Applications of Conformal Geometric Algebra in Computer Vision and Graphics

Rich Wareham, Jonathan Cameron, and Joan Lasenby

Cambridge University Engineering Department,
Trumpington St., Cambridge, CB2 1PZ, United Kingdom

Abstract. This paper introduces the mathematical framework of conformal geometric algebra (CGA) as a language for computer graphics and computer vision. Specifically it discusses a new method for pose and position interpolation based on CGA which firstly allows for existing interpolation methods to be cleanly extended to pose and position interpolation, but also allows for this to be extended to higher-dimension spaces and all conformal transforms (including dilations). In addition, we discuss a method of dealing with conics in CGA and the intersection and reflections of rays with such conic surfaces. Possible applications for these algorithms are also discussed.

1 Introduction

Since its inception in the mid-1970s, Computer Graphics (CG) has almost universally used linear algebra as its mathematical framework. This may be due to two factors; most early practitioners of computer graphics were mathematicians familiar with it and linear algebra provided a compact, efficient way of representing points, transformations, lines, etc.

As computing power becomes cheaper, the opportunity arises to investigate new frameworks for CG which, although not providing the time/space efficiency of linear algebra, may provide a conceptually simpler system or one of greater analytical power.

We intend to introduce a system that makes use of Clifford Algebras and in particular the geometric interpretation introduced by [14], termed Geometric Algebra. We then describe the use of this system for both pose and position interpolation and for the reflection and intersection of rays with conics.

1.1 Brief Overview of Geometric Algebra

We shall assume a basic familiarity with Clifford Algebras and merely describe the mechanism whereby they may be used to perform geometric operations. There already exist many introductions to Geometric Algebra that may be consulted [13, 9, 17].

We first write down a basis which can generate all elements of the Clifford Algebra with vector elements in $\mathbb{R}^2$ through linear combination

$$\{1, e_1, e_2, e_1 e_2\} \tag{1}$$

H. Li, P. J. Olver and G. Sommer (Eds.): IWMM 2004, LNCS 3519, pp. 329–349, 2005.

where e_1, e_2 are the usual orthonormal Euclidean basis vectors and hence $e_1 e_2 = e_1 \cdot e_2 + e_1 \wedge e_2 = e_1 \wedge e_2$. For convenience we shall denote $e_i e_j$ as e_{ij} and, generally, $e_i e_j \ldots e_k$ as $e_{ij\ldots k}$. A general linear sum of these components is termed a *multivector* whereas a sum of only grade-n components is called a *n-vector*. This paper will use the convention of writing multivectors, bivectors, trivectors and higher-grade elements in upper-case and vectors and scalars in lower-case.

Firstly, note that

$$(e_1 e_2)^2 = e_1 e_2 e_1 e_2 = -e_1 e_2 e_2 e_1 = -1 \tag{2}$$

and thus we have an element of the algebra which squares to -1. As shown by [12, 9] this can be identified with the unit imaginary $i = \sqrt{-1}$ in $\mathbb{C}$ and, as such, the highest-grade basis-element in a geometric algebra is often denoted I and referred to as the *pseudoscalar*.

Note that in spaces with dimension n the maximum grade object possible has grade n and the n-vector $e_1 \wedge \ldots \wedge e_n = I$ is therefore the pseudoscalar. The result of the product xI is termed the *dual* of x, denoted as x^*.

In [12] it was also shown that many of the identities and theorems dealing with complex numbers have a direct analogue in Clifford Algebra. Therefore a geometrical interpretation, which was named *Geometric Algebra*, was suggested.

It can be shown [17, 14] that there is a general method for rotating a vector x involving the formation of a multivector of the form $R = e^{-B\phi/2}$. This represents an anticlockwise rotation ϕ in a plane specified by the bivector B. The transformation is given by

$$x \mapsto RxR^{-1}. \tag{3}$$

We refer to these bivectors which have a rotational effect as *rotors*. The computation of the inverse of a multivector is rather computationally intensive in general (and indeed can require a full 2^n-dimension matrix inversion for a space of dimension n). To combat this we define the *reversion* of a n-vector $X = e_i e_j \ldots e_k$ as $\tilde{X} = e_k \ldots e_j e_i$, i.e. the literal reversion of the components. Since the reversion of $e_i e_j$ is $e_j e_i = -e_i e_j$, and by looking at the expression for R, it is clear that $\tilde{R} \equiv R^{-1}$ for rotors. Computing $\tilde{R}$ is easier since it involves only a sign change for some orthogonal elements of a multivector.

It is worth comparing this method of rotation to quaternions. The three bivectors $B_1 = e_{23}, B_2 = e_{31}$ and $B_3 = e_{12}$ act identically to the three imaginary components of quaternions, $\mathbf{i}, -\mathbf{j}$ and $\mathbf{k}$ respectively. The sign difference between B_2 and $\mathbf{j}$ is due to the fact that the quaternions are not derived from the usual right-handed orthogonal co-ordinate system.

A particular rotation is represented via the quaternion q given by

$$q = q_0 + q_1 \mathbf{i} + q_2 \mathbf{j} + q_3 \mathbf{k}$$

where $q_0^2 + q_1^2 + q_2^2 + q_3^2 = 1$. Interpolation between rotations is then performed by interpolating each of the q_i over the surface of a four-dimensional hypersphere. If we are interpolating between unit quaternions q_0 and q_1 the SLERP interpolation is

$$q = \begin{cases} q_0(q_0^{-1}q_1)^\lambda & \text{if } q_0 \cdot q_1 \geq 0 \\ q_0(q_0^{-1}(-q_1))^\lambda & \text{otherwise} \end{cases} \tag{4}$$

where λ varies in the range $(0, 1)$ [19, 23].

Recalling that, in complex numbers, the locus of $\exp(i\frac{\phi}{2})$ is the unit circle, it is somewhat simple to show that, for some bivector B, where $B^2 = -1$, the locus of the action of $\exp(-B\frac{\phi}{2})$ upon a point with respect to varying ϕ is also a circle in the plane of B. Therefore if we consider some rotations $R_1, R_2 = \exp(kB)R_1$, where k is a scalar and B is some normalised bivector, it is easy to see after some thought that the quaternionic interpolation is exactly given by

$$R_\lambda = (R_2\tilde{R}_1)^\lambda R_1 = \exp(\lambda kB)R_1$$

where λ, the interpolation parameter, varies in the range $(0, 1)$. A further moment's thought will reveal that this method, unlike quaternionic interpolation, is not confined to three dimensions but instead readily generalises to higher-dimensions.

Reflections. Reflections are particularly easy to represent in Geometric Algebra. Reflecting a vector a in a plane with unit normal $\hat{n}$ we obtain a reflected vector a' given by

$$a' = -\hat{n}a\hat{n}$$

which may easily be verified by expanding out to

$$a' = a - 2(\hat{n} \cdot a)\hat{n}$$

and noting that this does indeed give the reflection of a. This pattern of 'sandwiching' an object between two others is often found in GA-based algorithms and can be thought of as representing the reflection of the central object in the 'sandwiching' object.

1.2 Conformal Geometric Algebra (CGA)

In the Conformal Model [14, 13, 18, 17, 20] we extend the space by adding two additional basis vectors. We first define the *signature*, (p, q) of a space $\mathcal{A}(p, q)$ with basis vectors, $\{e_i\}$, such that $e_i^2 = +1$ for $i = 1, ..., p$ and $e_j^2 = -1$ for $j = p + 1, ..., p + q$. For example $\mathbb{R}^3$ would be denoted as $\mathcal{A}(3, 0)$. We extend $\mathcal{A}(3, 0)$ so that it becomes mixed signature and is defined by the basis

$$\{e_1, e_2, e_3, e, \bar{e}\}$$

where e and $\bar{e}$ are defined so that

$$e^2 = 1, \quad \bar{e}^2 = -1, \quad e \cdot \bar{e} = 0$$

$$e \cdot e_i = \bar{e} \cdot e_i = 0 \quad \forall\, i \in \{1, 2, 3\}.$$

This space is denoted as $\mathcal{A}(4,1)$. In general a space $\mathcal{A}(p,q)$ is extended to $\mathcal{A}(p+1,q+1)$. We may now define the vectors n and $\bar{n}$:

$$n = e + \bar{e}, \quad \bar{n} = e - \bar{e}$$

It is simple to show by direct substitution that both n and $\bar{n}$ are *null vectors* (i.e. $n^2 = \bar{n}^2 = 0$). It can be shown [14, 17] that rigid body transforms may be represented conveniently via rotors in this algebra.

We shall also make use of a transform based upon that proposed by Hestenes in [14]. The original transform is dimensionally inconsistent however so we introduce a fundamental unit length scale, λ, into the equation to make it dimensionally consistent

$$F(x) = \frac{1}{2\lambda^2}\left[x^2 n + 2\lambda x - \lambda^2 \bar{n}\right] \equiv X \tag{5}$$

where λ is usually set to be unity. At this point, notice that we can identify the origin with $\bar{n}$ since $F(0) \propto \bar{n}$ and that as $x \to \infty$, $F(x)$ becomes n. We may also find the inverse transform

$$F^{-1}(X) = \lambda(X \wedge n) \cdot e \tag{6}$$

Rotations. As one might expect from their relation to complex numbers, there exists an element of the algebra which performs rotation in the plane. These pure-rotation rotors all have the form e^{-B} where B has only components of the form e_{ij}, $i,j \in \{1,2,3\}$.

A useful property of this mapping is that pure-rotation rotors retain their properties as can be shown by considering the effect of a rotor $R = \exp(-\frac{\phi}{2}e_{ij})$, $i,j \in \{1,2,3\}$ upon $F(x)$. Setting $\lambda = 1$ for the moment,

$$RF(x)\tilde{R} = \frac{1}{2}R(x^2 n + 2x - \bar{n})\tilde{R} \tag{7}$$

$$= \frac{1}{2}\left(x^2 Rn\tilde{R} + 2Rx\tilde{R} - R\bar{n}\tilde{R}\right) = F(Rx\tilde{R}) \tag{8}$$

since rotors leave n and $\bar{n}$ invariant and $(Rx\tilde{R})^2 = x^2$.

Translations. The translation rotor T_a is defined [17, 18] as

$$T_a = \exp\left[\frac{na}{2}\right] = 1 + \frac{na}{2}$$

and will transform a null-vector representation of the vector x to the null-vector representation of $x_a = x + a$ in the following manner:

$$F(x_a) = T_a F(x)\tilde{T_a}$$

Dilations. It can also be shown [17, 18] that the rotor $D_\alpha = \exp(\alpha e\bar{e}/2)$ has the effect of dilating x by a factor of $e^{-\alpha}$, i.e. $D_\alpha F(x)\tilde{D}_\alpha \propto F(e^{-\alpha}x)$

Table 1. Representations of various geometric objects

Line – $L = X_1 \wedge X_2 \wedge n$	Circle – $C = X_1 \wedge X_2 \wedge X_3$
Plane – $\Phi = X_1 \wedge X_2 \wedge X_3 \wedge n$	Sphere – $\Sigma = X_1 \wedge X_2 \wedge X_3 \wedge X_4$

Inversions. Finally, inversions ($x \mapsto x^2/x$) may be represented [17, 18] as $F(x) \mapsto eF(x)e$. Although this will not be discussed here, this becomes particularly important when considering non-Euclidean geometry.

This mapping has provided a similar advantage to that of homogeneous coordinates, namely that rigid body transforms become multiplicative and any such transform may be represented by a rotor. In addition, transforms followed by, or preceded by, a dilation or inversion may also be represented multiplicatively.

Representation of Geometric Objects. We have shown how rigid body transformations may be performed on a vector x by operating on its null-vector representation $F(x)$. We may also ask what form the multivector M takes if the solutions of $F(x) \wedge M = 0$ lie on a circle, sphere, line or plane. It has been shown in [17, 18] that the form of M depends only on the null-vector representation of points which lie on the object. The forms of M for various objects are summarised in Table 1. Note that if we identify the vector n with the point at infinity, a line is just a special case of a circle which passes through infinity and a plane is equally a special case of a sphere. It becomes convenient, therefore, to group planes and spheres by the collective term *generalised spheres* and, similarly, to define a *generalised circle* as either a circle or a line.

Suppose we have a generalised sphere which passes through the points $x_1, \cdots, x_4$. Hence, for all points x on the object, $F(x) \wedge M = 0$ where

$$M = F(x_1) \wedge F(x_2) \wedge F(x_3) \wedge F(x_4). \tag{9}$$

Now consider the effect of a rotor R upon M

$$RM\tilde{R} = RF(x_1)\tilde{R} \wedge RF(x_2)\tilde{R} \wedge RF(x_3)\tilde{R} \wedge RF(x_4)\tilde{R} \tag{10}$$

as $R(a \wedge b)\tilde{R} = R(a)\tilde{R} \wedge R(b)\tilde{R}$. Furthermore we can say

$$RM\tilde{R} = F(Rx_1\tilde{R}) \wedge F(Rx_2\tilde{R}) \wedge F(Rx_3\tilde{R}) \wedge F(Rx_4\tilde{R}) \tag{11}$$

which represents a generalised sphere passing through the transformed points $Y_i = RX_i\tilde{R}$. Clearly, if R represents a conformal transformation, the generalised sphere represented by M is similarly transformed.

Intersections may be performed efficiently using the *meet* operator. The general meet operation can be found by noting that for r-grade and s-grade blades M_r and M_s, a point, X, on the intersection must satisfy $X \wedge M_r = X \wedge M_s = 0$ which can then be shown to be equivalent to

$$X \wedge \left[\langle M_r M_s \rangle_{2l-r-s}\right]^* = 0$$

where $[\,\cdot\,]^*$ denotes multiplication by the pseudoscalar, l is the dimension of the space (in the case of $A(4,1)$, $l = 5$) and $\langle X \rangle_i$ denotes the extraction of the i-grade component from X. If we define the meet operator

$$M_r \vee M_s = \left[\langle M_r M_s \rangle_{2l-r-s}\right]^*$$

then we can interpret the meet of two objects as their intersection in many cases.

The key feature of this approach is that we have placed few constraints on the form of M_r and M_s and thus we can intersect objects in a fairly general manner instead of using object-specific algorithms.

2 Pose and Position Interpolation

In this section we shall use the term *displacement rotor* to refer to a rotor which performs some rigid-body transform. Referring to the displacement rotors presented above, we see that all of them have a common form; they are all exponentiated bivectors. Rotations are generated by bivectors with no component parallel to n and translations by a bivector with no components perpendicular to n. We may therefore postulate that all displacement rotors (we shall deal with rotors including a dilation later) can be expressed as

$$R = \exp(B)$$

where B is the sum of two bivectors, one formed from the outer product of two vectors which have no components parallel to e or $\bar{e}$. The other is formed from the outer product with n of vectors with no components parallel to e or $\bar{e}$. The effect of this is to separate the basis bivectors of B into bivectors with components of the form $e_i \wedge e_j$, $e_i \wedge e$ and $e_i \wedge \bar{e}$.

We shall proceed assuming that all displacement rotors can be written as the exponentiation of a bivector of the form $B = ab + cn$ where a, b and c are *spatial* vectors, i.e. if $n \in \mathcal{A}(m+1,1)$ then $\{a,b,c\} \in \mathbb{R}^m$. It is clear that the set of all B is some linear subspace of all the bivectors.

We now suppose that we may interpolate rotors by defining some function $\ell(R)$ which acts upon rotors to give the generating bivector element. We then perform direct interpolation of this generator. We postulate that direct interpolation of such bivectors, as in the reformulation of quaternionic interpolation above, will give some smooth interpolation between the displacements. It is therefore a defining property of $\ell(R)$ that

$$R \equiv \exp(\ell(R)) \tag{12}$$

and so $\ell(R)$ may be considered to act as a logarithm-like function in this context. It is worth noting that $\ell(R)$ does not possess all the properties usually associated with logarithms, notably that, since $\exp(A)\exp(B)$ is not generally equal to $\exp(B)\exp(A)$ in non-commuting algebras, $\ell(\exp(A)\exp(B))$ cannot be equal to $A + B$ except in special cases.

To avoid the the risk of assigning more properties to $\ell(R)$ than we have shown, we shall resist the temptation to denote the function $\log(R)$. The most obvious property of $\log(\cdot)$ that $\ell(\cdot)$ doesn't possess is $\log(AB) = \log(A)+\log(B)$. This is clear since the geometric product is not commutative in general whereas addition is.

2.1 Form of exp(B) in Euclidean Space

The form of $\exp(B)$ may be derived after a little work. We start by assuming that B is of the form $B = \phi P + tn$ where $t \in \mathbb{R}^n$, ϕ is some scalar and P is a 2-blade where $P^2 = -1$ and it is formed from spatial vectors. We then define $t_\parallel$ to be the component of t lying in the plane of P and $t_\perp = t - t_\parallel$. From this assumption it can be shown [26, 25] that the form of $\exp(B)$ is

$$\exp(B) = [\cos(\phi) + \sin(\phi)P]\,[1 + t_\perp n] + \operatorname{sinc}(\phi)t_\parallel n$$

It is also shown in [25] that this can only represent an Euclidean translation and rotation rotor. It is worth exploring the geometric interpretation of this expression. me vector satisfying $a \cdot n = a \cdot P = 0$.

In [25] we discuss how to obtain a geometrical description of the action of the rotor in terms of the bivector B. We shall state this here without proof. We state that the action of the rotor

$$R = \exp\left(\frac{\psi}{2}P + \frac{tn}{2}\right)$$

is to translate along a vector $t_\perp$ which is the component of t which does not lie in the plane of P, rotate by ψ in the plane of P and finally translate along $t'_\parallel$ which is given by

$$t'_\parallel = -\operatorname{sinc}\left(\frac{\psi}{2}\right) t_\parallel \left(\cos\left(\frac{\psi}{2}\right) - \sin\left(\frac{\psi}{2}\right) P\right)$$

which is the component of t lying in the plane of P, rotated by $\psi/2$ in that plane.

2.2 Method for Evaluating $\ell(R)$

We have found a form for $\exp(B)$ given that B is in a particular form. Now we seek a method to take an arbitrary displacement rotor, $R = \exp(B)$ and reconstruct the original B. Should there exist a B for all possible R, we will show that our initial assumption that all displacement rotors can be formed from a single exponentiated bivector of special form is valid. We shall term this initial bivector the *generator* bivector (to draw a parallel with Lie algebras).

We can obtain the following identities for $B = (\psi/2)P + tn/2$ by simply considering the grade of each component of the exponential:

$$\begin{aligned}
\langle R\rangle_0 &= \cos\left(\frac{\psi}{2}\right) \\
\langle R\rangle_2 &= \sin\left(\frac{\psi}{2}\right) P + \cos\left(\frac{\psi}{2}\right) t_\perp n + \operatorname{sinc}\left(\frac{\psi}{2}\right) t_\parallel n \\
\langle R\rangle_4 &= \sin\left(\frac{\psi}{2}\right) P t_\perp n
\end{aligned}$$

It is somewhat straightforward to reconstruct $\psi, t_\perp$ and $t_\parallel$ from these components by partitioning a rotor as above. Once we have a method which gives the generator B for any displacement rotor R we have validated our assumption.

Theorem 1. *The inverse-exponential function $\ell(R)$ is given by*

$$\ell(R) = ab + c_\perp n + c_\parallel n$$

where

$$\begin{aligned}
\|ab\| &= \sqrt{|(ab)^2|} = \cos^{-1}(\langle R\rangle_0) \\
ab &= \frac{(\langle R\rangle_2 n)\cdot e}{\operatorname{sinc}(\|ab\|)} \\
c_\perp n &= -\frac{ab\,\langle R\rangle_4}{\|ab\|^2 \operatorname{sinc}(\|ab\|)} \\
c_\parallel n &= -\frac{ab\,\langle ab\,\langle R\rangle_2\rangle_2}{\|ab\|^2 \operatorname{sinc}(\|ab\|)}
\end{aligned}$$

Proof. It is clear from the above that the form of $\|ab\|$ is correct. We thus proceed to show the remaining equations to be true

$$\begin{aligned}
\langle R\rangle_2 &= \cos(\|ab\|)\, c_\perp n + \operatorname{sinc}(\|ab\|)\left[ab + c_\parallel n\right] \\
\langle R\rangle_2 n &= \operatorname{sinc}(\|ab\|)\, abn \\
(\langle R\rangle_2 n)\cdot e &= \operatorname{sinc}(\|ab\|)\, ab
\end{aligned}$$

and hence the relation for ab is correct.

$$\begin{aligned}
\langle R\rangle_4 &= \operatorname{sinc}(\|ab\|)\, abc_\perp n \\
ab\,\langle R\rangle_4 &= -\|ab\|^2 \operatorname{sinc}(\|ab\|)\, c_\perp n
\end{aligned}$$

and hence the relation for $c_\perp n$ is correct.

$$\begin{aligned}
\langle R\rangle_2 &= \cos(\|ab\|)\, c_\perp n + \operatorname{sinc}(\|ab\|)\left[ab + c_\parallel n\right] \\
ab\,\langle R\rangle_2 &= \cos(\|ab\|)\, abc_\perp n + \operatorname{sinc}(\|ab\|)\left[abc_\parallel n - \|ab\|^2\right] \\
\langle ab\,\langle R\rangle_2\rangle_2 &= \operatorname{sinc}(\|ab\|)\, abc_\parallel n
\end{aligned}$$

and hence the relation for $c_\parallel n$ is correct.

3 Interpolation via Logarithms

We have shown that any displacement of Euclidean geometry may be mapped smoothly onto a linear subspace of the bivectors. This immediately suggests applications to smooth interpolation of displacements. Consider a set of poses we wish to interpolate, $\{P_1, P_2, ..., P_n\}$ and a set of rotors which transform some origin pose to these target poses, $\{R_1, R_2, ..., R_n\}$. We may map these rotors onto the set of bivectors $\{\ell(R_1), \ell(R_2), ..., \ell(R_n)\}$ which are simply points in some linear subspace of the bivectors. We may now choose any interpolation of these bivectors which lies in this space and for any bivector on the interpolant, B'_λ, we can compute a pose, $\exp(B'_\lambda)$. We believe this method is more elegant and conceptually simpler than many other approaches based on Lie-algebras [11, 21, 22, 5].

Another interpolation scheme is to have the poses defined by a set of chained rotors so that $\{P_1, P_2, ..., P_n\}$ is represented by

$$\{R_1, \Delta R_1 R_1, \Delta R_2 R_2, ..., \Delta R_n R_n\}$$

where $R_i = \Delta R_{i-1} R_{i-1}$ as in figure 1. Using this scheme the interpolation between pose R_i and R_{i+1} involves forming the rotor $R_{i,\lambda} = \exp(B_{i,\lambda}) R_{i-1}$ where $B_{i,\lambda} = \lambda \ell(\Delta R_{i-1})$ and λ varies between 0 and 1 giving $R_{i,0} = R_{i-1}$ and $R_{i,1} = R_i$.

We now investigate two interpolation schemes which interpolate through target poses, ensuring that each pose is passed through. This kind of interpolation is often required for key-frame animation techniques. The first form of interpolation is piece-wise linear interpolation of the relative rotors (the latter case above). The second is direct quadratic interpolation of the bivectors representing the final poses (the former case).

3.1 Piece-Wise Linear Interpolation

Direct piece-wise linear interpolation of the set of relative bivectors is one of the simplest interpolation schemes we can consider. Consider the example shown in figure 1. Here there are three rotors to be interpolated. We firstly find a rotor, ΔR_n which takes us from rotor R_n to the next in the interpolation sequence, R_{n+1}.

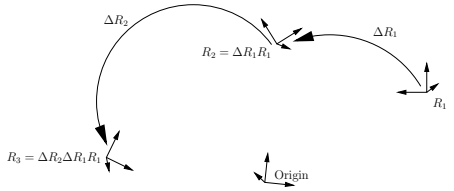

Fig. 1. Rotors used to piece-wise linearly interpolate between key-rotors

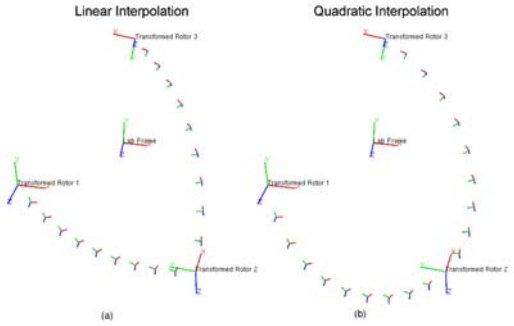

Fig. 2. Examples of (a) piece-wise linear and (b) quadratic interpolation for 3 representative poses

$$R_{n+1} = (\Delta R_n) R_n$$
$$\Delta R_n = R_{n+1} \tilde{R}_n.$$

We then find the bivector, ΔB_n which generates $\Delta R_n = \exp(\Delta B_n)$. Finally we form a rotor interpolating between R_n and R_{n+1}:

$$R_{n,\lambda} = \exp(\lambda \Delta B_n) R_n$$

where λ is in the range $[0, 1]$ and $R_{n,0} = R_n$ and $R_{n,1} = R_{n+1}$. Clearly this interpolation scheme changes abruptly at interpolation points, something which is reflected in the resulting interpolation as shown in figure 2. It is interesting to note, after a moment's thought, that for pure-rotation rotors this reduces *exactly* to the quaternionic SLERP interpolation in equation 4.

3.2 Quadratic Interpolation

Another simple form for interpolation is the quadratic interpolation where a quadratic is fitted through three interpolation points, $\{B_1, B_2, B_3\}$ with an interpolation parameter varying in the range $(-1, +1)$:

$$B'_\lambda = \left(\frac{B_3 + B_1}{2} - B_2 \right) \lambda^2 + \frac{B_3 - B_1}{2} \lambda + B_2$$

giving

$$B'_{-1} = B_1, \quad B'_0 = B_2 \quad \text{and} \quad B'_{+1} = B_3$$

This interpolation varies smoothly through B_2 and is reflected in the final interpolation, as shown in figure 2. Extensions to the quadratic interpolation for more than three interpolation points, such as smoothed quadratic interpolation [7] or even a traditional cubic spline or Bézier interpolation are readily available.

It is worth noting that each of the methods described above may be performed using either direct interpolation of the bivector $\ell(R)$ corresponding to a rotor R or by interpolating the relative rotors which take one rotor to another. It is not yet clear which will give the best results and indeed it is probably application dependent.

3.3 Form of the Interpolation

We now derive a clearer picture of the precise form of a simple linear interpolation between two rotors in order to relate the interpolation to existing methods used in mechanics and robotics. We will consider the method used above whereby the rotor being interpolated takes one pose to another.

Path of the Linear Interpolation. Since we have shown that $\exp(B)$ is indeed a rotor, it follows that any Euclidean pure-translation rotor will commute with it. Thus we only need consider the interpolant path when interpolating from the origin to some other point since any other interpolation can be obtained by simply translating the origin to the start point. This location independence of the interpolation is a desirable property in itself but also provides a powerful analysis mechanism.

We have identified in section 2.1 the action of the $\exp(B)$ rotor in terms of $\psi, P, t_{\parallel}$ and $t_{\perp}$. We now investigate the resulting interpolant path when interpolating from the origin. We shall consider the interpolant $R_\lambda = \exp(\lambda B)$ where λ is the interpolation co-ordinate and varies from 0 to 1. For any values of $\psi, P, t_{\parallel}$ and $t_{\perp}$,

$$\lambda B = \frac{\lambda\psi}{2}P + \frac{\lambda(t_\perp + t_\parallel)n}{2}$$

from our expansion of $\exp(B)$ we see that the action of $\exp(\lambda B)$ is a translation along $\lambda t_{\perp}$, a rotation by $\lambda\psi$ in the plane of P and finally a translation along

$$t'_{\parallel} = -\text{sinc}\left(\frac{\lambda\psi}{2}\right)\lambda t_{\parallel}\left(\cos\left(\frac{\lambda\psi}{2}\right) - \sin\left(\frac{\lambda\psi}{2}\right)P\right).$$

We firstly resolve a three dimensional orthonormal basis, $\{a, b, t_{\perp}\}$, relative to P where a and b are orthonormal vectors in the plane of P and hence $P = ab$. We may now express $t_{\parallel}$ as $t_{\parallel} = t^a a + t^b b$ where $t^{\{a,b\}}$ are suitably valued scalars.

The initial action of $\exp(B)$ upon a frame centred at the origin is therefore to translate it to $\lambda t_{\perp}$ followed by a rotation in the plane of P. Due to our choice of starting point, this has no effect on the frame's location (but will have an effect on the pose, see the next section).

Finally there is a translation along $t'_{\parallel}$ which, using $c = \cos\left(\frac{\lambda\psi}{2}\right)$ and $s = \sin\left(\frac{\lambda\psi}{2}\right)$, can be expressed in terms of a and b as

$$\begin{aligned} t'_{\parallel} &= -\frac{2s}{\lambda\psi}\lambda(t^a a + t^b b)(c - sab) \\ &= -\frac{2s}{\psi}\left[c(t^a a + t^b b) + s(t^b a - t^a b)\right] \\ &\equiv -\frac{2s}{\psi}\left[a(t^a c + t^b s) + b(t^b c - t^a s)\right]. \end{aligned}$$

The position, r_λ, of the frame at λ along the interpolation is therefore

$$r_\lambda = -\frac{2s}{\psi}(a(t^a c + t^b s) + b(t^b c - t^a s)) + \lambda t_{\perp}$$

which can easily be transformed via the harmonic addition theorem to

$$r_\lambda = -\frac{2s}{\psi}\alpha\left[a\cos\left(\frac{\lambda\psi}{2}+\beta_1\right)+b\cos\left(\frac{\lambda\psi}{2}+\beta_2\right)\right]+\lambda t_\perp$$

where $\alpha^2 = (t^a)^2 + (t^b)^2$, $\tan\beta_1 = -\frac{t^b}{t^a}$ and $\tan\beta_2 = -\frac{-t^a}{t^b}$. It is easy, via geometric construction or otherwise, to verify that this implies that $\beta_2 = \beta_1 + \frac{\pi}{2}$. Hence $\cos(\theta+\beta_2) = -\sin(\theta+\beta_1)$. We can now express the frame's position as

$$r_\lambda = -\frac{2\alpha}{\psi}\left[a\sin\left(\frac{\lambda\psi}{2}\right)\cos\left(\frac{\lambda\psi}{2}+\beta_1\right)-b\sin\left(\frac{\lambda\psi}{2}\right)\sin\left(\frac{\lambda\psi}{2}+\beta_1\right)\right]+\lambda t_\perp$$

which can be re-arranged to give

$$\begin{aligned} r_\lambda &= -\frac{\alpha}{\psi}\left[a\left(\sin\left(\lambda\psi+\beta_1\right)-\sin\beta_1\right)+b\left(\cos\left(\lambda\psi+\beta_1\right)-\cos\beta_1\right)\right]+\lambda t_\perp \\ &= -\frac{\alpha}{\psi}\left[a\sin\left(\lambda\psi+\beta_1\right)+b\cos\left(\lambda\psi+\beta_1\right)\right]+\frac{\alpha}{\psi}\left[a\sin\beta_1+b\cos\beta_1\right]+\lambda t_\perp \end{aligned}$$

noting that in the case $\psi \to 0$, the expression becomes $r_\lambda = \lambda t_\perp$ as one would expect. Since a and b are defined to be orthonormal, the path is clearly some cylindrical helix with the axis of rotation passing through $\alpha/\psi\,[a\sin\beta_1 + b\cos\beta_1]$.

It is worth noting a related result in screw theory, Chasles' theorem [2], which states that a general displacement may be represented using a screw motion (cylindrical helix) such as we have derived. Screw theory is widely used in mechanics and robotics and the fact that the naïve linear interpolation generated by this method is indeed a screw motion suggests that applications of this interpolation method may be wide-ranging, especially since this method allows many other forms of interpolation, such as Bézier curves or three-point quadratic to be performed with equal ease. Also the pure rotation interpolation given by this method reduces exactly to the quaternionic or Lie group interpolation result allowing the method to easily extend existing ones based upon these interpolations.

Pose of the Linear Interpolation. The pose of the transformed frame is unaffected by pure translation and hence the initial translation by $\lambda t_\perp$ has no effect. The rotation by $\lambda\psi$ in the plane, however, now becomes important. The subsequent translation along $t'_\parallel$ also has no effect on the pose. We find, therefore, that the pose change λ along the interpolant is just the rotation rotor $R_{\lambda\psi,P}$.

3.4 Interpolation of Dilations

In certain circumstances it is desirable to add in the ability to interpolate dilations. It can be shown [6] that this can be done by extending the form of the bivector, B, which we exponentiate, as follows

$$B = \phi P + tn + \omega N$$

where $N = e\bar{e}$. This bivector form is now sufficiently general [6] to be able to represent dilations as well. In this case obtaining the exponentiation and logarithm function is somewhat involved [6]. We obtain finally that

$$\begin{aligned}
&\exp(\phi P + tn + \omega N) \\
&\quad = (\cos(\phi) + \sin(\phi)P)\,(\cosh(\omega) + \sinh(\omega)N + \mathrm{sinhc}(\omega)t_\perp n) \\
&\qquad +(\omega^2+\phi^2)^{-1}[-\omega\sin(\phi)\cosh(\omega) + \phi\cos(\phi)\sinh(\omega)]P \\
&\qquad +(\omega^2+\phi^2)^{-1}[\omega\cos(\phi)\sinh(\omega) + \phi\sin(\phi)\cosh(\omega)]t_\parallel n
\end{aligned}$$

where $\mathrm{sinhc}(\omega) = \omega^{-1}\sinh(\omega)$. Note that this expression reduces to the original form for $\exp(B)$ when $\omega = 0$, as one would expect.

It is relatively easy to use the above expansion to derive a logarithm-like inverse function.

If we let $R = \exp(B)$ then we may recreate B from R using the method presented below. Here we use $\langle R|e_i\rangle$ to represent the component of R parallel to e_i, i.e. $\langle R|N\rangle = \langle R|e_{45}\rangle$ in 3-dimensions. We also use $\langle R\rangle_i$ to represent the i-th grade-part of R and $S(X)$ to represent the 'spatial' portion of X (i.e. those components not parallel to e and $\bar{e}$).

$$\begin{aligned}
\omega &= \tanh^{-1}\left(\tfrac{\langle R|N\rangle}{\langle R|1\rangle}\right) & W &= \langle R\rangle_2 - \cos(\phi)\sinh(\omega)N \\
\phi &= \cos^{-1}\left(\tfrac{\langle R|N\rangle}{\sinh(\omega)}\right) & &\quad - \sin(\phi)\cosh(\omega)P \\
P &= \tfrac{S(\langle R\rangle_2)}{\sin(\phi)\cosh(\omega)} & &\quad - \cos(\phi)\mathrm{sinhc}(\omega)t_\perp n \\
t_\perp &= -\tfrac{\langle R\rangle_4 - \sin(\phi)\sinh(\omega)PN}{\sin(\phi)\mathrm{sinhc}(\omega)}\left(\tfrac{P\bar{n}}{2}\right) & X &= -\omega\sin(\phi)\cosh(\omega) + \phi\cos(\phi)\sinh(\omega) \\
t &= t_\parallel + t_\perp & Z &= \omega\cos(\phi)\sinh(\omega) + \phi\sin(\phi)\cosh(\omega) \\
& & t_\parallel &= \tfrac{(-XP+Z)}{\sin^2(\phi)\cosh^2(\omega)+\cos^2(\phi)\sinh^2(\omega)}W
\end{aligned}$$

Path of Interpolation. We now consider the path resulting from the simple linear interpolation of both pose and dilation using the results derived above. In [6] a full derivation is given and here we present only the result. As before, the derivation proceeds by considering the position, r_λ, of the frame formed by applying the interpolation rotor $R = \exp(\lambda(\phi P + tn + \omega N))$ to $\bar{n}$ (the origin), with the interpolation parameter λ varying from 0 to 1. After a little work we obtain the path of the interpolation as

$$\begin{aligned}
r_\lambda = &-\frac{1}{\omega}|t_\perp|\hat{t}_\perp - \frac{\omega}{\omega^2+\phi^2}|t_\parallel|\hat{t}_\parallel + \frac{\phi}{\omega^2+\phi^2}|t_\parallel|\hat{t}_\pm + e^{-2\lambda\omega}\cdot \\
&\left(\frac{|t_\perp|}{\omega}\hat{t}_\perp + \frac{\cos\left(2\lambda\phi + \tan^{-1}\left(\frac{\phi}{\omega}\right)\right)}{\sqrt{\omega^2+\phi^2}}|t_\parallel|\hat{t}_\parallel - \frac{\sin\left(2\lambda\phi + \tan^{-1}\left(\frac{\phi}{\omega}\right)\right)}{\sqrt{\omega^2+\phi^2}}|t_\parallel|\hat{t}_\pm\right)
\end{aligned}$$

where $\hat{t}_\parallel$ is the normalised component of t parallel to P, $\hat{t}_\perp$ is the normalised component perpendicular to P and $\hat{t}_\pm$ is a unit vector perpendicular to both $\hat{t}_\parallel$ and $\hat{t}_\perp$. This is the equation of a conical helix.

It can be shown [6] that when setting the dilation component, ω, to zero this path is equivalent to that derived for pose interpolation.

3.5 Applications

The method outlined is applicable to any problem which requires the smooth interpolation of pose. We have chosen to illustrate an application in mesh deformation. Smooth mesh deformation is often required in medical imaging applications [8] or video coding [15]. Here we use it to deform a 3d mesh in a manner which has a visual effect similar to 'grabbing' a corner of the mesh and 'twisting' it into place. We do this by specifying a set of 'key-rotors' which certain parts of

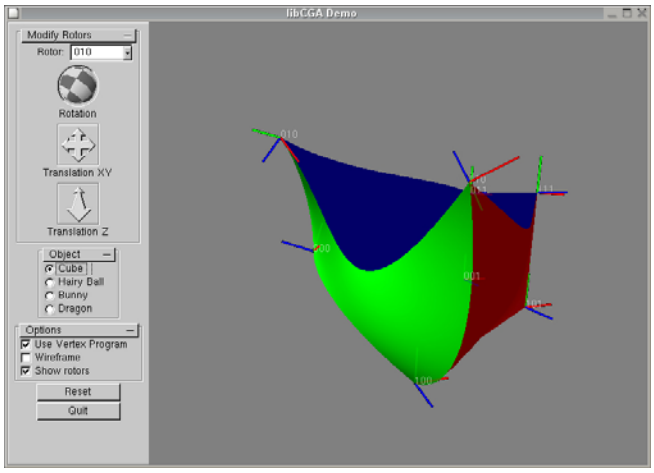

Fig. 3. A cube distorted via the linear interpolation of rotors specifying its corner vertices

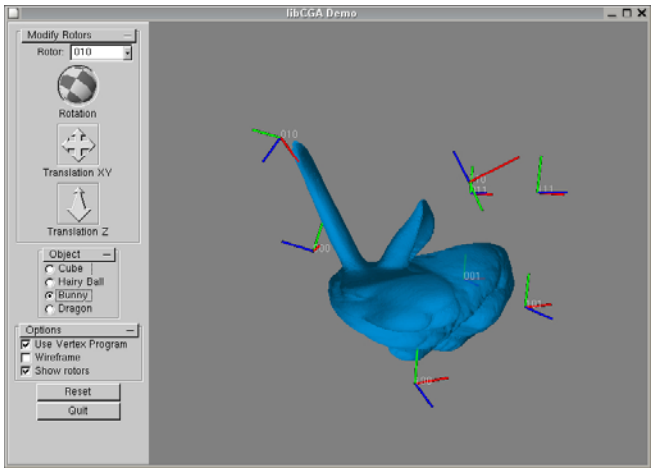

Fig. 4. The Stanford Bunny[24] distorted by the same rotors

the mesh must rotate and translate to coincide with. Our implementation takes advantage of the hardware acceleration offered by the Graphics Processing Units (GPUs) available on today's consumer-level graphics hardware. A full discussion of the method will appear elsewhere.

We believe this method leads to images which are intuitively related to the rotors specifying the deformation. Furthermore, the method need not only be applied to meshes with simple geometry and can readily be applied to meshes with complex geometry without any tears or other artifacts appearing in the mesh. Figures 3 and 4 illustrate the method in the case of a cube and a mesh with approximately 36,000 vertices.

4 Conics

So far we have been limited in which objects we can manipulate and intersect in CGA. Specifically we can only deal with planes, circles, spheres, lines and points. In this section we consider a method of extending our approach to conics and intersections thereof. We shall proceed by considering the 3-dimensional Euclidean case but this method may be generalised to higher dimensions if required; here we follow the procedure outlined in [16]. We shall begin by considering the set of points on the unit sphere at the origin:

$$\{c_\theta e_1 + s_\theta c_\phi e_2 + s_\theta c_\phi e_3 \,:\, \theta \in (0, 2\pi],\, \phi \in (0, \pi]\}$$

where $c_\theta = \cos(\theta)$, $s_\theta = \sin(\theta)$ and similarly for s_ϕ and c_ϕ. We apply the transform $F(\cdot)$ to obtain the set, S, of representations for these points:

$$S = \left\{ \frac{n}{2} + c_\theta e_1 + s_\theta c_\phi e_2 + s_\theta c_\phi e_3 - \frac{\bar{n}}{2} \,:\, \theta \in (0, 2\pi],\, \phi \in (0, \pi] \right\}$$

and consider the effect of the transform from [16] below

$$C_\alpha(X) = \beta \left[X + \frac{\alpha}{2}(X \cdot e_1) \wedge \bar{n} \right]$$

which we shall term the *conic transform*, upon the elements of S where β is chosen so that our normalisation constraint $C_\alpha(X) \cdot n = -1$ still holds:

$$\begin{aligned} &\{C_\alpha(X) : X \in S\} \\ &\equiv \left\{ \frac{1}{1 - \alpha \cos\theta} \left(\frac{n}{2} + c_\theta e_1 + s_\theta c_\phi e_2 + s_\theta c_\phi e_3 \right) - \frac{\bar{n}}{2} \,:\, \theta \in (0, 2\pi],\, \phi \in (0, \pi] \right\} \end{aligned}$$

The elements of this set are no-longer null-vectors (they have extra components parallel to n) but one may still apply the inverse mapping $F^{-1}(\cdot)$ and extract a spatial vector (i.e. one with no components parallel to e or $\bar{e}$). It is found [16, 6] that the spatial vectors extracted thus lie on the surface of a quadric with a rotational cross-section corresponding to a conic. The form of this conic cross-section is determined by the parameter α which can be interpreted as the eccentricity:

- $\alpha = 0$: The set of points lie on the unit sphere.
- $0 < \alpha < 1$: The set of points lie on an ellipsoid formed by rotating an ellipse in the e_{12} plane about e_1. The origin lies at one of the foci and α is the eccentricity.
- $\alpha = 1$: The set of points lie on a paraboloid specified by the points

$$\{x : 2x \cdot e_1 = [(x \cdot e_2)^2 + (x \cdot e_3)^2] - 1\}.$$

- $\alpha > 1$: The set of points lie on a two-sheeted hyperboloid formed by rotating an hyperbola in the e_{12} plane about e_1. The origin is once again one of the foci and the eccentricity is α.

We may therefore form a set of representations for points lying on these conics by simply applying appropriate translation, dilation and rotation rotors after the conic transform.

4.1 Properties of the Conic Transform

It can further be shown [6] that the conic transform preserves the outer product but not the inner (and hence geometric) products. That is to say $C_\alpha(X \wedge Y) = C_\alpha(X) \wedge C_\alpha(Y)$. It also preserves the 'special' vectors n and $\bar{n}$. From this we can easily show that $C_\alpha(\cdot)$ does not change the nature of 'flat' objects (i.e. planes and lines transform to other planes and lines) since

$$\begin{aligned} C_\alpha\left(X_1 \wedge X_2 \wedge \cdots \wedge X_i \wedge n\right) &= C_\alpha(X_1) \wedge C_\alpha(X_2) \wedge \cdots \wedge C_\alpha(X_i) \wedge C_\alpha(n) \\ &= C_\alpha(X_1) \wedge C_\alpha(X_2) \wedge \cdots \wedge C_\alpha(X_i) \wedge n. \end{aligned}$$

It is also easy to verify by direct substitution that the conic transform preserves direction of points from the origin, i.e. that

$$F^{-1}(C_\alpha(F(a_i e_i))) = \frac{a_i}{1 - \alpha a_1} e_i$$

where we have adopted the usual summation convention.

4.2 Intersections

We may now consider how to apply this transform to intersecting conics with lines. Suppose that we have found a rotor R and value of α such that the set of points represented by

$$\{RC_\alpha(X)\tilde{R} : X \in S\}$$

lie on the conic we wish to intersect. Suppose further that we wish to intersect it with the line represented by the trivector L (as in figure 5a). We shall denote the points of intersection A and B. Now consider the effect of applying $C_\alpha^{-1}(\tilde{R}XR)$ to each of our objects and points (where X is the appropriate object). The points on the conic are transformed to the unit sphere. The line, L, is transformed to $C_\alpha^{-1}(\tilde{R}LR)$ (figure 5) which is still a line since the conic transform maps lines to

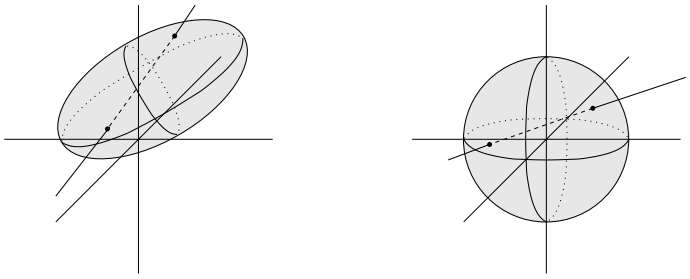

Fig. 5. Intersection of line L with (a) conic and (b) transformation back to intersection with unit sphere

lines. The points of intersection between this new line and our unit sphere may be found via the usual meet formulation:

$$A' \wedge B' = \Sigma \vee C_\alpha^{-1}(\tilde{R}LR)$$

where Σ is the multivector representing the unit sphere which may be formed as

$$\Sigma = F(e_1) \wedge F(e_2) \wedge F(e_3) \wedge F(-e_1)$$

or similar.

Finally we consider the transformed points $A = RC_\alpha(A')\tilde{R}$ and $B = RC_\alpha(B')\cdot$ $\tilde{R}$. They must lie upon our original line L since A' and B' clearly lie on $C_\alpha^{-1}(\tilde{R}LR)$ and

$$RC_\alpha(C_\alpha^{-1}(\tilde{R}LR))\tilde{R} = L.$$

They must also lie upon our conic since A' and B' lie on the unit sphere and the transformation is exactly that which generated our original conic. If they lie upon our conic and upon L they must therefore be the points of intersection. We have therefore formulated a method for intersecting lines and conics.

4.3 Reflections

We now consider the reflection of rays from conics. In order to reflect a ray we wish to find the tangent plane for a conic at the intersection point of the incoming ray with the conic. Again we make use of the fact that $C_\alpha(\cdot)$ maps planes to planes and apply $C_\alpha^{-1}(\tilde{R}XR)$ to all objects moving the points on the conic to the unit sphere. We then find the intersection point and tangent plane for the unit sphere. It can be shown [6] that when transformed back via $RC_\alpha(\cdot)\tilde{R}$ the tangent plane of the sphere maps to the tangent plane for the conic. Reflection can then be performed by simply reflecting the incoming ray L in this plane, Φ:

$$L' = \Phi L \Phi.$$

5 Line Images in a Para-Catadioptric Camera

As an illustration of the power of the techniques described above, let us consider the simple but useful question of what the image of a straight line in a parabolic mirror based Single View Point catadioptric (SVP Para-catadioptric) camera is. This problem has been considered in a number of papers e.g. [10, 27, 3, 4] and is useful for calibration and scene reconstruction.

The setup in question is illustrated in figure 6. With a parabolic mirror the SVP constraint requires that all the rays from the mirror imaged by the camera are parallel to the mirror axis and hence the camera is orthographic. Let us consider the source of light forming an image at the point $xe_2 + ye_3$. As we are imaging with an orthographic camera, for the setup shown, this ray is in the e_1 direction, which is axis of both the camera and the parabolic mirror. The focus of the parabolic mirror is situated at the origin.

$$S_1 = [e_{23} + (xe_3 - ye_2)n]^*$$

This result is easily established if we consider the form of the dual of a line [17].

$$L^* = \hat{m}I_3 + [(a \wedge \hat{m})I_3]n$$

with $\hat{m} = e_1$ (the ray's direction), $a = xe_2 + ye_3$ (a point on the ray) and $I_3 = e_{123}$. This form is analogous to writing the line in terms of Plücker coordinates where 3 of the coordinates give the line's direction and the other 3 give its moment about the origin.

We are interested in the reflection of this ray in the mirror. This reflected ray, S_2 is found as described in Section 4.3.

$$S_2 = \frac{1 - x^2 - y^2}{2} e_{145} - xe_{245} - ye_{345} \tag{13}$$

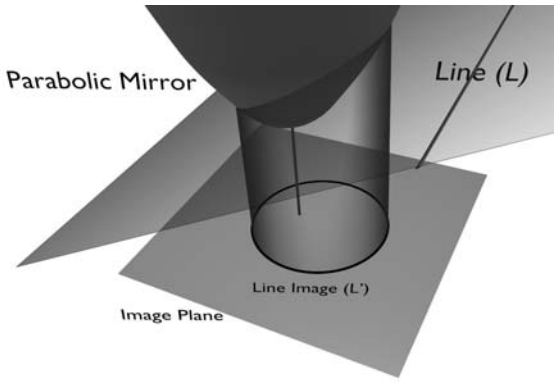

Fig. 6. Para-catadioptric Line Image Formation

For this ray to be the image ray of a point on a line L they must intersect and hence

$$L \vee S_2 = 0 \tag{14}$$

As stated above, a line, L, with unit direction vector $\hat{m} = m_1e_1 + m_2e_2 + m_3e_3$ (such that $\hat{m}^2 = 1$) passing through the point $a = a_1e_1 + a_2e_2 + a_3e_3$, can be expressed as

$$L = [\hat{m}I_3 + ((a \wedge \hat{m})I_3)n]^* \tag{15}$$

Substituting equations 15 and 13 into 14 and rearranging, the condition obtained is

$$\left(x + \frac{a_3m_1 - a_1m_3}{a_2m_3 - a_3m_2}\right)^2 + \left(y + \frac{a_1m_2 - a_2m_1}{a_2m_3 - a_3m_2}\right)^2 - \left(1 + \left(\frac{a_3m_1 - a_1m_3}{a_2m_3 - a3_m2}\right)^2 + \left(\frac{a_1m_2 - a_2m_1}{a_2m_3 - a_3m_2}\right)^2\right) = 0 \tag{16}$$

As others have observed [10], this is a circle (L' in figure 6). A similar derivation can be used to establish the image of a sphere in such a camera the only significant change being that the resulting condition is the product of two separate circle equations, indicating an ambiguity which can trivially be resolved.

6 Conclusions

In this paper we have developed and discussed a natural, extensible method for interpolating pose and position in both 3-dimensions and higher. The method allows for any traditional path interpolation method to be extended to encompass both pose and position whilst retaining its desirable properties. We further extend this method to deal with dilations. A method of handling conics within the CGA framework together with algorithms for ray-conic intersections and reflections was also discussed.

These algorithms could form a useful basis for many applications in computer graphics and computer vision. For example the use of jointed ellipsoidal models in marker-less motion capture requires computing the intersection of a ray with an ellipse which may be calculated using the method above. In the case of single viewpoint catadioptric cameras [1, 10] the reflection of rays from conics is often used and the method above may be used to form analytic solutions to a number of problems [6].

References

1. S. Baker and S. Nayar. A theory of single-viewpoint catadioptric image formation. *International Journal of Computer vision*, pages 1–22, 1999.
2. Sir R. Ball. *A Treatise on the Theory of Screws*. Cambridge University Press, 1900.

3. J. Barreto and H. Araujo. Paracatadioptric camera calibration using lines. In *Proceedings of ICCV*, pages 1359–1365, 2003.
4. E. Bayro-Corrochano and C. López-Franco. Omnidirectional vision: Unified model using conformal geometry. In *Proceedings of ECCV*, pages 536–548, 2004.
5. S. Buss and J. Fillmore. Spherical averages and applications to spherical splines and interpolation. *ACM Transactions on Graphics*, pages 95–126, 2001.
6. J. Cameron. Applications of Geometric Algebra, August 2004. PhD First Year Report, Cambridge University Engineering Department.
7. Z. Cendes and S. Wong. C1 quadratic interpolation over arbitrary point sets. *IEEE Computer Graphics and Applications*, pages 8–16, Nov 1987.
8. S. Cotin, H. Delingette, and N. Ayache. Real-time elastic deformations of soft tissues for surgery simulation. *IEEE Transactions on Visualization and Computer Graphics*, 5(1):62–73, 1999.
9. C. Doran and A. Lasenby. *Geometric Algebra for Physicists*. CUP, 2003.
10. C. Geyer and K. Daniilidis. Catadioptric projective geometry. *International Journal of Computer Vision*, pages 223–243, 2001.
11. V. Govindu. Lie-algebraic averaging for globally consistent motion estimation. In *Proceedings of CVPR*, pages 684–691, 2004.
12. D. Hestenes. *New Foundations for Classical Mechanics*. Reidel, Second Edition, 1999.
13. D. Hestenes. Old wine in new bottles: A new algebraic framework for computational geometry. In E. Bayro-Corrochano and G. Sobczyk, editors, *Geometric Algebra with Applications in Science and Engineering*, chapter 1. Birkhäuser, 2001.
14. D. Hestenes and G. Sobczyk. *Clifford Algebra to Geometric Calculus: A unified language for mathematics and physics*. Reidel, 1984.
15. S. Kshirsagar, S. Garchery, and N. Magnenat-Thalmann. Feature point based mesh deformation applied to mpeg-4 facial animation. In *Proceedings of the IFIP TC5/WG5.10 DEFORM'2000 Workshop and AVATARS'2000 Workshop on Deformable Avatars*, pages 24–34. Kluwer, 2001.
16. A. Lasenby. Recent applications of conformal geometric algebra. In *International Workshop on Geometric Invariance and Applications in Engineering*, Xi'an, China, May 2004.
17. J. Lasenby, A. Lasenby, and R. Wareham. A Covariant Approach to Geometry using Geometric Algebra. Technical Report CUED/F-INFENG/TR-483, Cambridge University Engineering Department, 2004.
18. H. Li, D. Hestenes, and A. Rockwood. Generalized homogeneous co-ordinates for computational geometry. In G. Sommer, editor, *Geometric Computing with Clifford Algebra*, pages 25–58. Springer, 2001.
19. M. Lillholm, E.B. Dam, and M. Koch. Quaternions, interpolation and animation. Technical Report DIKU-TR-98/5, University of Copenhagen, July 1998.
20. S. Mann and L. Dorst. Geometric algebra: A computational framework for geometrical applications (part 2). *IEEE Comput. Graph. Appl.*, 22(4):58–67, 2002.
21. M. Moakher. Means and averaging in the group of rotations. *SIAM Journal of Applied Matrix Analysis*, pages 1–16, 2002.
22. F. Park and B. Ravani. Smooth invariant interpolation of rotations. *ACM Transactions on Graphics*, pages 277–295, 1997.
23. K. Shoemake. Animating rotation with quaternion curves. *Computer Graphics*, 19(3):245–251, 1985.
24. G. Turk and M. Levoy. Zippered polygon meshes from range images. In *SIGGRAPH 1996 Proceedings*, pages 331–318, 1994.

25. R. Wareham and J. Lasenby. Rigid body pose and position interpolation using geometric algebra. *Submitted to ACM Transactions on Graphics*, September 2004.
26. R. Wareham, J. Lasenby, and A. Lasenby. Computer Graphics using Conformal Geometric Algebra. *Philosophical Transactions A of the Royal Society, special issue*. To appear soon.
27. X. Ying and Z. Hu. Spherical objects based motion estimation for catadioptric cameras. In *Proceedings of ICPR*, pages 231–234, 2004.