- Capital for multivectors
- Lowercase for vectors/1-blades
- Greek for scalars/0-blades
- $\mathbb{R}^{3,3}$
- Geometric product: $A\,B \leftarrow$ small space
- Geometric division $A\,/\,B$ or too trivial?
- Geometric division $A\,/\,B$ or too trivial?
- Outer product $\wedge$
- Left contraction $\rfloor$
- Dot product $\cdot$
- Dualization A^*, undualization A^{-*}
- Euclidean (un-)dualization $\mathbf{A} \rfloor \mathbf{I}_3^{-1}$, $\mathbf{A} \rfloor \mathbf{I}_3$

2.2 Plücker model

This part will be paraphrased from [3] and [5].

In the homogeneous model, the line through points $p = e_0 + \mathbf{p}$ and $q = e_0 + \mathbf{q}$ is represented as

$$L = p \wedge q = e_0 \wedge (\mathbf{q} - \mathbf{p}) + \mathbf{p} \wedge \mathbf{q}. \tag{1}$$

The direction and moment of a line are easily recognised in this expression. The direction $\mathbf{d} = \mathbf{p} - \mathbf{q}$ is encoded in the first factor as $e_0 \wedge -\mathbf{d} = \mathbf{d} \wedge e_0 = \mathbf{d}e_0$. The moment $\mathbf{m} = \mathbf{p} \times \mathbf{q} = (\mathbf{p} \wedge \mathbf{q}) \rfloor \mathbf{I}_3^{-1}$ can be found in the second term as $\mathbf{m} \rfloor \mathbf{I}_3 = ((\mathbf{p} \wedge \mathbf{q}) \rfloor \mathbf{I}_3^{-1}) \rfloor \mathbf{I}_3 = \mathbf{p} \wedge \mathbf{q}$, which results in another general formula for lines:

$$L = \mathbf{d}e_0 + \mathbf{m} \rfloor \mathbf{I}_3. \tag{2}$$

Within classical literature of linear algebra, the same object is written as $-\{\mathbf{d}\,;\mathbf{m}\}$, using *Plücker coordinates* to denote a line as a 6D vector.

3 Classifying and parameterizing elements of different geometric interpretations

3.1 The problem

Needs a better subsection title.

3.2 The solution

Needs a better subsection title.

4 Implementation

4.1 Design decision

Extending an existing program with more functionality can be done in two ways. 1: wrapping the original program in another layer of coding. 2: changing the code of the original program and recompile the whole thing.

	Wrap program		Edit source code
−	User will insert text in a separate box from the connected GAViewer, OR I need to build a whole new GUI.	+	Is nicely integrated in current program
+	Don't need to understand a complex code base	−	Code is not well-documented
+	Can be done in C, C++, Java (limited by GAIGEN's output). My Java programming skills are better than those in C/C++	−	Must be done in C++
−	Java is often (a bit) slower than C++	+	C++ is fast in array manipulation
+	GAIGEN 2.5 must be much faster than GAIGEN 1.0	?	Need to find out if APIs of different GAIGENs are the same
+	Can be wrapped over (and over and over...), or use other GAViewer implementation.	−	Is not out-of-the-box compatible with other extensions of GAViewer, but can be made by patches and quite some more work
−	Multiple wrappers make running it difficult/slow	+	Merging multiple edits of codebase makes implementation difficult, but running easy
?	How to cope with `dynamic`/dragged and interdepencency?	+	Dragging and `dynamic` is implemented
−	Code to implement user interaction will be quite complex	+	Is provided, doesn't need to be touched
+	Can run "modern" versions of libraries (FLTK 1.3, ANTLR 3.4, GAIGEN 2.5)	−	Is based on old libraries (FLTK 1.1, ANTLR ??? GAIGEN 1.0), implies less functionality/speed
−	Need to write a whole new grammar	?	Need to edit old, undocumented grammar, of *which* edition of ANTLR?
−	No possibility to add new graphical elements; everything must be composed from CGA elements	?	Should be possible, not sure how much effort that will cost (regulus, screws)

Decided to edit the source code.

5 Conclusion

References

[1] Johan Antony Barrau. *De ruimte*, volume 2 of *Analytische meetkunde*. Noordhoff, Groningen, 1918–1927.

[2] Johan Antony Barrau. *Het platte vlak*, volume 1 of *Analytische meetkunde*. Noordhoff, Groningen, 1918–1927.

[3] Leo Dorst, Daniel Fontijne, and Stephen Mann. *Geometric Algebra for Computer Science: An Object-Oriented Approach to Geometry (The Morgan Kaufmann Series in Computer Graphics)*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2007.

[4] Daniel Fontijne. *GAViewer Documentation: Version 0.84*. University of Amsterdam, March 2010.

[5] Hongbo Li and Lixian Zhang. Line geometry in terms of the null geometric algebra over $\mathbb{R}^{3,3}$, and application to the inverse singularity analysis of generalized stewart platforms. In Leo Dorst and Joan Lasenby, editors, *Guide to Geometric Algebra in Practice*, pages 253–272. Springer London, 2011.

[6] Helmut Pottmann and Johannes Wallner. *Computational Line Geometry*, chapter 2, pages 133–158. Springer, August 2001.

[7] Helmut Pottmann and Johannes Wallner. *Computational Line Geometry*, chapter 3, pages 159–194. Springer, August 2001.