MATHEMATICAL FRAMEWORK FOR UNCERTAINTY PROPAGATION IN GEOMETRIC NETWORKS

Project Design and Interface Specifications

X.M. Christine Zhu
xzhu83@jhu.edu
Spring, 2026

Contents

1 System Overview and Design Philosophy

This document specifies the complete mathematical and software architecture for a simulation-based framework that propagates uncertainty through networks of rigid-body geometric relationships.

The system is designed to:

- Represent uncertain rigid transformations in $SE(3)$.
- Represent uncertain 3D points.
- Propagate uncertainty analytically through arbitrary geometric networks.
- Support closed-loop conditioning.
- Provide Monte Carlo validation capability.

This framework performs **uncertainty propagation only**. It does not solve optimization or estimation problems.

> **Principle 1.1.** All uncertainty shall be represented in a vector space (Lie algebra) while all geometry shall remain on the Lie group $SE(3)$.

2 Mathematical Foundations

2.1 Pose Perturbation Model

Each uncertain transformation is represented as:

$$F = \{F_{\text{nom}}, C\},$$

with perturbation:

$$\vec{\eta} = \begin{bmatrix} \vec{\alpha} \\ \vec{\epsilon} \end{bmatrix} \sim \mathcal{N}(0, C).$$

The true transformation is modeled using a left-multiplicative perturbation:

$$T = \exp(\vec{\eta}) F_{\text{nom}}.$$

> **Principle 2.1.** All perturbations are expressed in the parent coordinate frame.

3 Uncertain Transform Composition

3.1 First-Order Composition Rule

Theorem 3.1 (First-Order Composition)**.** Let

$$F_{ab} = \{F_{\text{nom},ab}, C_{ab}\}, \quad F_{bc} = \{F_{\text{nom},bc}, C_{bc}\}.$$

Then the composed uncertainty satisfies

$$\vec{\eta}_{ac} \approx \vec{\eta}_{ab} + \text{Ad}_{F_{\text{nom},ab}} \vec{\eta}_{bc},$$

and

$$C_{ac} \approx C_{ab} + \text{Ad}_{F_{\text{nom},ab}} C_{bc} \text{Ad}^T_{F_{\text{nom},ab}}.$$

3.2 Algorithm: Transform Composition

Algorithm 1: ComposeTransforms
Input: F_{ab}, F_{bc}
Output: F_{ac}

1. Compute $F_{\text{nom},ac} = F_{\text{nom},ab} F_{\text{nom},bc}$.
2. Compute adjoint $A = \text{Ad}_{F_{\text{nom},ab}}$.
3. Compute $C_{ac} = C_{ab} + A C_{bc} A^T$.
4. Symmetrize C_{ac}.
5. Return $(F_{\text{nom},ac}, C_{ac})$.

4 Transform Inversion

Proposition 4.1 (Covariance Under Inversion)**.** For $F_{ba} = F_{ab}^{-1}$,

$$C_{ba} = \text{Ad}_{F_{\text{nom},ba}} C_{ab} \text{Ad}^T_{F_{\text{nom},ba}}.$$

Algorithm 2: InvertTransform

1. Compute $F_{\text{nom},ba} = F^{-1}_{\text{nom},ab}$.
2. Compute $A = \text{Ad}_{F_{\text{nom},ba}}$.
3. Compute $C_{ba} = A C_{ab} A^T$.

4. Symmetrize.
5. Return result.

5 Geometric Network Propagation

The system is modeled as a directed graph:

$$\mathcal{G} = (\mathcal{V}, \mathcal{E})$$

5.1 Path Propagation

Theorem 5.1 (Path Covariance Propagation)**.** For path $v_0 \to \cdots \to v_N$,

$$C_{st} = \sum_{k=0}^{N-1} \mathrm{Ad}_{F_{\mathrm{nom},0k}} C_{v_k v_{k+1}} \mathrm{Ad}^T_{F_{\mathrm{nom},0k}}.$$

Algorithm 3: QueryTransform

1. Find path between nodes.
2. Initialize identity transform.
3. For each edge:
   - If forward, compose.
   - If backward, invert then compose.
4. Return accumulated result.

6 Uncertain Point Propagation

Given:

$$p' = Rp + t$$

Covariance:

$$C_{p'} = J_{\vec{\eta}} C_\eta J_{\vec{\eta}}^T + R C_p R^T.$$

Algorithm 4: TransformPoint

1. Extract R, t.

2. Compute $p' = Rp + t$.
3. Compute Jacobian $J_{\vec{\eta}}$.
4. Propagate covariance.
5. Return result.

7 Closed-Loop Conditioning

Theorem 7.1 (Linear-Gaussian Conditioning)**.** Let

$$r = J\vec{\eta} + \epsilon.$$

Then posterior covariance is

$$C_{\text{post}} = C - CJ^T(JCJ^T + C_m)^{-1}JC.$$

Algorithm 5: ConditionOnResidual

1. Compute $S = JCJ^T + C_m$.
2. Compute gain $K = CJ^T S^{-1}$.
3. Update $C_{\text{post}} = C - KJC$.
4. Symmetrize.
5. Return posterior covariance.

8 Monte Carlo Validation

Principle 8.1. Monte Carlo simulation is used to validate the first-order approximation.

Algorithm 6: MonteCarloValidation

1. Sample $\vec{\eta}_i \sim \mathcal{N}(0, C)$.
2. Compute $T_i = \exp(\vec{\eta}_i)F_{\text{nom}}$.
3. Repeat n times.
4. Compute empirical covariance.

5. Compare to analytic covariance.

9 Numerical Stability and PSD Enforcement

After every covariance update:

$$C \leftarrow \frac{1}{2}(C + C^T).$$

Optional PSD enforcement:

1. Eigen-decompose C.
2. Clamp negative eigenvalues.
3. Reconstruct matrix.

10 Software Architecture

10.1 Layered Structure

- Lie Algebra Utilities
- Uncertain Entities
- Graph Network
- Propagation Engine
- Conditioning Engine
- Validation Layer

Each layer is independently testable.

11 Complexity Analysis

$$\text{Path Query: } O(|V| + |E|)$$

$$\text{Propagation: } O(k)$$

$$\text{Monte Carlo: } O(nk)$$

12 Verification Plan

12.1 Unit Tests

- Adjoint correctness
- Inversion correctness
- Composition symmetry
- PSD preservation

12.2 Statistical Validation

Relative Frobenius error:

$$\frac{\|C_{\text{analytic}} - C_{\text{empirical}}\|_F}{\|C_{\text{analytic}}\|_F} < \epsilon$$

13 Extensibility

Future extensions:

- Time-dependent state propagation
- Correlated edges
- Information-form representation
- Non-Gaussian models
- AMBF visualization integration
- GUI-based network construction

14 Conclusion

This document defines the complete mathematical and software architecture for uncertainty propagation in geometric networks. The framework preserves rigid-body structure, ensures probabilistic consistency under Gaussian assumptions, and provides a modular implementation pathway suitable for research and teaching applications.