Ray-Tracing Objects and Novel Surface Representations in CGA

Sushant Achawal[1][0000−0002−9847−329X], Joan Lasenby[1][0000−0002−0571−0218], Hugo Hadfield[1][0000−0003−4318−050X], and Anthony Lasenby[2][0000−0002−8208−6332]

[1] Cambridge University, Department of Engineering, Cambridge, UK, {ssa43,jl221,hh409}@cam.ac.uk
[2] Cambridge University, Department of Physics, Cambridge, UK, a.n.lasenby@mrao.cam.ac.uk

Abstract. Conformal Geometric Algebra (CGA) provides a unified representation of both geometric primitives and conformal transformations, and as such holds great promise in the field of computer graphics [1–3]. In this paper we implement a simple ray tracer in CGA with a Blinn-Phong lighting model and use it to examine ray intersections with surfaces generated from interpolating between objects [7]. An analytical method for finding the normal line to these interpolated surfaces is described. The expression is closely related to the concept of surface principal curvature from differential geometry and provides a novel way of describing the curvature of evolving surfaces.

Keywords: Conformal Geometric Algebra · Ray-Tracing · Direct Object Interpolation · Surface Curvature.

1 Motivation and Related Work

The development of Conformal Geometric Algebra (CGA) has shown how simple expressions can explain complex geometrical operations [1–3]. This suggests the use of CGA in Computer Graphics, and indeed ray-tracers using CGA have been implemented in the past [3–6]. Recent developments have explored direct-interpolation and its use in generating surfaces and splines [7]. In this paper we investigate some of the properties of these surfaces and their incorporation into an experimental ray tracer.

2 Conformal Geometric Algebra, CGA

The ray-tracer used in this paper is constructed using CGA and all algebraic expressions given will be in terms of elements of this algebra. CGA adds two more basis vectors, e and $\bar{e}$, to the original basis vectors of 3D Euclidean space, giving a complete basis for the 5D space with the following signature: $e_1^2 = e_2^2 = e_3^2 = e^2 = 1$ and $\bar{e}^2 = -1$. These extra basis vectors are used to define two null vectors: $n = e + \bar{e} \equiv n_\infty$ and $n_0 = \frac{\bar{n}}{2} = \frac{e-\bar{e}}{2}$. The mapping from a 3D vector, x, to its corresponding CGA vector, X, is given by:

$$X = F(x) = \frac{1}{2}\left(x^2 n + 2x - \bar{n}\right)\frac{1}{2} \equiv \frac{1}{2}x^2 n + x - n_0 \tag{1}$$

All vectors formed from such a mapping are null. More background on CGA can be found in [1–3].

3 Camera Model and Ray Casting

A pinhole camera model is used with the geometry shown in Figure 1. It is defined by a rotor R_{MV} incorporating rotation and translation that takes the camera from the origin to its pose in space, a focal length f and two bounds $x_{\max}$ and $y_{\max}$ on the size of the image plane.

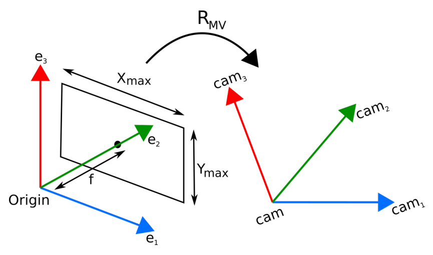

Fig. 1. The camera is defined by a focal length, a transformation from the origin, and bounds on the image plane.

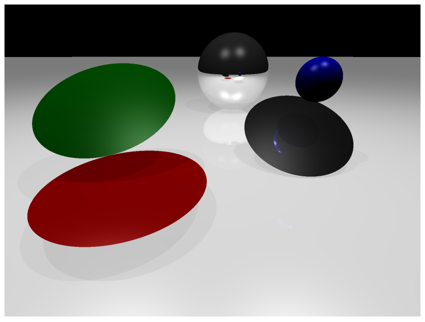

Fig. 2. An image from the ray-tracer containing examples of disks, spheres and planes.

We take $(i, j) = (0, 0)$ to be at the bottom left hand corner of the image. For an image of width w and height h the world coordinates of the point P_{ij} at the centre of pixel (i, j) are given by:

$$P_{ij} = R_{MV}\left[F(fe_2 - \frac{x_{\max}}{2}(1-(2i/w))e_1 - \frac{y_{\max}}{2}(1-(2j/h))e_3)\right]\tilde{R}_{MV} \tag{2}$$

We then generate the ray from the camera centre, L_{ij}, that passes through P_{ij}, via the expression

$$L_{ij} = cam \wedge P_{ij} \wedge n_\infty$$

4 Ray Tracing Evolved Circles

The intersection and reflection of lines with the other blades of CGA (planes, disks, spheres) has already been investigated in [2, 3, 5] and we will not repeat

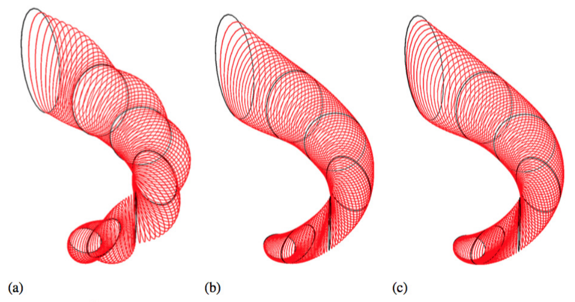

Fig. 3. Interpolation through circular control objects. (a) linear, (b) quadratic, (c) cubic

those results here. Instead we will turn directly to an interesting class of surface that arises from the direct interpolation of CGA circles [7], examples of which are shown in Figure 3. In order to generate such a surface, a direct interpolation is first performed between two boundary circles, C_1 and C_2:

$$X_\alpha = \alpha C_1 + (1-\alpha)C_2 \tag{3}$$

where we take α moving between 0 and 1, which moves us from C_2 to C_1. The result of this interpolation is not itself a valid circle and needs to be 'projected' onto a blade via multiplication by a *projector*. This projector has *only* scalar and 4-vector parts and its construction is detailed in [7] and outlined in the following. First form

$$K_\alpha = \sqrt{-X_\alpha \tilde{X}_\alpha} \tag{4}$$

where the square root is as defined in [8]. We then form K^*_α by reversing the sign of the 4-vector part, ($K^*_\alpha = \langle K_\alpha \rangle - \langle K_\alpha \rangle_4$), and use this to produce the following expression for the interpolated circle:

$$C_\alpha = \frac{K^*_\alpha}{K^*_\alpha K_\alpha} X_\alpha \qquad \alpha \in [0,1] \tag{5}$$

Given that these surfaces may find genuine applications in computer graphics and CAD, it is desirable to explore their properties with respect to the ray tracing framework. Specifically, for a given ray and scene object, the geometric constructions of interest for lighting models are the point of intersection between a ray and a surface, and the surface normal at that specific intersection point.

4.1 Intersection Point of Ray and Interpolated Surface

In the intersection of a ray with a circle, the meet produces the 1-vector Σ. If $\Sigma = 0$ there are two intersections and if $\Sigma^2 = 0$ there is one intersection [2] . Therefore, to find the intersection point between our interpolated surface and a ray L, we need to find a value of α for which:

$$(C_\alpha \vee L)^2 = \langle C_\alpha L \rangle_4^2 = 0$$

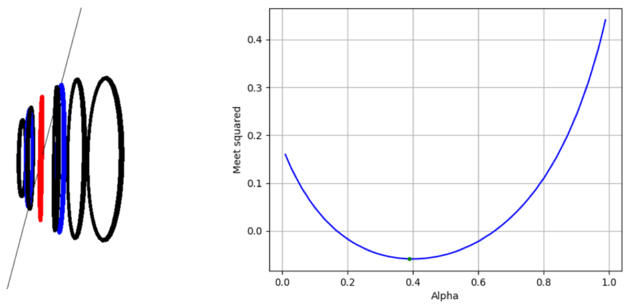

Fig. 4. Left: An image showing an example interpolated surface and a ray passing through it, the circles in blue show the circles which have a meet squared of 0 with the incident ray, the red circle shows where the meet squared is minimised. Right: A plot showing the value of the meet squared as a function of alpha for this case.

Figure 4 provides a simple visual illustration of one example of the shape of this curve as a function of α. While this example shown in the figure is particularly smooth, experiments indicate that in the general case this function is not well approximated by low order polynomials and the roots of this function have to be found by general non-linear root finding techniques such as a bisection search method. These general root finding methods are computationally expensive and an analytical solution would be desirable.

4.2 Analytic Form for Normal

Given the α for which the ray intersects the surface, we have both the interpolated circle, C_α, and the point of intersection X. We now use the result in [2] to extract a tangental line L_C in the plane of the circle at X:

$$L_C = (X \cdot C_\alpha) \wedge n_\infty \tag{6}$$

We would now like an analytic form for the tangent to the surface corresponding to evolving the surface through an increment of α, postulating that this will be orthogonal to L_C and that these two tangent vectors will then be the directions of principal curvature. Clearly $\frac{dC(\alpha)}{d\alpha} \equiv \dot{C}_\alpha$ will be a key quantity in deriving this additional tangent vector. A first observation is that the circle and its derivative will be orthogonal to one another, i.e. $\dot{C} \cdot C = 0$, and that the geometric product is minus itself under reversion, i.e. $\dot{C}\,C = -C\,\dot{C}$ (note that here, and in what follows, we will drop the α subscript on C). This follows from the fact that $C^2 = C \cdot C = 1$ (our circles are all normalised), so that:

$$\frac{d}{d\alpha}(C \cdot C) = C \cdot \dot{C} + \dot{C} \cdot C = 0 \;\; , \;\; \frac{d}{d\alpha}(C\,C) = C\,\dot{C} + \dot{C}\,C = 0 \tag{7}$$

Since $C \cdot \dot{C} = -C \cdot \dot{C}$ and they are both scalars, this tells us $\dot{C} \cdot C = 0$. Since $C\,\dot{C} = -C\,\dot{C} = -\widetilde{(C\,\dot{C})}$, this further indicates that the product can only have bivector parts (this is a standard construct in many areas, the most obvious being rigid body dynamics [10]). Let us call this bivector, Ω_C:

$$\Omega_C = C\dot{C} \tag{8}$$

Using the analogy with rigid body dynamics, we think of this bivector as the *angular velocity bivector* of the circles as they evolve under the parameter α. We note here that a similar construction would be possible for all other main objects that we use in CGA, since they are all normalised to 1 or 0. The null vectors representing points, X, have a constant 'length' due to normalisation, so their 'velocity' can simply be found via the inner product with the angular velocity bivector given in equation 8.

$$\dot{X} = X \cdot \Omega_C = X \cdot (C\dot{C}) \tag{9}$$

It can be shown that this in fact provides the tangential direction required, so the line can be formed by:

$$L_T = \dot{X} \wedge X \wedge n \tag{10}$$

The fact that lines L_C and L_T are perpendicular can be verified by showing that the quantity $L_T L_C$ has only a bivector part (see [2] for a discussion of when intersecting lines are orthogonal). Given these two tangent lines L_C and L_T, we can construct the plane tangent to the surface at X by computing the join of the two lines. Or, we can bypass the plane entirely and compute the surface normal line directly as:

$$N = \langle L_T L_C I_5 \rangle_3 \tag{11}$$

5 Blinn-Phong Lighting Model

The ray-tracer employs the Blinn-Phong lighting model, this simple model is well studied in the graphics literature, for more in depth information see [9]. Under this model the intensity value for each pixel is given by the following expression:

$$I_\lambda = c_\lambda k_a + k_r I_{r\lambda} + \sum_i S_i f_{\text{att}} I_{pi\lambda} \left(c_\lambda k_d \left(L_i \cdot N \right) + k_s \left(N \cdot H \right)^q \right) \quad \forall \lambda \in \{R, G, B\} \tag{12}$$

Note that traditionally the terms N, L_i and H are standard 3-vectors. Since the language of our ray-tracer is CGA, these terms in Equation 12 actually represent the normalised lines which have the same directions as the 3-vectors which pass through the ray-object intersection point. As all these lines are normalised such that $L^2 = 1$ the inner products between them will simply give the cosine of the angles between them as in the traditional 3d vector case [10].

The *normal line* is required across multiple terms and is crucial to the lighting model. This is given in CGA as $N = L' - L$ where L' is the reflected ray and L is the incident ray. Note that L' is necessarily calculated for tracing reflected rays. The line L_i specifying the direction to the i^{th} light source is given by forming the wedge product between the point of intersection of the incident ray and the object, the point position of the i^{th} light source and n_∞. The half way line H can be found by $H = L_i - V$, where V is the line from the intersection point to the viewer.

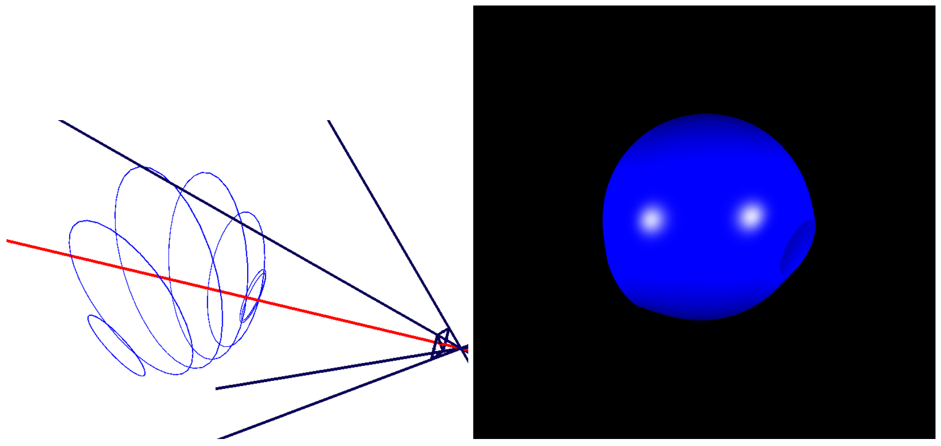

Fig. 5. Left: A scene composed of only an evolved surface in blue and a camera. Right: The rendering of the scene from the camera

6 Examples of ray tracing simple objects and evolved surfaces

Putting together the material from previous sections we can now raytrace both simple objects and evolved surfaces. Figure 2 shows an example of simple objects, spheres, planes and disks being rendered. Figure 5 shows an example of an evolved surface being rendered on its own. The class of surfaces that are able to be generated with the interpolation of circles is large and Figure 6 shows a more unusual surface being rendered in a scene with a sphere and a plane.

7 Summary and Conclusions

In this paper we have outlined the basic workings of a CGA ray tracer that can render geometric primitives as well as more advanced interpolated surfaces defined by two circles and an *evolution* parameter, α. We have shown how the ray-surface intersections and surface normals can be determined for these and additionally, the analytic form of the normals for such surfaces promises to provide us with interesting future tools to relate CGA quantities with principal, gaussian and mean curvatures, as well as other important differential geometry concepts.

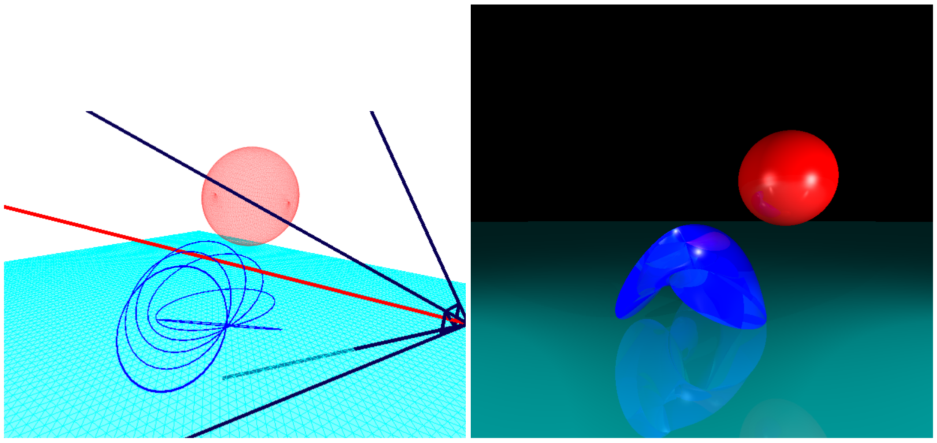

Fig. 6. Left: A scene composed of a ground plane in cyan, an evolved surface in blue and a sphere in red. Right: The rendering of the scene from the camera.

References

1. Hestenes, D.: Old Wine in New Bottles: A New Algebraic Framework for Computational Geometry. In: Corrochano E.B., Sobczyk G. (eds) Geometric Algebra with Applications in Science and Engineering. Birkhäuser, Boston, MA. (2001)
2. Lasenby, A., Lasenby, J. and Wareham, R.: A Covariant Approach to Geometry Using Geometric Algebra., Cambridge University Engineering Department (2004).
3. Dorst, L., Fontijne, D. and Mann, S.: Geometric algebra for computer science: an object-oriented approach to geometry. 1st edn. Elsevier; Morgan Kaufmann, Amsterdam (2007).
4. Breuils, S.,Nozick, V. and Fuchs, L.: GARAMON: GEOMETRIC ALGEBRA LIBRARY GENERATOR. In: AACA: Topical Collection AGACSE 2018, IMECC UNICAM, Campinas, Brazil (2018).
5. Hildenbrand, D.: Geometric Computing in Computer Graphics using Conformal Geometric Algebra. Computers & Graphics, Vol.29,No.5, pp. 802–810. (2005).
6. Wareham, R.J and Lasenby, J.: Generating Fractals Using Geometric Algebra. Advances in Applied Clifford Algebras 21(3):647-659. (2011).
7. Hadfield, H. and Lasenby, J.: Direct linear interpolation of conformal geometric objects. In: AACA: Topical Collection AGACSE 2018, IMECC UNICAM, Campinas, Brazil (2018).
8. Dorst, L., Valkenburg, R.: Square Root and Logarithm of Rotors in 3D Conformal Geometric Algebra Using Polar Decomposition. In: Guide to Geometric Algebra in Practice. pp. 81-104. Springer, London (2011)
9. Blinn, J.: Models of light reflection for computer synthesized pictures. ACM SIGGRAPH Computer Graphics **11**(2), 192–198 (1977)
10. Doran, C., Lasenby, A.: Geometric Algebra for Physicists. 1st edn. Cambridge University Press, Cambridge (2003)