Journal of Space Exploration, Propulsion and Aerospace Systems

PRIME OPEN ACCESS

Volume 1, Issue 1

Research Article

Date of Submission: 25 Mar, 2026
Date of Acceptance: 30 Apr, 2026
Date of Publication: 08 May, 2026

The R3 Spin Group Rotation Vectors in Three-Dimensional Space

James Buckeyne*

Independent Research, Fernandina Beach, United States

***Corresponding Author:** James Buckeyne, Independent Research, Fernandina Beach, United States.

Citation: Buckeyne, J. (2026). The R3 Spin Group Rotation Vectors in Three-Dimensional Space. *J Space Explor Propuls Aerosp Syst, 1*(1), 1-14.

Abstract

Three-dimensional rotations have three degrees of freedom, yet the standard representations used in computation often hide that simplicity behind sequential parameterizations, redundant constraints, or less direct geometric coordinates. This monograph develops a Rotation Vector framework in which a rotation is represented by a vector in R3 whose direction gives the axis and whose magnitude gives the angle.

The framework is constructive throughout. A Rotation Vector may be applied directly to points through Rodrigues' rotation formula, converted to a matrix when needed, or mapped to the unit quaternions through the exponential map. More importantly, rotations may be composed directly in Rotation Vector form, without treating quaternion or matrix conversion as the primary computational path. This yields a compact description of identity, inverse, relative rotation, sequential composition, and additive combination of simultaneous torques, while keeping the geometry visible.

The presentation also clarifies the relation of this framework to the classical theory of rotations. Rotation Vectors are used here as the natural pure-rotation logarithmic coordinates of Spin(3) and therefore sit naturally in the Lie-algebraic description of rigid rotation, while still remaining accessible as ordinary vectors. The result is not a rejection of matrices or quaternions, but a re-centering of rotation theory around a representation that matches the axis-angle content of the motion itself.

Introduction

• The Problem

Rotations in three-dimensional space are among the most common objects in geometry, mechanics, robotics, graphics, and simulation. A rigid orientation has three degrees of freedom: one may specify an axis and an angle, or equivalently a point in the rotation group. In practice, however, the most widely used representations often separate those degrees of freedom in ways that obscure their geometric meaning.

Euler-angle systems describe a rotation as a sequence of coordinate-axis turns. They are easy to name and easy to enter, but they are order-dependent and admit singular configurations in which the parameterization loses local rank. Rotation matrices avoid that singular parameterization and give a direct linear action on vectors, but they store nine numbers subject to orthogonality constraints in order to represent only three degrees of freedom. Unit quaternions provide an efficient and elegant group structure, but the representation is less immediate geometrically: it uses a normalization constraint, a half-angle encoding, and a double-cover identification that often makes the represented motion less transparent than the underlying axis-angle picture.

The central claim of this monograph is that a rotation can be treated directly as a vector quantity. Its direction gives the rotation axis, and its magnitude gives the rotation angle in radians. The identity rotation is therefore the zero vector. The aim is to show that the Rotation Vector can serve not merely as a descriptive convenience, but as the basis of a coherent computational and geometric framework.

• Motivation and Scope

This work did not begin as an abstract reformulation of Lie theory. It began as a practical dissatisfaction with the behavior and interpretation of common rotation machinery in software systems. Earlier notes in this project grew out of camera and transformation problems in interactive 3D systems, especially the mismatch between local geometric intention and the order-dependent behavior of yaw-pitch-roll style updates.

From that practical starting point, the project moved through matrix-based constructions, direct basis recovery, axis-angle updates, and comparisons with quaternion formulations. The present monograph extracts the durable mathematical core from those exploratory drafts and presents it in a form that can be read as a coherent reference.

• What this Monograph Does

The monograph first defines the Rotation Vector as an element of R3 whose norm is the angle and whose direction is the axis. From that definition follow the identity rotation, inverse rotations, direct action on points through Rodrigues' formula, and conversion to matrices and quaternions.

It then presents a direct composition law for rotations in axis-angle form, avoiding the standard detour of converting to quaternions, multiplying there, and converting back. It also distinguishes sequential composition from vector addition of simultaneous angular influences.

Finally, it relates the Rotation Vector framework to the classical structures surrounding rigid rotation: Rodrigues' formulas, the exponential map, the Lie algebra so(3), the double cover by Spin (3), and the practical distinction between intrinsic and extrinsic rotation.

• What this Monograph Does not Claim

This monograph does not claim that matrices or quaternions are obsolete, nor that standard rotation theory must be replaced. Matrices remain indispensable for linear action and composition in many pipelines. Quaternions remain an elegant and efficient realization of Spin (3). The claim is instead that the Rotation Vector deserves to be treated as a first-class representation rather than as a transient encoding between more established ones.

Some surrounding ideas suggested by the broader project remain exploratory. That is not a weakness so long as the boundaries are kept clear. The present work is concerned first with the core framework itself.

• Plan of the Monograph

The exposition begins with notation and basic objects, then develops the composition law. After that, the monograph places the framework historically and structurally, relates it to quaternions and Lie theory, addresses recovery and branch behavior, and ends with geometric interpretation. Practical notes and examples are collected in the appendices.

• Notation and Basic Objects

This section fixes the basic notation used throughout the monograph. The aim is modest: to make the later composition formulas readable without overloading the reader at the outset.

Symbol / term	Meaning
$\mathbb{R}^3$	Three-dimensional real vector space
$\mathbf{r}$	A Rotation Vector
$\|\mathbf{r}\|$	Magnitude of $\mathbf{r}$, interpreted as rotation angle
$\hat{\mathbf{r}}$	Unit direction of $\mathbf{r}$ when $\mathbf{r} \neq \mathbf{0}$
axis	Rotation axis given by $\mathbf{r}/\|\mathbf{r}\|$
angle	Rotation angle given by $\|\mathbf{r}\|$
$\mathbf{0}$	Identity rotation
$\mathbf{u} \cdot \mathbf{v}$	Dot product
$\mathbf{u} \times \mathbf{v}$	Cross product
$R_{\mathbf{r}}(\mathbf{x})$	Action of Rotation Vector $\mathbf{r}$ on point $\mathbf{x}$
$\exp(\mathbf{r})$	Quaternion image of $\mathbf{r}$ under the exponential map
$\log(Q)$	Rotation Vector coordinate associated with quaternion Q
principal representative	A chosen bounded representative of an orientation
non-principal representative	A representative retaining winding information

Notation Table

Vectors in $\mathbb{R}^3$ are written in boldface, for example $\mathbf{v} = (v_x, v_y, v_z)$. Their Euclidean norm is written $\|\mathbf{v}\|$, their dot product as $\mathbf{u} \cdot \mathbf{v}$, and their cross product as $\mathbf{u} \times \mathbf{v}$. A unit vector is marked with a hat, so that $\hat{\mathbf{v}} = \mathbf{v}/\|\mathbf{v}\|$ whenever $\mathbf{v} \neq \mathbf{0}$.

A Rotation Vector is a vector $\mathbf{r} \in \mathbb{R}^3$ whose direction is the rotation axis and whose magnitude is the rotation angle in radians

$$\hat{\mathbf{r}} = \frac{\mathbf{r}}{\|\mathbf{r}\|}, \qquad \theta = \|\mathbf{r}\|.$$

The identity rotation is therefore the zero vector 0. This is one of the conceptual advantages of the representation: the identity is not a special normalized object or a constrained matrix, but simply the origin of the space.

Two distinct uses of the same quantity must be kept separate from the beginning. First, a Rotation Vector may denote a realized orientation: the accumulated rotation from a chosen reference frame. Second, it may denote an angular increment or angular velocity to be integrated over time. The underlying vector form is the same in both cases, but the interpretation is different.

The Basic Vector Operations are Standard

$$\mathbf{u} + \mathbf{v} = (u_x + v_x, u_y + v_y, u_z + v_z),$$

$$\lambda\mathbf{v} = (\lambda v_x, \lambda v_y, \lambda v_z),$$

$$\mathbf{u} \cdot \mathbf{v} = u_x v_x + u_y v_y + u_z v_z,$$

$$\mathbf{u} \times \mathbf{v} = (u_y v_z - u_z v_y,\ u_z v_x - u_x v_z,\ u_x v_y - u_y v_x).$$

These formulas matter here because the entire framework is built from them. In particular, the dot product controls angular agreement between axes, while the cross product carries the oriented failure of commutativity that later appears in the composition law.

• Action on Points
A Rotation Vector acts on a point through Rodrigues' formula. If^a is a unit axis and *θ* an angle, then the rotated image of a point x is

$$R(\mathbf{x}; \hat{\mathbf{a}}, \theta) = \cos\theta\, \mathbf{x} + \sin\theta\, (\hat{\mathbf{a}} \times \mathbf{x}) + (1 - \cos\theta)(\hat{\mathbf{a}} \cdot \mathbf{x})\hat{\mathbf{a}}.$$

In Rotation Vector form, this becomes

$$R_{\mathbf{r}}(\mathbf{x}) = R\left(\mathbf{x}; \frac{\mathbf{r}}{\|\mathbf{r}\|}, \|\mathbf{r}\|\right),$$

with the identity case handled separately by $R_{\theta}(x) = x$.

Relation to Matrices and Quaternions

A Rotation Vector may be converted to a rotation matrix by applying Rodrigues' formula to the standard basis vectors. The resulting matrix is naturally read column-wise as the images of the coordinate axes, which connects the abstract formula to practical frame recovery without requiring code in the text.

The framework also connects cleanly to the unit quaternions through the exponential map

$$\exp(\mathbf{r}) = \left(\cos\frac{\|\mathbf{r}\|}{2},\ \sin\frac{\|\mathbf{r}\|}{2}\,\frac{\mathbf{r}}{\|\mathbf{r}\|}\right),$$

with inverse logarithm $\log(\varrho) = r$. In this way the Rotation Vector serves as a logarithmic coordinate on Spin(3) while still living in the familiar vector space R3.

Principal and Non-Principal Representatives

For many practical purposes one may reduce attention to a principal representative, but the framework does not require the angle to remain bounded. The same physical orientation may be represented with different winding counts. That distinction matters whenever accumulated rotation, winding number, or long-running integration is part of the problem.

When only orientation matters, a principal representative may be chosen for convenience. When accumulated turning matters, the full unbounded vector should be retained. The important point is not to confuse these two uses.

• Composition of Rotations
This section gives the central operation of the framework: how to combine two rotations directly in Rotation Vector form.

The first distinction is conceptual. There are two different ways in which rotational quantities combine, and they should not be confused.

Sequential composition means that one rotation is applied after another. This is the case for rigid hierarchies, articulated chains, body-frame updates, and accumulated orientation.

Sequential composition is not commutative. Order matters.

Simultaneous combination means that several angular influences act at the same instant, as in summed torques or angular velocities before integration. In that case ordinary vector addition is the correct operation. It is commutative because simultaneous influences have no order.

• The Problem of Direct Composition
Suppose one rotation is represented by axis A^ and angle *a*, and a second by axis B^ and angle *b*. The usual computational advice is to convert both to quaternions or matrices, compose them there, and convert back. The present framework instead treats direct axis-angle composition as a first-class operation.

The goal is therefore this given

(A^,a) and (B^ ,b),

compute the resulting rotation

(C^ ,c)

without taking quaternion or matrix form as the primary language of the section.

• Closed-Form Composition
Introduce the half-angle terms

$$S_+ = \sin\frac{a+b}{2}, \qquad C_+ = \cos\frac{a+b}{2},$$
$$S_- = \sin\frac{a-b}{2}, \qquad C_- = \cos\frac{a-b}{2}.$$

The resulting angle is then recovered from the scalar term

$$c = 2\arccos\left(\frac{(\hat{\mathbf{A}}\cdot\hat{\mathbf{B}})(C_+ - C_-) + C_- + C_+}{2}\right).$$

The resulting axis is built from the vector term

$$\mathbf{C} = (\hat{\mathbf{B}}\times\hat{\mathbf{A}})(C_- - C_+) + \hat{\mathbf{A}}(S_+ + S_-) + \hat{\mathbf{B}}(S_+ - S_-),$$

followed by normalization of C to obtain C^ . The resulting Rotation Vector is then

$$\mathbf{r} = c\,\hat{\mathbf{C}}.$$

This is the main constructive result of the framework: direct composition in axis-angle variables, stated cleanly and then interpreted.

• How to Read the Formula
The formula has a clear internal structure.

The dot product $\mathrm{A}\hat{}\cdot\mathrm{B}\hat{}$ controls angular agreement between the two axes. When the axes are aligned, the composition behaves like direct addition of angles. When they are anti-parallel, it behaves more like subtraction. For intermediate configurations, the dot product continuously interpolates between those regimes.

The cross-product $\hat{B} \times \hat{A}$ carries the non-commutative part of the composition. This is the oriented term that records the fact that rotating around one axis and then another is not the same as reversing the order. In the language of the later theory chapter, this is the visible remnant of the Lie bracket.

Singular Cases and Aligned-Axis Branches

The general formula should not be evaluated blindly in every case. Some configurations are not merely numerically delicate; they are geometrically simpler and should be recognized early.

The first early check is the dot product $\hat{A} \cdot \hat{B}$.

If $\hat{A} \cdot \hat{B}$ is near 1, then the two axes are effectively the same. In that case the composition reduces directly to addition along a common axis. The resulting angle is the sum of the two angles, and the resulting axis is that shared axis.

If $\hat{A} \cdot \hat{B}$ is near -1, then the two axes are effectively opposite. In that case the composition reduces to subtraction of angles along a common line, with the sign interpreted according to the chosen axis convention.

After those aligned-axis checks, the resulting angle may still lie near 0 or near 2π. These are the usual axis-angle boundary cases. When the result is near the identity, the axis is indeterminate and the correct result is simply the zero vector. When the result is near a full turn, normalization again becomes delicate because the axis is being recovered from a nearly vanishing denominator.

Intrinsic and Extrinsic Application

One of the strongest observations in the current monograph is that the same composition formula also encodes the distinction between intrinsic and extrinsic rotation.

Intrinsic rotation means that the applied axis is attached to the moving body frame. Extrinsic rotation means that the applied axis is fixed in the ambient frame. The two are therefore not merely different verbal descriptions of the same update. They correspond to genuinely different geometric interpretations of how the new turn is being applied.

Earlier in this chapter, that distinction appeared through the sign and orientation of the noncommutative term. It may also be described in a second way: as a change-of-frame relation. In operator language, passing between body-frame and world-frame descriptions is closely related to conjugation. Geometrically, one may think of this as rotating into a frame versus rotating out of a frame.

Those Are Three Views of the Same Issue

- order of composition,
- body frame versus world frame,
- conjugate change of frame.

The essential point is that intrinsic and extrinsic application differ because the rotation being applied is being expressed in different frames.

Sequential Composition Versus Addition

The section closes with a simple but important distinction.
If rotations are chained through a hierarchy, use composition.
If angular influences are simultaneous, add the vectors.

These are not competing conventions. They answer different physical and geometric questions.

Historical Background and Classical Formulas

The Rotation Vector framework does not arise in isolation. It belongs to a long mathematical lineage in which three-dimensional rotation is described by an axis together with an angle, and in which composition is understood through formulas that predate modern matrix and quaternion practice.

Rodrigues and Axis-Angle Rotation

The most direct classical expression of axis-angle rotation is Rodrigues' formula for the action of a rotation on a point. Given a unit axis and an angle, the rotated point is obtained by decomposing the original vector into components parallel and perpendicular to the axis, rotating the perpendicular part in its plane, and leaving the axial part unchanged.

Rodrigues' formula matters here for more than historical completeness. It shows that axisangle language is already operational: a Rotation Vector is enough to define the action of a rotation on space using only vector operations and trigonometric weights.

Direct Composition as a Classical Possibility

The same historical lineage also contains direct formulas for composing rotations expressed by axis and angle. These formulas are less common in modern computational practice, where the standard route is to convert to matrices or quaternions, compose there, and convert back if needed.

This monograph gives direct composition in axis-angle variables primary presentation. That is not a denial of quaternion or matrix methods. It is a claim about which language best preserves the geometry of the operation being performed.

The Logarithmic Viewpoint

A modern reader will naturally ask how the Rotation Vector framework relates to the usual Lie-theoretic description of rotations. The answer is straightforward: Rotation Vectors are logarithmic coordinates for the rotation group.

A unit quaternion may be written in exponential form, and the corresponding logarithm is a vector whose direction is the rotation axis and whose magnitude is the rotation angle. In this sense the Rotation Vector is the Lie-algebra coordinate associated with a finite rotation, while quaternion multiplication expresses the corresponding group operation in covered form.

This is all the background needed for the later sections. The main ingredients of the framework are classical in spirit, even when the presentation is not standard.

Relation to Spinors, Quaternions, and Logarithmic Coordinates

A reader familiar with modern rotation theory will often assume that the present framework is merely quaternion language written in different coordinates. That reaction is understandable, but it is incomplete.

The Rotation Vector framework is fully compatible with the spinor description of rotation. A Rotation Vector maps to a unit quaternion by the exponential map, and the quaternion acts as the corresponding element of Spin(3). In that sense, the framework sits naturally inside the spinor picture.

What does not follow is that the paper should treat the quaternion as the hidden primary object. The mathematical compatibility is real, but the editorial and geometric center of the monograph remains the axis-angle vector itself. The purpose of the framework is not just to parameterize quaternion multiplication. It is to keep the represented rotation legible as an axis and an angle while still admitting direct composition, inversion, and comparison.

Why the Framework is not Merely "Spinors in Disguise"

If one begins with the quaternion as the primary object, the half-angle encoding, sign ambiguity, and normalization constraint come first, and the axis-angle content must be recovered afterward. If one begins with the Rotation Vector, the axis-angle content is primary from the start, and the quaternion becomes a derived exponential image.

That difference of emphasis matters. Two formalisms may be mathematically equivalent while still differing in what they treat as native, what they make transparent, and what they force the reader to reconstruct.

On the phrase "log-quaternion"

The older drafts sometimes use the phrase "log-quaternion" for the Rotation Vector. That terminology captures something important: the vector acts like a logarithmic coordinate for the quaternionic representation. But the phrase can also mislead.

A computation may be informed by logarithmic coordinates without being nothing more than the Lie-group multiplication law written entirely in log space. In some derivations the scalar part is separated, simplified, or reconstructed alongside the vector part. In that case the procedure is related to quaternion structure, but not identical to a naive "take logs, multiply in log space, and invert" picture.

For that reason, this monograph should prefer the phrase "logarithmic coordinate" to "logquaternion." The latter may still appear when discussing historical drafts or comparisons, but it should not be left undefined.

Round-Tripping and Inverse Maps

The passage from Rotation Vector to quaternion is clean. Given a Rotation Vector r, the exponential map produces the corresponding unit quaternion. The more delicate issue is the inverse map: how one recovers the axis-angle vector from quaternion data.

The most reliable presentation here is to recover the angle magnitude from the real component using arccos, then recover the axis direction from the normalized imaginary component.

In Plain Terms
- the real part gives the half-angle through arccos,
- the imaginary direction gives the axis,
- the full Rotation Vector is then angle times axis,
- with the chosen branch convention stated explicitly.

Why Matrices are Different
The matrix representation stands somewhat apart from this discussion. A rotation matrix is an excellent action operator, but as an inverse source for Rotation Vector recovery it sees only the principal physical rotation.

Matrix recovery is therefore best described as a projection back to principal orientation, not as a full inverse of accumulated rotation. Quaternions preserve more structure than matrices, but they still require branch selection. Rotation Vectors, by contrast, may retain unbounded accumulated turning directly when that information matters.

Summary
The Rotation Vector formalism is compatible with spinors. It may be embedded in quaternion language by the exponential map, and its inverse relation to quaternions must be discussed with care because logarithms have branches. But the framework should not be dismissed as mere alternate notation. Its main point is representational: it keeps the axis-angle content native while still connecting cleanly to the standard group-theoretic machinery.

Lie Groups, Lie Algebras and Local Structure
The Rotation Vector framework becomes easier to place mathematically once its relation to Lie theory is stated directly. The group of spatial rotations is non-commutative. Infinitesimal rotations therefore do not combine like ordinary scalars, and the order of finite rotations matters. The Rotation Vector gives a concrete coordinate in which those facts remain geometrically visible.

The Local Linear Picture
At very small scales, rotations behave almost linearly. A sufficiently small Rotation Vector may be treated as an infinitesimal turn whose components describe angular change about the coordinate axes. In that regime, ordinary vector addition is the first approximation to composition.

This local linearity is one reason the Rotation Vector is such a natural object. Near the identity, the space of rotations looks like ordinary three-dimensional vector space, and the zero vector represents the identity rotation itself.

so(3) and the Cross Product
The Lie algebra so(3) may be identified with three-dimensional vectors through the skewsymmetric matrix correspondence. Under that identification, the Lie bracket is represented by the cross product.

This is the mathematical reason the cross-product term in the composition formula matters so much. It is not an arbitrary correction term and not a computational accident. It is the visible sign of the non-commutative algebra carried by finite rotations.

Exponential Map and Finite Rotation
A Rotation Vector may be interpreted as a logarithmic coordinate for a finite rotation. Exponentiating it gives the corresponding group element, whether one chooses to view that element as a unit quaternion, a rotation matrix, or another equivalent realization.

From this point of view, the Rotation Vector is not merely an informal axis-angle label. It is a coordinate on the group obtained from its tangent structure.

Why Composition is not Ordinary Addition
If the group were commutative, exponentials of generators would combine by direct addition. But spatial rotation is not commutative. As a result, finite composition cannot be reduced globally to simple vector addition.

The composition formula derived earlier is the exact nonlinear law. Near the identity it approaches addition, but away from the identity it must include the directional interaction between axes. The cross-product structure is the simplest visible expression of that interaction.

Local Frame Interpretation
The Lie-algebra viewpoint also clarifies why the framework is so natural for frame updates. A Rotation Vector may represent either a finite orientation relative to a reference frame or an infinitesimal update expressed in the current local frame. The distinction is interpretive, not structural: the same vector object participates in both uses.

Summary

The Rotation Vector is compatible with the Lie-group description of rotation. Its local linear behavior comes from tangent-space structure, its non-commutative composition is reflected by the cross-product bracket, and its finite realizations arise through the exponential map.

That is enough structure to justify the framework mathematically without overwhelming the constructive development of the monograph.

Recovery, Branches and Practical Computation

The earlier sections define the Rotation Vector, develop its composition law, and place it within the usual group-theoretic picture. This section addresses the inverse direction: how one recovers a Rotation Vector from other representations, what information is preserved or lost in the process, and where branch choices must be handled explicitly.

Recovery from Matrices

A rotation matrix determines a physical orientation, and from that orientation one may recover a principal axis-angle representative. In that limited sense matrix-to-Rotation-Vector conversion is straightforward.

The limitation is equally important. A matrix does not preserve winding history. It records the realized orientation, not the accumulated turning by which that orientation was reached. As a result, recovery from a matrix should be described as recovery of a principal representative, not as full recovery of an unbounded Rotation Vector.

Recovery from Quaternions

Quaternion recovery preserves more structure than matrix recovery, but it still requires a branch convention.

The forward map from Rotation Vector to quaternion is clean: exponentiate the vector and obtain the corresponding unit quaternion. The inverse map is where care is needed. One must recover both an angle and an axis from a quantity whose sign and half-angle structure introduce multiple equivalent descriptions of the same physical rotation.

Why Arcos is Preferable here

The reason to prefer a real-part arccos reconstruction in this monograph is not that other inverse formulas are impossible. It is that they can obscure the branch structure.

Arctan-style inversions may be valid in carefully controlled settings, but they can also make one half of the inverse map look artificially discontinuous because the half-angle information is being reassembled through a branch-sensitive quotient. Using arccos on the real component keeps the geometric meaning of the recovered angle visible.

Principal and Non-Principal Representatives

A single realized orientation may correspond to many Rotation Vectors differing by full-turn winding. This is not an oddity of the framework; it is part of its expressive power.

When the problem concerns only current orientation, one may reduce to a principal representative. When the problem concerns accumulated turning, rotational history, or longrunning integration, that reduction is no longer appropriate. In those cases the unbounded Rotation Vector should be retained.

Early Checks and Boundary Cases

The practical evaluation of composition and recovery should not wait until the end of a calculation to detect degenerate cases.

The first early check is axis alignment. If two input axes are nearly parallel, composition reduces to direct addition of angles on the shared axis. If they are nearly anti-parallel, composition reduces to a difference of angles on a common line, interpreted according to the chosen axis convention.

Only after those checks does one need the usual output-angle boundary handling. When the resulting angle is near zero, the correct result is the zero vector and the axis is immaterial. When the resulting angle is near a full turn, axis recovery again becomes delicate because a small denominator appears in normalized reconstruction.

Scope of the Main Discussion

The main discussion concerns what can be recovered from matrices, what can be recovered from quaternions, where branch choices occur, and why winding information matters. Detailed threshold rules, implementation tolerances, and numerical stitching policies are better left to an appendix unless they are needed to understand a mathematical claim.

Geometric Interpretation

The earlier sections develop the Rotation Vector as a formal object, a composition law, and a logarithmic coordinate

compatible with the usual group-theoretic descriptions. This section returns to the geometric reason the framework is useful: it keeps a rotation readable as a directed angular quantity.

Rotation as a Visible Geometric Object

A Rotation Vector does not merely stand in for a hidden operator. Its direction gives the axis of the rotation and its magnitude gives the angular amount. When one looks at a matrix, the represented rotation must usually be inferred from its action on basis vectors or from a derived axis-angle calculation. When one looks at a quaternion, the represented rotation must be unpacked from half-angle and sign conventions. A Rotation Vector presents the axis-angle content directly.

Local Frame Behaviour

Many practical problems are naturally stated as angular instructions: turn a little to the left in the current frame, pitch upward relative to the present body axis, compose a local roll with an already accumulated orientation, or compare two nearby orientations by their relative turn. The Rotation Vector lets such instructions be expressed in a language that stays close to that intention.

This matters most when order matters. A body-frame update is not the same as a world-frame update, and the difference is a real geometric distinction. Earlier sections showed that this appears in the orientation of the non-commutative term in composition. Here the same point may be stated more intuitively: the framework keeps local angular intent visible while still respecting the true geometry of rotation.

Relative Rotation and Continuity

Suppose two orientations are given. One often wants not merely to know both of them independently, but to know the rotation that carries one frame into the other. In the Rotation Vector framework, this is the relative Rotation Vector: the directed angular displacement needed to align one frame with the other.

This is a natural quantity for stabilization, interpolation, control correction, pose alignment, and tracking. In each case the useful quantity is not just the target pose, but the angular mismatch still remaining.

A final physical orientation may admit more than one representative in the covered Rotation Vector space, and the most useful target may be the representative closest to the current state rather than the principal one. That observation matters whenever continuity in the covered space is part of the problem.

Simultaneous Influence Versus Sequential Action

The framework also helps separate two situations that are often blurred in casual language. If one is describing multiple angular influences acting at the same instant, vector addition is the natural operation. If one is describing one realized rotation followed by another, composition is the correct operation. The Rotation Vector helps because both kinds of quantity live in a similar-looking vector language, which makes the contrast easier to state explicitly.

Practical Value

The framework does not eliminate all subtlety. Rotations remain non-commutative, branchsensitive on inversion, and globally nonlinear. Its value is that the main subtleties remain tied to visible geometric features: axis agreement appears in dot products, non-commutativity appears in cross products, winding appears in branch choice and magnitude, and frame dependence appears in order of composition.

Conclusion

This monograph has presented the Rotation Vector as a practical and mathematical representation of finite rotation in three dimensions.

The framework begins with a simple encoding: a vector in R3 whose direction gives the axis and whose magnitude gives the angle. From that starting point it develops direct action on points, direct composition in axis-angle form, inverse and relative constructions, and a natural relation to quaternions, matrices, and the Lie-theoretic structure of rotation.

Its main contribution is not a claim of exclusivity, but of perspective. Matrices are powerful action operators. Quaternions are an elegant covered group representation. The Rotation Vector deserves attention because it keeps the angular content of the motion explicit while still participating in the same underlying mathematics.
That explicitness is more than cosmetic. It helps separate sequential composition from simultaneous addition, clarifies the role of branch choice and winding, and keeps local frame updates and relative corrections close to the language in which they are naturally described.

Some related ideas suggested by the broader project remain open, especially where interpolation across the covered space and other extensions are concerned. Even so, the central framework already stands independently: a constructive, geometrically legible way to describe and combine three-dimensional rotations without surrendering either mathematical seriousness or practical use.

Appendix A.
Usage and Application Notes

This appendix gathers material that is useful in practice but not required for the central mathematical argument.

Local Frame Update

Consider an object with an already accumulated orientation r. Suppose one now wishes to apply a small local yaw to the object: not a world-axis turn, but a turn relative to the object's current frame.

The new local yaw is represented by a small Rotation Vector d whose axis is the object's current local up direction and whose magnitude is the desired small turning angle. The updated orientation is then obtained by composing d with r in the intrinsic order.

If the same small yaw were applied in the world frame instead, the order of composition would change and the resulting orientation would, in general, be different.

Relative Rotation Between Two Poses

Now suppose two orientations are given, r1 and r2. A common practical question is: what directed turn carries the first pose into the second?

In the Rotation Vector framework, this is the relative Rotation Vector. One seeks the finite angular displacement that, when composed with the first orientation in the appropriate order, produces the second.

This quantity is useful because it says more than "these poses differ." It says how they differ, in the same axis-angle language used everywhere else in the framework.

Interpolation, Branch Choice, and Slerp

The standard language for rotational interpolation is SLERP. In quaternion form, SLERP selects a shortest-path interpolation on the covered sphere once a branch has been chosen.

The Rotation Vector viewpoint adds two useful questions before interpolation begins. The first is branch selection: a final physical orientation may admit more than one representative in the covered Rotation Vector space, and the most useful target may be the one closest to the current state rather than the principal one.

The second is that the framework appears to support a distinct interpolation construction that is not SLERP. In the present formulation, the difference between the endpoint Rotation Vectors determines a rotation axis in the covering space. That axis is then extrinsically rotated into the source frame, and the interpolation angle is recovered from the cross-product relation of the endpoint vectors. This appears to provide a meaningful continuation outside the usual interpolation interval, where continuity in the covered space may matter more than reduction to principal-angle form.

SLERP remains the standard quaternion interpolation method on its usual interval. The Rotation Vector framework clarifies branch choice before interpolation, and it may also support a distinct interpolation method when extrapolation or continuity in the covered space matters.

Application Domains

The framework is especially natural in settings where rotations are manipulated as intentions, updates, or comparisons rather than only as finished action operators. These include camera and view control, rigid-body simulation, robotics and articulated kinematics, attitude tracking and stabilization, interpolation, and geometric software systems that must move repeatedly between intuition and computation.

Appendix B.
Useful Operations

This appendix gathers practical operations that are often needed in software but do not require construction of a full basis matrix. It also records the full rotation matrix and its inverse recovery.

Let the Rotation Vector be written as r, with

$$\hat{\mathbf{r}} = \frac{\mathbf{r}}{\|\mathbf{r}\|}, \qquad \theta = \|\mathbf{r}\|.$$

where^r = ($\hat{r}_x$, $\hat{r}_y$, $\hat{r}_z$) denotes the unit direction of r. The following shorthands appear throughout

$$s = \sin\theta, \qquad c = \cos\theta, \qquad m = 1 - \cos\theta.$$

The Full Rotation Matrix

The rotation matrix is obtained by applying Rodrigues' formula to each standard basis vector. In column-major form (columns are the images of the *x-*, *y-*, and *z*-axes)

$$M = \begin{bmatrix} c + m\hat{r}_x^2 & m\hat{r}_x\hat{r}_y - s\hat{r}_z & m\hat{r}_x\hat{r}_z + s\hat{r}_y \\ m\hat{r}_x\hat{r}_y + s\hat{r}_z & c + m\hat{r}_y^2 & m\hat{r}_y\hat{r}_z - s\hat{r}_x \\ m\hat{r}_x\hat{r}_z - s\hat{r}_y & m\hat{r}_y\hat{r}_z + s\hat{r}_x & c + m\hat{r}_z^2 \end{bmatrix}$$

The common subexpressions *s*, *c*, and *m* appear in every entry. The off-diagonal terms come in symmetric pairs that differ only by the sign of the *s* term — this is the cross-product contribution. Computing the full matrix is only slightly more work than computing two individual axes, because the subexpressions are shared.

Recovery of a Rotation Vector from a Matrix

Given a rotation matrix *M* with entries M_{ij} (column *i*, row *j*)

The angle is recovered from the trace

$$\theta = \arccos\left(\frac{M_{00} + M_{11} + M_{22} - 1}{2}\right).$$

The axis (unnormalized) is recovered from the skew-symmetric part

$$\mathbf{a} = \big(M_{12} - M_{21},\ M_{20} - M_{02},\ M_{01} - M_{10}\big).$$

The Rotation Vector is then $\mathrm{r} = \theta\mathrm{a}/\|\mathrm{a}\|$.

When $\theta \approx 0$, the skew-symmetric part vanishes and the axis is indeterminate — the result is 0. When $\theta \approx \pi$, the skew-symmetric part is again small; the axis must instead be recovered from the symmetric part (the eigenvector of eigenvalue +1).

As noted in Section 7.1, matrix recovery returns only a principal representative. Winding information is not preserved.

Recovering Individual Basis Axes

The columns of the rotation matrix are the current frame axes. When only one axis is needed, it may be computed directly without the full matrix.

The current right axis (image of (1,0,0));

$$\mathrm{Right}(\mathbf{r}) = \begin{pmatrix} c + m\hat{r}_x^2 \\ s\hat{r}_z + m\hat{r}_x\hat{r}_y \\ -s\hat{r}_y + m\hat{r}_x\hat{r}_z \end{pmatrix}.$$

The current up axis (image of $(0, 1, 0)$):

$$\mathrm{Up}(\mathbf{r}) = \begin{pmatrix} -s\hat{r}_z + m\hat{r}_y\hat{r}_x \\ c + m\hat{r}_y^2 \\ s\hat{r}_x + m\hat{r}_y\hat{r}_z \end{pmatrix}.$$

The current forward axis (image of $(0, 0, 1)$):

$$\mathrm{Forward}(\mathbf{r}) = \begin{pmatrix} s\hat{r}_y + m\hat{r}_z\hat{r}_x \\ -s\hat{r}_x + m\hat{r}_z\hat{r}_y \\ c + m\hat{r}_z^2 \end{pmatrix}.$$

These are the columns of *M*, written vertically for readability. Each shares the same *s*, *c*, *m* subexpressions.

Pure Rotation About a Chosen Axis

If one wants a pure turn about a chosen unit axis^u by angle ϕ, then the corresponding Rotation Vector is simply

$$d = \phi \hat{u}.$$

This is the most direct form of an axis-constrained update.

Rotation About the Current Local Axes

Once a local axis has been recovered, a turn about that axis may be expressed directly as a Rotation Vector and then composed with the current orientation.

A yaw about the current up axis is generated from

$$d_{yaw} = \phi \, \mathrm{Up}(r).$$

A pitch about the current right axis is generated from

$$d_{pitch} = \phi \, \mathrm{Right}(r).$$

A roll about the current forward axis is generated from

$$d_{roll} = \phi \, \mathrm{Forward}(r).$$

The resulting orientation is then obtained by intrinsic or extrinsic composition, depending on whether the update is expressed in the moving frame or the ambient frame. In the language of Intrinsic and extrinsic application, this is the distinction between body-frame and world-frame application.

Twist alignments on the spherical mapping use the yaw construction.

Why these Operations Matter

In many practical systems the needed operation is not "recover the whole orientation basis" but something narrower: recover the current up axis, apply a roll about the forward axis, compute a local correction about one constrained direction, or compare only one angular component of a broader pose update. In such cases the Rotation Vector often lets the operation be written directly, without detouring through a full matrix reconstruction.

Appendix C.
Inverse kinematics

This appendix develops inverse kinematics (IK) in Rotation Vector language. The core idea is that the same cross-product and dot-product operations used elsewhere in the framework also describe how joint corrections propagate through an articulated chain. The treatment here uses the same geometric vocabulary as collision response in a physics engine: moment arms, torque vectors, and iterative constraint solving.

The Problem

Forward kinematics composes a chain of joint rotations sequentially (From Closed-form Composition) and reports the end-effector pose. Inverse kinematics asks the reverse: given a desired end-effector position and orientation, what joint angles produce it?

For chains of more than two links in three dimensions, the problem generally has no closedform solution. The system may be underdetermined (more joint freedoms than constraints), overdetermined (fewer), or exactly determined but nonlinear. An iterative approach is natural.

The Error Rotation

Given a chain of joints with current Rotation Vectors $r_1, r_2, \ldots, r_n$, forward kinematics computes the current end-effector orientation r_{actual} by sequential composition. The target orientation is r_{target}.

The rotation that would carry the current orientation to the target is itself a Rotation Vector

$$\mathbf{r}_{\text{error}} = \mathbf{r}_{\text{target}} \circ \left(-\mathbf{r}_{\text{actual}}\right)$$

where $\circ$ denotes the composition of Closed-form Composition and $-r$ is the inverse rotation (same axis, negated angle). This error vector gives the axis and magnitude of what is missing. If it is zero, the chain already reaches the target orientation.

How Much Can Each Joint Help?

Each joint has an axis as seen from the end-effector frame. Call this^ji: the axis of joint i, rotated through all downstream compositions to express it at the end effector.

The Dot Product

$$\hat{\mathbf{j}}_i \cdot \hat{\mathbf{r}}_{\text{error}}$$

measures how well joint i is aligned with the error. When this is near ±1, the joint can absorb the correction directly by adjusting its angle — this is the aligned-axis case of Section 3.4, where composition reduces to addition. When this is near 0, the joint's axis is perpendicular to the error and turning it does not help with this particular error component.

This is the same geometric test used in collision response. When a force is applied at a contact point, the torque it produces is the cross product of the moment arm with the force. The component of that torque along the body's spin axis changes the spin rate; the perpendicular component changes the spin direction. Here, each joint is a moment arm for affecting the end effector, and the dot product tells you how much of the error each joint can reach.

The Correction Step
For Each Joint, Compute Its Projected Influence on The Error

- Get joint i's axis in end-effector coordinates: $\hat{j}_i$.
- Project the error onto this axis: $\delta_i = r_{error} \cdot \hat{j}_i$.
- Apply a fraction of this as a correction: $r_i \leftarrow r_i + \alpha\delta_i \hat{j}_i$, where $\alpha < 1$ is a damping factor.

Joints aligned with the error get large corrections. Perpendicular joints get none. The damping factor prevents overshoot: correcting one joint changes the frame that every downstream joint operates in, so a single pass cannot account for all the nonlinear interactions.

Why Iterate?

The composition formula (From Closed-form Composition) is nonlinear. Correcting joint 1 changes the frame that joint 2 operates in, which changes joint 2's effective axis at the end effector, which changes how joint 3's correction propagates. A single pass through the chain does not capture these interactions. But each pass reduces the error, and the corrections get smaller as the chain converges.

This is directly analogous to how physics engines iterate constraint solvers: apply corrections, recompute the state, apply smaller corrections, repeat until the residual is below threshold.

Position Constraints

The discussion above handles orientation. For position constraints — the end effector must reach a specific point in space — the error is a translation vector p_{error}, not a rotation vector.

Converting a position error to joint corrections requires knowing how each joint's rotation moves the endpoint. If joint i changes by a small rotation δr_i, the endpoint moves by approximately

$$\delta\mathbf{p} \approx \delta\mathbf{r}_i \times \mathbf{L}_i$$

where L_i is the vector from joint i to the end effector — the moment arm. This is the same cross product used in collision response to compute linear velocity at a contact from angular velocity. The joint's ability to correct a position error depends on the cross product of its rotation axis with the moment arm to the endpoint.

The position correction for joint i is therefore: find the small rotation that, crossed with the moment arm L_i, produces a displacement toward the target. Given a desired displacement δp and a moment arm L_i, the required angular correction is the rotation about the axis perpendicular to both, scaled by the distance.

Combined Iteration

For a chain with both position and orientation targets, each iteration step computes both errors (one rotation vector, one translation vector), projects both onto each joint's influence, and applies a combined damped correction. The six-component error (3 position + 3 orientation) is distributed across all joints based on each joint's geometric ability to affect both types of error.

What the Rotation Vector Framework Provides

The error between two orientations is a vector in R^3. It can be scaled, projected onto joint axes, split among joints, and compared in magnitude for convergence. The dot product with each joint axis immediately tells you which joints are useful for a given correction. The cross product converts between angular and linear quantities at a contact or joint. None of this requires converting between representations.

The aligned-axis shortcut (Section 3.4) is particularly relevant: when a joint's axis is nearly aligned with the error, the correction is a simple angle addition. When perpendicular, the full composition is needed. The dot product tells you which regime applies, and therefore where the cheap path is available.

Implementation, testing, and convergence analysis for multi-segment chains — including behavior near singularities where moment arms collapse — are developed in a separate companion document on demonstrations and worked examples.

Appendix D.
The Covering Structure

This appendix collects observations about the topology of the Rotation Vector space that are useful for understanding winding, shells, and the double cover, but are not required for the core composition framework.

The Signed Ball

The natural domain of Rotation Vectors is the signed ball $[-2\pi,2\pi)$: vectors of magnitude up to 2π, with the direction carrying the sign of the rotation. The composition formula returns an angle in $[0,2\pi)$ and an axis; the sign of the spin is encoded in the direction of the axis. When the rotation is negative, the axis flips, placing the Rotation Vector on the opposite side of the ball.

Two Identities
Within this Ball there are Two Identity Configurations

- $\|r\| = 0$ (the origin): quaternion +1, physical identity.
- $\|r\| = 2\pi$ (the boundary sphere): quaternion −1, also physical identity on SO(3), but spinor ally distinct.

The Double Cover

Every physical rotation in SO(3) other than identity has exactly two representatives in the ball $[-2\pi,2\pi)$:

- A Rotation Vector r with magnitude $\theta \in (0,2\pi)$.
- The vector $-\hat{\mathbf{r}}(2\pi - \theta)$: same physical rotation, opposite side of the ball.

- These two map to quaternions q and $-q$ respectively, which act identically on vectors. This is the standard double cover of SO(3) by Spin(3).

The Modular Wrap

When a rotation accumulates beyond 2π — for example to $2\pi + \epsilon$ — the Rotation Vector wraps: $2\pi + \epsilon$ becomes $-(2\pi - \epsilon)$, placing the vector on the opposite side of the ball near the boundary.

The trajectory through the space is

$$0 \xrightarrow{\text{outward}} 2\pi \xrightarrow{\text{wrap}} -2\pi \xrightarrow{\text{inward}} 0$$

This traversal covers a total angle of 4π but stays within the $[-2\pi,2\pi)$ ball. The quaternion goes $+1 \to -1 \to +1$. The return to the origin at 4π total angle is the 720° return: a full spinorial cycle.

Shells and Winding

For tracking accumulated rotation beyond one cycle, the full unbounded Rotation Vector is used. The structure is periodic in 2π shells, each shell spanning 2π of angle

- $[-2\pi,0)$: one shell
- $[0,2\pi)$: one shell
- $[2\pi,4\pi)$: one shell

Each shell independently covers SO(3) once. The quaternion sign alternates at each 2π boundary: $+1$ at $0,4\pi,8\pi,\ldots$ and -1 at $2\pi,6\pi,10\pi,\ldots$ The full spinorial period — returning to the same quaternion — is 4π, spanning two shells.

Additive Composition and Shell Jumping

Sequential composition (From Closed-form Composition) returns an angle in $[0,2\pi)$ via the arccos. It always projects the result into the principal shell. Shell information is lost unless tracked externally.

Additive composition (Section 3.6) is unconstrained vector addition. The result can have any magnitude, including magnitudes beyond 2π. Several simultaneous torques, roughly aligned and large enough, sum directly into a higher shell without passing through intermediate angles. The shell position then reflects the total angular impulse, not just the net orientation — a physically meaningful distinction related to stored rotational energy.