

Hexapod Kinematics with Geometric Algebra: A Practical Refactor

Prepared for Bjørn Remseth

September 15, 2025

Acknowledgment: This work is strongly inspired by and builds upon the excellent foundation provided by Hans Jørgen Grimstad’s `goik` package ¹, which pioneered practical hexapod kinematics implementations in Go.

1 Why Geometric Algebra (GA)?

Hexapod math based on per-joint rotation matrices and translations quickly accumulates frame bookkeeping. *Projective Geometric Algebra* (PGA) and *Conformal Geometric Algebra* (CGA) [1] provide unified representations (motors/rotors) that act by a single sandwich product, reducing complexity and improving robustness.

PGA for SE(3). Rigid motions (rotations + translations) are represented by *motors* [2], which you can view as the geometric-algebraic analog of dual quaternions. Composition is associative; points are updated via $X' = MX\tilde{M}$.

CGA for incidence/distance. CGA encodes points as null vectors and treats planes, lines, circles, and spheres as native objects [1]. Translations are rotors in CGA, and meet/join operations give concise formulas for projections, distances, and constraints.

2 Representations

2.1 PGA (practical)

We implement motors as dual quaternions $(r + \varepsilon d)$ with unit r [3]. Given axis-angle $(\hat{\mathbf{u}}, \theta)$ and translation \mathbf{t} :

$$r = \cos \frac{\theta}{2} + \sin \frac{\theta}{2} (\hat{u}_x \mathbf{i} + \hat{u}_y \mathbf{j} + \hat{u}_z \mathbf{k}), \quad d = \frac{1}{2} \mathbf{t} r.$$

Composition is $(r_1, d_1)(r_2, d_2) = (r_1 r_2, r_1 d_2 + d_1 r_2)$. A point \mathbf{p} transforms as

$$\mathbf{p}' = r \mathbf{p} r^* + 2 \text{vec}(d r^*).$$

This matches PGA motor action for SE(3).

¹<https://github.com/hansj66/goik/tree/main/goik>

2.2 CGA (parallel sketch)

CGA lifts \mathbb{R}^3 into a 5D space with two null directions e_0, e_∞ [1]. Points, planes, spheres, and circles become algebraic blades. Rotors (including translations) act by $X' = RX\tilde{R}$. We provide an API mirroring the PGA motor API; the current code routes to the PGA backend so you can wire usage now and swap later.

3 Forward and Inverse Kinematics

Forward Kinematics (FK). Define per-joint screw motions as motors and compose:

$$P_{\text{toe}} = M_B M_{\text{hip}} R_1 M_{\text{thigh}} R_2 M_{\text{knee}} R_3 P_0 \widetilde{(\cdot)}.$$

Our Go code builds these via `Screw` and `Translator`.

Analytic IK (per leg). Use line/point relations to recover joint angles: for a hip yaw, take the angle between the hip axis and the projected toe direction; for the planar thigh-knee pair, use cosine law on the triangle defined by link lengths and the hip→toe distance.

Numerical IK. Build Jacobian columns from twists. For a revolute joint about axis $(\mathbf{c}, \hat{\mathbf{u}})$, the instantaneous linear velocity of a point is $\hat{\mathbf{u}} \times (\mathbf{p} - \mathbf{c})$. Stack columns and solve $\Delta\theta = J^\dagger(\mathbf{x}^* - \mathbf{x})$ with damping.

4 Contact and Constraints

With CGA, the ground plane Π and toe point P satisfy $\Pi \wedge P = 0$ at contact. Projection is a meet/join expression. In the current package, you can emulate with vector math or upgrade the `cga` package later.

5 Migration Notes

Replace sequences of rotation matrices + translations with motors. Precompute static offsets as motors. FK becomes a product; IK uses either the analytic triangle plus axis-angle recovery or a Jacobian solver.

6 Complexity and Robustness

GA reduces trig and frame conversions, unifies operations, and offers a clear path to autodiff and MPC [4]. Dual-quaternion motors are numerically stable and cheap to normalize.

References

- [1] L. Dorst, D. Fontijne, and S. Mann, *Geometric Algebra for Computer Science: An Object-Oriented Approach to Geometry*. Morgan Kaufmann, 2009.
- [2] C. Gunn, “On the homogeneous model of euclidean geometry,” 2011. PGA overview and applications.

- [3] L. Kavan, S. Collins, J. Zara, and C. O’Sullivan, “Skinning with dual quaternions,” in *Proceedings of the 2008 ACM Symposium on Interactive 3D Graphics and Games*, 2008.
- [4] J. Lasenby and L. Dorst, *Geometric Algebra and its Application to Computer Graphics and Robotics*. Cambridge University Press, 2024.