Chapter 2
Geometric Algebra for Modeling in Robot Physics

In this chapter, we discuss the advantages for geometric computing that geometric algebra offers for solving problems and developing algorithms in the fields of artificial intelligence, robotics, and intelligent machines acting within the perception and action cycle. We begin with a short tour of the history of mathematics to find the roots of the fundamental concepts of geometry and algebra.

2.1 The Roots of Geometry and Algebra

The lengthy and intricate road along the history of mathematics shows that the evolution through time of the two domains, *algebra* from the Arabic "alg-jbar" and geometry from the ancient Greek $\gamma\varepsilon\omega\mu\varepsilon\tau\rho\acute{\iota}\alpha$ ($geo = earth$, $metria = measure$), started to intermingle early, depending upon certain trends imposed by the different groups and schools of mathematical thought. It was only at the end of the nineteenth century that they become a sort of a clear, integrated mathematical system.

Broadly speaking, on the one hand, algebra is a branch of mathematics concerning the study of structure, relation, and quantity. In addition, algebra is not restricted to work with numbers, but it also covers the work involving symbols, variables, and set elements. Addition and multiplication are considered general operations, which, in a more general view, lead to mathematical structures such as groups, rings, and fields.

On the other hand, geometry is concerned with essential questions of size, shape, and relative positions of figures, and with properties of space. Geometry is one of the oldest sciences initially devoted to practical knowledge concerned with lengths, areas, and volumes. In the third century, Euclid put geometry in an axiomatic form, and Euclidean geometry was born. During the first half of the seventeenth century, René Descartes introduced coordinates, and the concurrent development of algebra evolved into a new stage of geometry, because figures such as plane curves could now be represented analytically with functions and equations. In the 1660s, Gottfried Leibniz and Isaac Newton, both inventors of infinitesimal calculus, pursued a geometric calculus for dealing with geometric objects rather than with sequences of numbers. The analysis of the intrinsic structure of geometric objects with the works of Euler and Gauss further enriched the topic of geometry and led to

E. Bayro-Corrochano, *Geometric Computing: For Wavelet Transforms, Robot Vision, Learning, Control and Action*, DOI 10.1007/978-1-84882-929-9_2,

the creation of topology and differential geometry. Since the nineteenth-century discovery of non-Euclidean geometry, the traditional concept of space has undergone a profound and radical transformation. Contemporary geometry postulates the concept of manifolds, spaces that are greatly more abstract than the classical Euclidean space and that approximately look alike at small scales. These spaces endowed an additional structure: a differentiable structure that allows one to do calculus; that is, a Riemannian metric allows us to measure angles and distances; symplectic manifolds serve as the phase spaces in the Hamiltonian formalism of classical mechanics, and the 4D Lorentzian manifolds serve as a space–time model in general relativity.

Algebraic geometry is considered a branch of the mathematics that combines techniques of abstract algebra with the language and problems of geometry. It plays a central role in modern mathematics and has multiple connections with a variety of fields: complex analysis, topology, and number theory. Algebraic geometry is fundamentally concerned with the study of algebraic varieties that are geometric manifestations of solutions of systems of polynomial equations. One looks for the Gröbner basis of an ideal in a polynomial ring over a field. The most studied classes of algebraic varieties are the *plane*, *algebraic curves* such as lines, parabolas, lemniscates, and Cassini ovals. Most of the developments of algebraic geometry in the twentieth century were within an abstract algebraic framework, studying the intrinsic properties of algebraic properties independent of a particular way of embedding the variety in a setting of a coordinated space as in topology and complex geometry. A key distinction between projective geometry and algebraic geometry is that the former is more concerned with the more geometric notion of the point, whereas the latter puts major emphasis on the more analytical concepts of a regular function and a regular map and extensively draws on sheaf theory.

In the field of mathematical physics, geometric algebra is a multilinear algebra described more technically as a Clifford algebra that includes the geometric product. As a result, the theory, axioms, and properties can be built up in a more intuitive and geometric way. Geometric algebra is a coordinate-free approach to geometry based on the algebras of Grassmann [74] and Clifford [42]. Since the 1960s, David Hestenes [86] has contributed to developing geometric algebra as a unifying language for mathematics and physics [87, 94]. Hestenes also presented a study of projective geometry using Clifford algebra [95] and, recently, the essential concepts of conformal geometric algebra [119]. Hestenes summarized and precisely defined the role of algebra and geometry in a profound comment emphasizing the role of the capacities of language and spatial perception of the human mind, which, in fact, is the goal of this section. We reproduce it to finalize this part as a prelude to the next section to motivate and justify why geometric algebra can be of great use to build the intelligent machine of which Turing dreamed.

> In his famous survey of mathematical ideas, F. Klein championed the fusing of arithmetic with geometry as a major unifying principle of mathematics. Klein's seminal analysis of the structure and history of mathematics brings to light two major processes by which mathematics grows and becomes organized. They may be aptly referred to as the algebraic and geometric. The one emphasizes algebraic structure, while the other emphasizes geometric interpretation. Klein's analysis shows one process alternatively dominating the other in the historical development of mathematics. But there is no necessary reason that

> the two processes should operate in mutual exclusion. Indeed, each process is undoubtedly grounded in one of two great capacities of the human mind: the capacity for language and the capacity for spatial perception. From the psychological point of view, then, the fusion of algebra with geometry is so fundamental that one could say, Geometry without algebra is dumb! Algebra without geometry is blind!
>
> —D. Hestenes, 1984

2.2 Geometric Algebra: A Unified Mathematical Language

First of all, let us analyze the problems of the community when they use classical mathematical systems to tackle problems in physics or robotics. In this regard, let us resort to an enlightening paper of Hestenes [92] where the author discusses the main issues for modeling physical reality. The invention of analytical geometry and calculus was essential for Newton to create classical mechanics. On the other hand, the invention of tensor analysis was essential for Einstein to create the theory of relativity. The point here is that without essential mathematical concepts, both theories would not have been developed at all. We can observe in some periods of the history of mathematics certain stagnation and from time to time, thanks to new mathematical developments, astonishing progress in diverse fields. Furthermore, we can notice that as researchers attempt to combine different mathematical systems, unavoidably this attitude in research leads to a fragmentation of knowledge. Each mathematical system brings about some parts of geometry; however, together they constitute a highly redundant system, that is, an unnecessary multiplicity of representations for geometric concepts; see Fig. 2.2. This approach in mathematical physics and in robotics has the following pressing defects:

- Restricted access: The ideas, concepts, methods, and results are unfortunately disseminated across the mathematical systems. Being proficient only in a few of the systems, one does not have access to knowledge formulated in terms of the other mathematical systems.
- Redundancy: Less efficiency due to the repetitive representation of information in different mathematical systems.
- Deficient integration/articulation: Incoherences, incongruences, lack of generalizations, and ineffective integration and articulation of the different mathematical systems.
- Hidden common knowledge: Intrinsic properties and relations represented in different symbolic systems are very difficult to handle.
- Low information density: The information of the problem in question is reduced due to distribution over different symbolic systems.

According to Hestenes [92], the development of a unified mathematical language is, in fact, a problem of the design of mathematical systems based on the following major considerations:

- Optimal encoding of the basic geometric intrinsic characteristics: dimension, magnitude, direction, sense, or orientation.

- Coordinate-free methods to formulate and solve problems in physics and robotics.
- Optimal uniformity of concepts and methods across different domains, so that the intrinsic common structures are made as explicit as possible.
- Smooth articulation between different alternative systems in order to access and transfer information frictionlessly.
- Optimal computational efficiency: The computing programs using the new system should be as or more efficient than any alternative system in challenging applications.

Note that geometric algebra was constructed following these considerations, and in view of the progress of scientific theory, geometric algebra helps greatly to optimize expressions of the key ideas and consequences of the theory.

2.3 What Does Geometric Algebra Offer for Geometric Computing?

Next, we will describe the most remarkable features of geometric algebra from the perspective of geometric computing for perception action systems.

2.3.1 Coordinate-Free Mathematical System

In geometric algebra, one writes coordinate-free expressions to capture concepts and constraints in a sort of high-level geometric reasoning approach. It is expected that the geometric information encoded in the expressions involving geometric products and the actions of operators should not suffer from interference from the reference coordinate frame. As a matter of fact, the results obtained by algebraic computing based on coordinates are geometrically meaningless or difficult to interpret geometrically. This is essentially because they are neither invariant nor covariant under the action of coordinate transformations. In fact, geometric algebra enables us to express fundamental robotics physics in a language that is free from coordinates or indices. The geometric algebra framework gives many equations a degree of clarity that is definitively lost in matrix algebra or tensor algebra.

The introduction of coordinates by R. Descartes (1596–1650) started the algebraization of geometry. This step in geometry caused a big change from a qualitative description to a qualitative analysis. In fact, coordinates are sequences of numbers and do not have geometric meaning themselves. G. Leibniz (1646–1716) dreamed of a geometric calculus system that deals directly with geometric objects rather than with sequences of numbers. More precisely, in a mathematical system, an element of an expression should have a clear meaning of being a geometric object or a transformation operator for algebraic manipulations such as addition, subtraction, multiplication, and division.

We can illustrate the concept of independence under the action of coordinate transformations by analyzing in the Euclidean plane geometry the following operation with complex numbers $\mathbf{a}, \mathbf{b} \in \mathbb{C}$: $\bar{\mathbf{a}}\mathbf{b} := (a_1, a_2)(b_1, b_2) = (a_1 b_1 - a_2 b_2, a_1 b_2 + a_2 b_1)$. This product is not invariant under the Euclidean group; that is, the geometric information encoded in the results of the product cannot be separated from the interference of the reference coordinate frame. However, if we change the complex product to the following product: $\bar{\mathbf{a}}\mathbf{b} := (a_1, -a_2)(b_1, b_2) = (a_1 b_1 + a_2 b_2, a_1 b_2 - a_2 b_1)$, then under any rotation $r : \mathbf{a} \to \mathbf{a}e^{i\theta}$ centered at the origin, this product remains invariant: $r(\bar{\mathbf{a}})r(\mathbf{b}) := (\mathbf{a}\bar{e^{i\theta}})(\mathbf{b}e^{i\theta}) = \bar{\mathbf{a}}\mathbf{b}$. Consequently, if the complex numbers are equipped with a scalar multiplication, addition, subtraction, and a geometric product, they turn from a field to a 2D geometric algebra of the 2D orthogonal geometry. It is clear that increasing the dimension of the geometric space and the generalization of the transformation group, the desired invariance will be increasingly difficult. Leibniz's dream is fulfilled for the nD classical geometries using the framework of geometric algebras. In this book, we present the following coordinate-free geometric algebra frameworks: for 2D and 3D spaces with a Euclidean metric, for 4D spaces with a non-Euclidean metric, and for $\mathbb{R}^{n+1,1}$ spaces the conformal geometric algebras.

Assuming that we are handling expressions as independently as possible upon a specific coordinate system, in view of the implementation, one converts these expressions into low-level, coordinate-based ones that can be directly executed by a fast processor. In general, geometric algebra can be seen as a geometric inference engine of an automated code generator, which is able to take a high-level specification of a physical problem and automatically generate an efficient and executable implementation.

2.3.2 Models for Euclidean and Pseudo-Euclidean Geometry

When we are dealing with problems in robotics or neural computing, an important question is in which metric space we should work. In this book, we are basically concerned with three well-understood space models:

(i) Models for 2D and 3D spaces with a Euclidean metric: 2D and 3D are well suited to handle the algebra of directions in the plane and 3D physical space. 3D rotations are represented using rotors (isomorph to quaternions). You can model the kinematics of points, lines, and planes using $\mathcal{G}_{3,0,0}$. Rotors can be used for interpolation in graphics and estimation of rotations of rigid bodies. Chapter 3 offers three increasingly powerful models of Euclidean geometry.

(ii) Models for 4D spaces with a non-Euclidean metric: If you are interested in linearizing a rigid motion transformation, you will need a homogeneous representation. For that we should use a geometric algebra for the 4D space. Here it is more convenient to choose the motor algebra $\mathcal{G}^+_{3,0,1}$ described in Chap. 3. It is the algebra of Plücker lines, which can be used to model the kinematics of points, lines, and planes better than with $\mathcal{G}_3$. Lines belong to the nonsingular

study 6D quadric, and the motors to the 8D Klein quadric. In $\mathcal{G}^{+}_{3,0,1}$, you can formulate a motor-based equation of motion for constant velocity where in the exponent you use a bivector for twists. You can also use motors for the interpolation of 3D rigid motion and estimate trajectories using EKF techniques. When you are dealing with problems of projective geometry, like in computer vision, again you need a homogeneous coordinate representation, so that the image plane becomes $\mathbb{P}^2$ and the visual space $\mathbb{P}^3$. To handle the so-called n-view geometry [84] based on tensor calculus and invariant theory, you require $\mathcal{G}_{3,1}$ (Minkownski metric) for the visual space and $\mathcal{G}_{3,0,0}$ for the image plane. This is described in Chap. 9. Note that the intrinsic camera parameters are modeled with an affine transformation within geometric algebra as part of the projective mapping via a projective split between the projective space and the image plane. Incidence algebra, an algebra of oriented subspaces, can be used in $\mathcal{G}_{3,1}$ and $\mathcal{G}_{3,0,0}$ to treat problems involving geometric constraints and invariant theory.

(iii) Conformal models: If you consider conformal transformations (angle preserving), conformal geometric algebra in Chap. 6 offers a non-Euclidean geometric algebra, $\mathcal{G}_{n,1}$, that includes in its multivector basis the null vectors origin and point at infinity. As a computational framework, it utilizes the powerful horosphere (the meet between a hyperplane and the null cone). Even though the computational framework uses a nonlinear representation for the geometric entities, one can recover the Euclidean metric. The basic geometric entity is the sphere, and you can represent points, planes, lines, planes, circles, and spheres as vectors or in dual forms, the latter being useful to reduce the complexity of algebraic expressions. As you may have noticed, the above-presented geometric algebras can be used either for kinematics in robotics or for projective geometry in computer vision. Provided that you calibrate a digital camera, you can make use of the homogeneous models from conformal geometric algebra to handle problems of robotics and those of computer vision simultaneously, however, without the need to abandon the mathematical framework. Furthermore, incidence algebra of points, lines, planes, circles, and spheres can be used in this framework as well. The topic of omnidirectional vision exploits the model of an image projected on the sphere, whereas all problems such as rigid motion, depth, and invariant theory can also be handled using conformal geometric algebra (see Chap. 9).

2.3.3 Subspaces as Computing Elements

The wedge product of k basis vectors spans a new entity, the k-vector. A set of all k-vectors spans an oriented subspace, $\bigwedge^{k} V^n$. Thus, the entire geometric algebra, G_n, is given by

$$G_n = \bigwedge^{0} V^n \oplus \bigwedge^{1} V^n \oplus \bigwedge^{2} V^n \oplus \cdots \oplus \bigwedge^{k} V^n \oplus \cdots \oplus \bigwedge^{n} V^n.$$

Geometric algebra uses the subspace structure of the k-vector spaces to construct extended objects ranging from lines, planes, to hyperspheres. If we then represent physical objects in terms of these extended objects, we can model physical phenomena like relativistic particle motion or conformal mappings of the visual manifold into the neocortex using appropriate operators and blade transformations.

2.3.4 Representation of Orthogonal Transformations

Geometric algebra represents orthogonal transformations more efficiently than the orthogonal matrices by reducing the number of coefficients; think of the nine entries of a 3D rotation matrix and the four coefficients of a rotor. In geometric algebra, a versor product is defined as

$$\boldsymbol{O} \rightarrow \boldsymbol{V}\boldsymbol{O}\widetilde{\boldsymbol{V}},$$

where the versor acts on geometric objects of different grades, subspaces, and also on operators, quite a big difference from the matrices. The versor is applied sandwiching the object, because this is the result of successive reflections of the object with respect to hyperplanes. In geometric algebra, a physical object can be described by a *flag*, that is, a reference using points, lines, planes, circles, and spheres crossing the gravitational center of the object. By applying versors equivalent to an orthogonal transformation on the flag, the relation of the reference geometric entities of the flag will remain invariant, that is, topology-preserving group action.

2.3.5 Objects and Operators

A geometric object can be represented using multivector basis and wedge products, for example, a point, line, or a plane. These geometric entities can be transformed for rigid motion, dilation, or reflection with respect to a plane or a sphere. These transformations depend on the metric of the involved space. Thus, we can model the 3D kinematics of such entities in different computational models as in the 3D Euclidean geometric algebra $\mathcal{G}_{3,0,0}$, 4D motor algebra $\mathcal{G}^{+}_{3,0,1}$, or the conformal algebra $\mathcal{G}_{4,1}$. You can see that the used transformations as versors can be applied to an object regardless of the grade of its k-blades. For example, in Fig. 2.1, we can describe the arm with points, circles, and spheres, and screw lines for the revoluted or prismatic joints, and then using direct/inverse kinematics relate the geometric entities from the basis through the joints until the end effector. Here a pertinent question will be whether or not operators can be located in space like geometric objects. The answer is yes; we can attach to any position the rotors and translators. This is a big difference from matrices; in geometric algebra the operators or versors treated as geometric objects, however, have a functional characteristic as well. In geometric

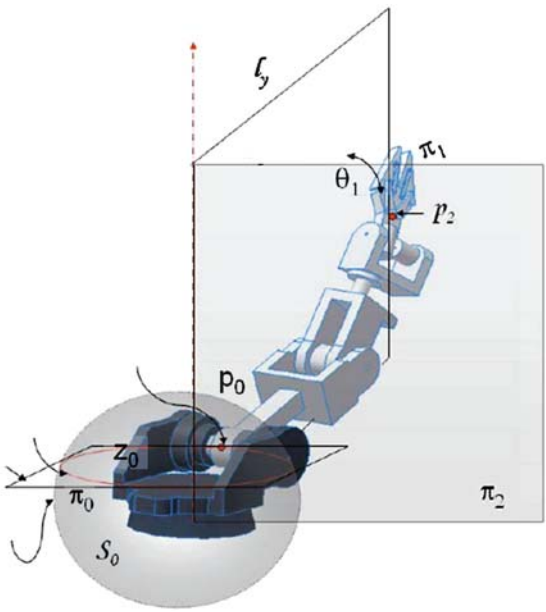

Fig. 2.1 Description of the kinematics of a 5-DOF robot arm using geometric entities of the conformal geometric algebra

algebra, objects are specified in terms of basic elements intrinsic to the problem, whereas the operators or versors are constructed depending upon which Lie group we want to use. A versor is built by successive reflections with respect to certain hyperplanes (lines, planes, spheres). In 3D space, an example of a versor is the rotor, which is built by two successive reflections with respect to two planes that intersect the origin.

A versor is applied sandwiching a geometric object. Since a versor represents a Lie group of the general linear groups, it can also be represented as an exponential form wherein the exponent in the Lie algebra space is spanned with a bivector basis (Lie generators). The versor and its exponential form are quite effectively represented using bivectors, which is indeed a less redundant representation than that by the matrices. Versor-based techniques can be applied in spaces of arbitrary signature and are particularly well suited for the formulation of Lorentz and conformal transformations.

In tasks of kinematics, dynamics, and modern control theory, we can exploit the Lie algebra representation acting on the bivectorial exponents rather than at the level of Lie group versor representation. In Chaps. 3 and 5, we describe bivector representations of Lie groups.

2.3.6 Extension of Linear Transformations

Linear transformations act on n-D vectors in $\mathbb{R}^n$. Since in geometric algebra a subspace is spanned by vectors, the action of the linear transformation on each

individual vector will directly affect the spanned subspace. Subspaces are spanned by wedge products; an outermorphism of a subspace equals the wedge products of the transformed vectors. The outermorphism preserves the grade of any k-blade it acts on; for example, the unit pseudoscalar must be mapped onto some multiple of itself; this multiple is the determinant of $\underline{f}(I) = \det(\underline{f})I$. We can proceed similarly when we compute the intersection of subspaces via the meet operation. We can transform linearly the result of the meet using duality and then apply an outermorphism. This would be exactly the same if we first transformed the subspaces and then computed the meet. In Chap. 9, we exploit outermorphisms for computing projective invariants in the projective space and image plane.

2.3.7 Signals and Wavelets in the Geometric Algebra Framework

One may wonder why we should interest ourselves in handling n-D signals and wavelets in the geometric algebra framework. The major motivation is that since in image processing, robotics, and control engineering, n-D signals or vectors are corrupted by noise, our filters and observers should smooth signals and extract features but do so as projections on subspaces of the geometric algebra. Thinking in quadrature filters, we immediately traduce this in quaternionic filters, or further we can generalize over non-Euclidean metrics the Dirac operator for multidimensional image processing. Thus, by weighting the bivectors with appropriate kernels like Gauss, Gabor, or wavelets, we can derive powerful Clifford transforms to analyze signals using the extended phase concept, and carry out convolutions in a certain geometric algebra or on the Riemann sphere. So we can postulate that complex filters can be extended over bivector algebras for computing with Clifford–Fourier transforms and Clifford wavelet transforms or even the space and time Fourier transform. In the geometric algebra framework, we gain a major insight and intuition for the geometric processing of noisy n-D signals. In a geometric algebra with a specific non-Euclidean metric, one can compute geometrically coupling, intrinsically different information, for example, simultaneously process color and thermal images with a multivector derivative or regularize color optical flow with the generalized Laplacian. Chapter 8 is devoted to studying a variety of Clifford–Fourier and Clifford wavelet transforms.

2.3.8 Kinematics and Dynamics

In the past, some researchers computed the direct and inverse kinematics of robot arms using matrix algebra and geometric entities like points and lines. In contrast, working in geometric algebra, the repertoire of geometric entities and the use of efficient representation of 3D rigid transformations make the computations easy and intuitive, particularly for finding geometric constraints. In conformal geometric

algebra, we can perform kinematic computations using meets of spheres, planes, and screw axes, so that the resulting pair of points yields a realistic solution to the problem. Robot object manipulation together with potential fields can be reformulated in conformal geometry using a language of spheres for planning, grasping, and manipulation.

The dynamics of a robot mechanism is normally computed using a Euler–Lagrange equation, where the inertial and Coriolis tensors depend on the degrees of freedom of the robot mechanism. Even though conformal geometry does not have versors for coding affine transformations, we can reformulate these equations, so that the entries of the tensors are projections of the centers of mass points of the limbs with respect to the screw–axes of the joints; as a result, we can avoid quadratic entries and facilitate the estimation of the tensor parameters. This is the benefit to handling this kind of a problem in either motor algebra or conformal algebra, making use of the algebra of subspaces and versors. Chapters 11 and 12 cover the computation of various problems of the kinematics and dynamics of robot mechanisms.

2.4 Solving Problems in Perception and Action Systems

In this section, we outline how we approach the modeling and design of algorithms to handle tasks of robotic systems acting within the perception–action cycle. In the previous section, we explained what geometric algebra offers as a mathematical system for geometric computing. Here we will be slightly more concrete and precise, illustrating the design and implementation of algorithms for real-time geometric computing.

Figure 2.2 shows an abstraction of the attitudes of many researchers and practitioners: How do they approach developing algorithms to solve problems in the domain of PAC systems? Briefly, they split the knowledge across various mathematical systems. As a consequence, as we discussed in Sect. 2.2, the ideas, concepts, methods, and results are unfortunately disseminated across various mathematical systems. Being proficient in only a few of the systems, one does not have access to knowledge formulated in terms of other mathematical systems. There is high redundancy due to the repetitive representation of information in different mathematical systems. A deficient articulation of the different mathematical systems degrades their efficiency. The intrinsic properties and relations represented in different symbolic systems are very difficult to handle. The information density is low, because the information from the problem is reduced due to distribution over different symbolic systems.

Bear in mind that geometric algebra was constructed for the optimal encoding of geometric intrinsic characteristics using coordinate-free methods. It ensures an optimal uniformity of concepts and methods across different domains, and it supports a smooth articulation between different alternative systems. The efficient representation of objects and operations guarantees computational efficiency. Since geometric

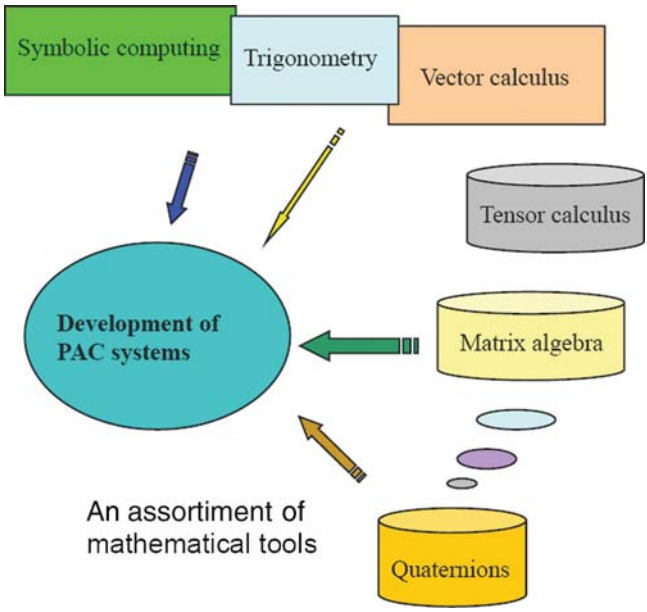

Fig. 2.2 Application of diverse mathematical systems to solve PAC problems

algebra was constructed following these considerations and in view of the progress of scientific theory, geometric algebra is the adequate framework to optimize expressions of the key ideas and consequences of the theory related to perception–action systems; see Fig. 2.3.

Of course it will not be possible for all PAC problems to be formulated and computed in geometric algebra. We have to ensure that the integration of techniques proceeds in a kind of top-down approach. First, we should get acquainted with the physics of the problem aided by the contributions of researchers, paying attention particularly to how they solve the problems. Recall the old Western interpretation of the saying *Nanos gigantum humeris insidentes*: one who develops future intellectual pursuits by understanding the research and works created by notable thinkers of the past.

For a start, we should identify where we can make use of geometric algebra. Since geometric algebra is a powerful language for efficient representations and finding geometric constraints, we should first, in a high-symbolic-level approach, postulate formulas (top-down reasoning), symbolically simplify them optimally, and execute them using cost-effective and fast hardware. To close the code generation loop, the conflicts and contradictions caused by our algorithms in their application are fed back in a bottom-up fashion to a negotiation stage in order to ultimately improve our geometric algorithms.

Let us now briefly illustrate this procedure. As a first example, we compute efficiently the inverse kinematics of a robot arm. Figure 2.4 shows the rotation planes

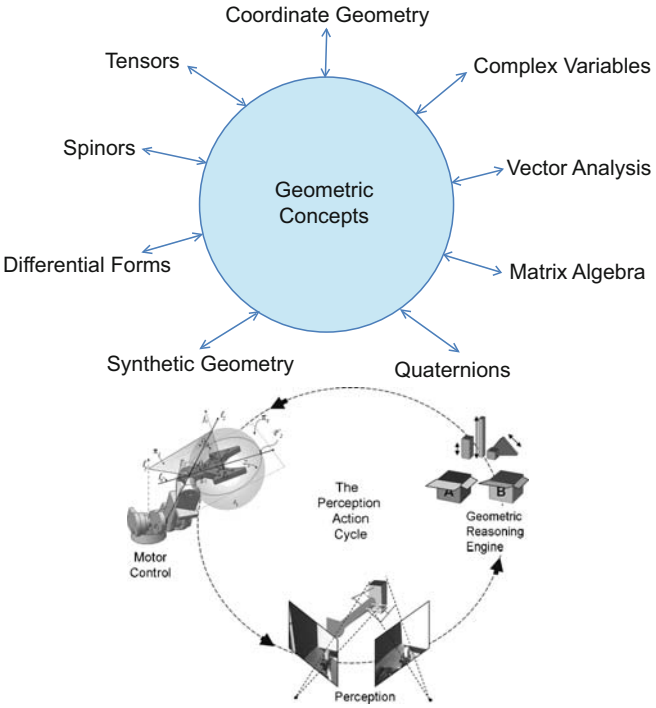

Fig. 2.3 Geometric algebra framework includes for the development of PAC systems and essential mathematical systems

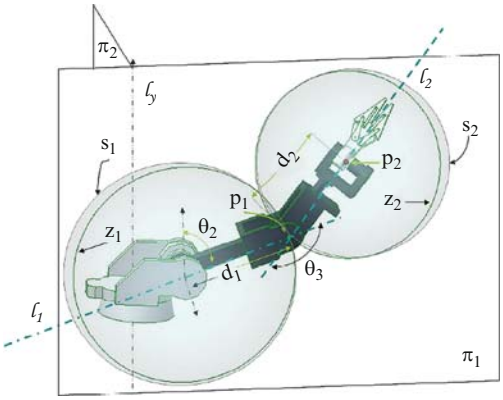

Fig. 2.4 Computing the inverse kinematics of a 5-DOF robot arm

of a robot arm. While computing its inverse kinematic, one considers a circle that is the intersection of two reference spheres $\boldsymbol{z} = \boldsymbol{s}_1 \wedge \boldsymbol{s}_2$. We then compute the pair of points ($\boldsymbol{PP}$) as the meet of the swivel plane and the circle: $\boldsymbol{PP} = \boldsymbol{z} \wedge \pi_{swivel}$, so that we can choose one as a realistic position of the elbow. This is a point that lies on the meet of two spheres. After the optimization of the whole equation of

the inverse kinematics, we get an efficient representation for computing this elbow position point and other parts of the robot arm, which can all be executed using fast hardware. Note that using a matrix formulation, it will not be possible to generate an optimal code.

A second example involves the use of the meet of ruled surfaces. Imagine that a laser welding device has to follow the intersection of two highly nonlinear surfaces. You can estimate these using a vision system and model them using the concept of ruled surface. In order to control the welding laser attached to the end-effector of the robot manipulator, we can follow the contour gained by computing the meet of the ruled surfaces, as depicted in Fig. 2.5.

A third example is to solve the sensor–body calibration problem depicted in Fig. 2.6. This problem can be simplified and linearized, exploding the intrinsic

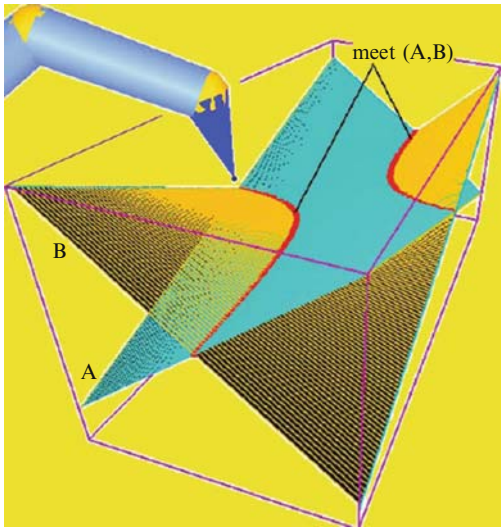

Fig. 2.5 Robot arm welding with laser along the meet of two ruled surfaces

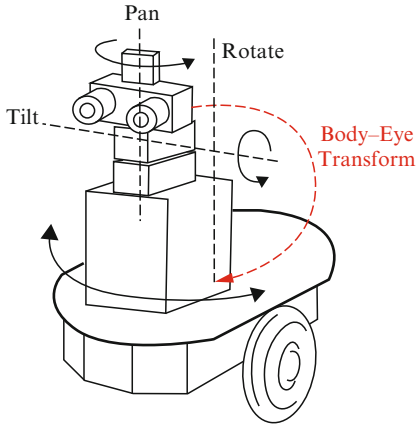

Fig. 2.6 Calibration of the coordinate system of a binocular head with respect to the robot coordinate system

relation of screw lines of the sensors and the robot body. The problem is reduced to finding only the unknown motors between these lines. Using matrices will yield a nonlinear problem of the kind $\mathbf{AX} = \mathbf{XB}$. As is evident, a line language of motor algebra suffices to tackle such a problem. A fourth problem entails representing 3D shapes in graphics engineering or medical image processing that traditionally should involve the standard method called *marching cubes*. However, we generalized this method as *marching spheres* [163] using the conformal geometric algebra framework. Instead of using points on simplexes, we use spheres of $\mathcal{G}_{4,1}$. In this way, not only do we give more expressive power to the algorithm but we also manage to reutilize the existing software by using, instead of the 2D or 3D vectors, the 4D and 5D vectors, which represent circles and spheres in conformal geometric algebra, respectively. In Fig. 2.7, see the impressive results of carving a 3D shape, where the spheres fill the gaps better than the cubes. As a fifth and last example, let us move to geometric computing. Traditionally, neural networks have been vector-based approaches with the burden of applying a coordinate-dependent algorithm for adjusting the weights between neuron layers. In this application, we have to render a contour of a certain noisy shape like a brain tumor. We use a self-organizing neural network called the neural gas [63]. Instead of adjusting the weights of the neurons to locate them along the contour or shape, we adjust the exponents of motors. In this way, we are operating in the linear space of the bivector or Lie algebra. The gain is twofold. On the one hand, due to the properties of the tangential Lie algebra space, the approach is linear. On the other hand, we exploit the coordinate-free advantage by working on the Lie algebra manifold using bivectors. Figure 2.8 shows the excellent results from segmenting a section of a brain tumor.

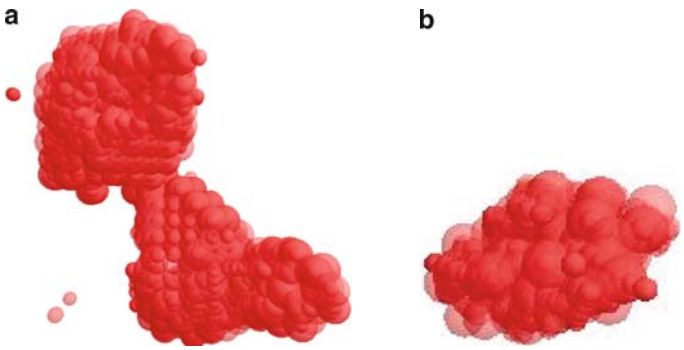

Fig. 2.7 Approximation of shape of three-dimensional objects with marching spheres. (**a**) Approximation of brain structure extracted from CT images (synthetic data); (**b**) approximation of a tumor extracted from real patient data

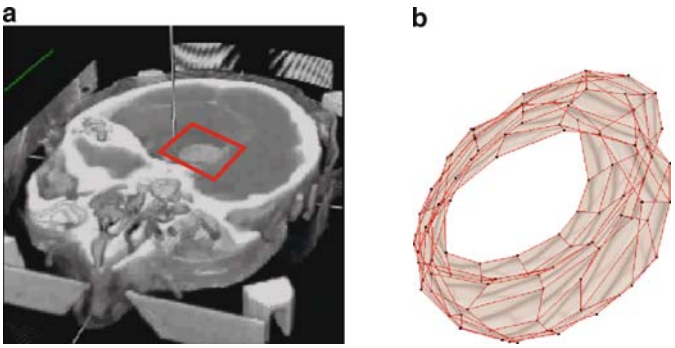

Fig. 2.8 Algorithm for a 3D object's shape determination. (**a**) 3D model of the patient's head containing a section of the tumor in the marked region; (**b**) final shape after training is completed with a total of 170 versors M (associated with 170 neural units)

Finally, in Parts V and VI, the reader will find plenty of illustrations using real images and robots where we have used, in a smart and creative manner, the geometric algebra language. We do hope to encourage readers to use this powerful and promising framework to design new real-time algorithms for perception and action systems.