Shedding new light on ray tracing for optical metrology systems by using Geometric Algebra

Simon Hartel, Christian Faber

Hochschule Landshut – University of Applied Sciences

mailto:simon.hartel@haw-landshut.de

Simulating camera-based optical metrology and imaging systems can be done well with geometric optics in the Euclidean space. Although this is a problem long-solved by using linear algebra and vector analysis, these "classic" tools exhibit several drawbacks. It is shown how the powerful combination of geometric optics and geometric algebra can be used for a more concise and realistic modelling.

1 Introduction

Camera-based optical metrology systems, like stereo vision, fringe projection, deflectometry and light sectioning, can be described well with geometric optics by tracing chief rays. To do this, geometric primitives (points, directions, lines, planes) alongside operations (intersections, conjunctions, projections) and transformations (rotations, translations, reflections) have to be modelled. Although this problem is long-solved by using linear algebra and vector analysis, cumbersome case differentiations for different types of geometric primitives as well as tedious, usually coordinate-dependent exception handling (e.g. to avoid gimbal lock) are required. Useful concepts such as Plücker coordinates, homogeneous coordinates, or (dual) quaternions have been introduced to overcome this, which appear "cobbled together" and can quickly become hard to maintain.

Recently the use of geometric algebras was suggested in computer science as a unifying approach. In this contribution it is shown how ray tracing for camera-based optical metrology and imaging systems can benefit from the powerful combination of geometric optics and geometric algebra.

2 Geometric Algebra (GA)

The idea behind *geometric algebra (GA)* [1], also known as *Clifford algebra*, is that both geometric primitives and transformations become elements of an algebra that can be used for calculations – just as in the transition from the vector space $\mathbb{R}^2$ to the complex numbers $\mathbb{C}$. To generate these algebras the so-called *geometric numbers* are introduced:

$$e_+{}^2 = 1\ , \qquad e_-{}^2 = -1\ , \qquad e_0{}^2 = 0\ . \tag{1}$$

How many of each type are used is denoted in the *signature* $\mathbb{R}_{p,q,r}$ of the algebra with the number p of e_+, q of e_- and r of e_0 elements [2,3].

Fundamental for GA is its *geometric product*. For two vectors $\boldsymbol{u}$ and $\boldsymbol{v}$ it can be written as

$$\boldsymbol{uv} = \boldsymbol{u} \cdot \boldsymbol{v} + \boldsymbol{u} \wedge \boldsymbol{v}\ . \tag{2}$$

It consists in part of an *inner product* ("·") and an *outer product* ("∧"). The result for two vectors is a scalar plus a so-called *bivector*, which can be imagined as an oriented plane element spanned by the two vectors it is constructed from. As with complex numbers, this sum cannot be simplified further. Instead, the result is called a *multivector*.

Another basic principle in GA is the so-called *sandwich product*

$$\boldsymbol{mX\tilde{m}}\ , \tag{3}$$

where the operator, in general a multivector $\boldsymbol{m}$, is applied from the front and (reversed) from the back to the multivector $\boldsymbol{X}$ with respect to the geometric product. This structure is used in GA for describing transformations and can be applied to *any* primitive.

3 Plane-Based Geometric Algebra (3D-PGA)

Throughout the past years, the use of the special geometric algebra with signature $\mathbb{R}_{3,0,1}$ was suggested in the field of computer science to describe Euclidean space. This algebra is called *plane-based* [2,3,4,5] (or similar: projective [6]) *geometric algebra (3D-PGA)*. As can be seen from the signature, the algebra is generated among others by a nilpotent geometric number e_0, which leads to a *degenerated metric*. This may look disconcerting at first glance but ultimately brings capable concepts for the representation of Euclidean space like *projective geometry*, *duality* as well as *unifying and merging of geometric primitives and transformations*.

For a more complete introduction and definition of both GA and 3D-PGA see the listed references.

4 Basics of ray tracing with 3D-PGA

Since 3D-PGA provides a unifying representation of the Euclidean space it is well suited for geometric optics, which deals mainly with rays, points and planes as well as corresponding operations and transformations [7]. As shown below, ray tracing for camera-based optical metrology and imaging systems can strongly benefit from using geometric

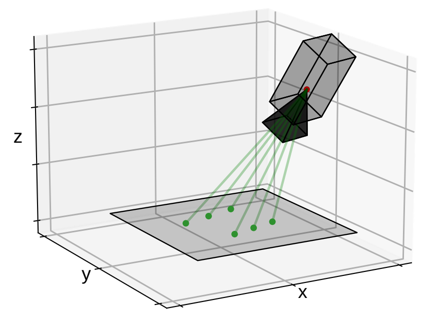
submitted: 29.07.2022 published: 05.09.2022

algebra combined with geometric optics and can lead to concise and clean program code.

Camera-based metrology systems are often simulated by their inverted light ray path, known as *sight rays*. Intersecting the (chief) sight rays $\boldsymbol{r}$ of a camera, which e.g. are known from the camera calibration, with a planar object $\boldsymbol{p}$ is simply done by multiplying using the outer product (Fig. 1):

$$\boldsymbol{P}_{intersect} = \boldsymbol{r} \wedge \boldsymbol{p} \; . \tag{4}$$

The result $\boldsymbol{P}_{intersect}$ are the corresponding intersection points. Note that due to the incorporated projective geometry in 3D-PGA, intersections of parallel rays and planes are automatically represented by *ideal* elements ("at infinity"). The outer product is not just a shorthand for an overly complicated function "in the background". Rather, it is a simple operation of the algebra itself, literally allowing to "calculate" with geometric primitives. By defining a multivector datatype and overloading operators, formulas like (4) can directly be written as program code (Fig. 1).

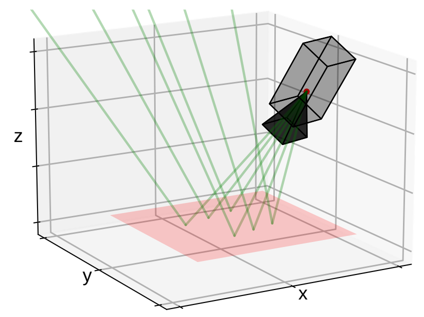


```
intersect = ray ^ plane
```

Fig. 1 *Intersection of rays with a planar object and corresponding program code.*

Other operations and transformations can be performed accordingly [2,3,4,5,6]: E.g., reflecting the sight rays on a specular plane by the sandwich product (3) (Fig. 2):

$$\boldsymbol{r}_{refl} = \boldsymbol{p}\boldsymbol{r}\widetilde{\boldsymbol{p}} \; . \tag{5}$$



```
ray_refl = plane * ray * ~plane
```

Fig. 2 *Reflection of rays on a planar object and corresponding program code ("*" = geometric product).*

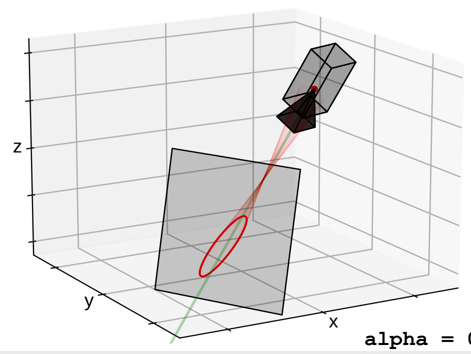
Rotations and translations, e.g. to model the extrinsic calibration of a camera, can also be described in a similar way. Given a rotation axis $\boldsymbol{a}$ (*Euclidean line)* and a translation direction $\boldsymbol{d}$ (*ideal line* at infinity), both are represented by bivectors in 3D-PGA, and the transformations can be uniformly written as:

$$\boldsymbol{t}_{rot} = e^{\boldsymbol{a}/2} \; , \qquad \boldsymbol{t}_{trans} = e^{\boldsymbol{d}/2} \; . \tag{6}$$

As with complex numbers or quaternions, the same exponential function is used. The transformations can simply be combined by the geometric product to a so-called *motor* $\boldsymbol{m} = \boldsymbol{t}_{trans}\boldsymbol{t}_{rot}$ and applied to *any* primitive by (3). In fact, this is isomorphic to (dual) quaternions, so *no gimbal lock* occurs [4,5].

5 Conclusion

It was shown how the powerful combination of geometric optics and geometric algebra can be used for ray tracing camera-based optical metrology and imaging systems, and how this approach allows for a more concise and realistic modelling. The use of GA allows to realize quite complex ray tracing tasks in a versatile way and with comparable few lines of program code (Fig. 3). Case differentiations or exception handling are ordinarily not required.



```
                                  alpha = 0…2π
      motor = exp(alpha/2 * ray)
 aperture = motor * marg_ray * ~motor
circle_of_confusion = aperture ^ plane
```

Fig. 3 *Constructing the circle of confusion of a defocused camera on a tilted plane using the chief ray as axis to create the aperture from a given marginal ray.*

References

[1] W. Clifford: "Applications of Grassmann's extensive algebra", American Journal of Mathematics,1(4): S. 350–358, 1878.

[2] C. Gunn, S. De Keninck: "Geometric Algebra for Computer Graphics", SIGGRAPH 2019.

[3] L. Dorst, S. De Keninck: "May the Forque Be with You", SIBGRAPI 2021.

[4] L. Dorst, D. Fontijne, S. Mann: "Geometric Algebra for Computer Science: An Object-oriented Approach to Geometry", Amsterdam: Elsevier, 2009.

[5] L. Dorst, S. De Keninck: "A Guided Tour to the Plane-Based Geometric Algebra PGA", version 2.0, 2022, *https://geometricalgebra.net* and *https://bivector.net*.

[6] E. Lengyel: "Projective Geometric Algebra Done Right", 2020, *https://projectivegeometricalgebra.org*.

[7] T. Corcovilos: "Geometric Algebra - beyond vectors", 2019, *https://www.corcoviloslab.com*.

submitted: 25.07.2022 published: 05.09.2022