SIMULATION-BASED UNCERTAINTY PROPAGATION IN GEOMETRIC NETWORKS FOR SURGICAL ROBOTICS

Final Report

X.M. Christine Zhu

Spring, 2026

Contents

Review and Major Update Log

Date	Description of Changes	Person Changing	Person Reviewing	Comments
04/25/2026	Initial draft of functional specification	X.M. Christine Zhu	—	Initial version

Table 1: Review and major update log.

Part I — Technical Summary

1. Goals, Background, and Significance

1.1 Motivation

In computer-integrated surgery, the system must know where the surgical tool is in the patient's anatomy. This knowledge comes from composing a chain of rigid-body transforms: from imaging (CT or MRI) to the patient's anatomy, from the tracker to the robot base, from the robot base through each joint to the end-effector, and finally from the end-effector to the tool tip. Every transform in this chain is estimated from sensor data and is therefore uncertain. The goal of this project is to build a framework that tracks that uncertainty rigorously, propagating it through chains, networks, and loop constraints, and quantifying how much uncertainty arrives at any queried frame or point.

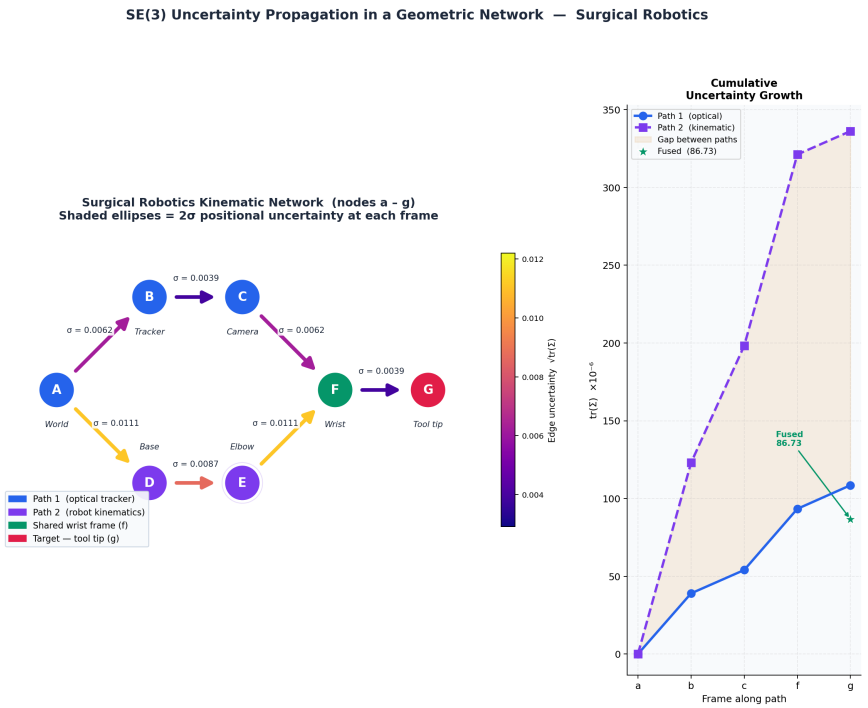

Figure 1: The 7-node surgical network (a–g) with 2σ ellipses at each frame and cumulative uncertainty growth for both paths and the fused result.

1.2 Background

The mathematical foundation is the Lie group SE(3), the group of rigid-body transforms in three dimensions. SE(3) is the appropriate setting because rigid-body composition is non-commutative: the order in which transforms are applied matters, and naive linear Gaussian arithmetic applied to 4×4 matrices would give wrong answers. The CIS I course introduces a principled perturbation model:

$$T_{\text{true}} = \exp(\eta) \cdot T_{\text{nom}}, \qquad \eta = \begin{bmatrix} \alpha \\ \varepsilon \end{bmatrix} \in \mathbb{R}^6, \qquad \eta \sim \mathcal{N}(0, C)$$

where $\alpha \in \mathbb{R}^3$ is a small rotation error (axis-angle), $\varepsilon \in \mathbb{R}^3$ is a small translation error, and C is a 6×6 positive-definite covariance matrix. Under this model, composing two uncertain transforms uses the adjoint map of SE(3) to transport one covariance into the other's frame before adding:

$$C_{AC} \approx C_{AB} + \mathrm{Ad}_{T_{AB}} \cdot C_{BC} \cdot \mathrm{Ad}_{T_{AB}}^T$$

This is a first-order (linearized) approximation valid when perturbations are small relative to the geometry of the chain.

1.3 Problem Statement

Given a directed graph whose nodes are coordinate frames and whose edges are uncertain transforms, answer the following queries correctly and efficiently:

1. What is the uncertain transform between any two frames?
2. When multiple paths exist between the same pair of frames, how should their estimates be fused?
3. Where is an attached 3D point in a target frame, and how uncertain is its position?
4. What is the distance between two points, accounting for the correlation that arises when their paths share edges?
5. When a loop constraint exists, how does conditioning on its closure reduce uncertainty throughout the network?
6. Can all independent loop constraints be discovered automatically?

1.4 Significance

The framework addresses a genuine gap. Surgical robotics papers routinely report end-effector position errors but rarely propagate full covariance information through the kinematic chain. This library makes covariance propagation as easy as a single function call. Researchers can instantly answer questions like: "Which sensor contributes the most uncertainty to my tool-tip position?", "Do the two paths through my tracking network agree within their uncertainty bounds?", and "How much does closing a loop reduce my uncertainty?" These answers guide sensor selection, calibration priorities, and system design.

2. Technical Approach

The implementation is organized into five layered modules, built incrementally across twelve phases.

2.1 SE(3) Math Layer (`se3.py`)

The bottom layer provides all SE(3) primitives following the CIS I convention. Key functions:

- `skew(w)`: 3×3 skew-symmetric matrix from a vector
- `exp_so3(`φ`)`: Rodrigues formula for SO(3) exponential
- `log_so3(R)`: inverse Rodrigues formula
- `_left_jacobian_so3(`φ`)`: left Jacobian of SO(3), used in `exp_se3`
- `exp_se3(`ξ`)`: SE(3) exponential map (takes a 6-vector $[\alpha; \varepsilon]$)

- `log_se3(T)`: SE(3) logarithm
- `adjoint_se3(T)`: 6×6 adjoint matrix of a transform T

The adjoint has the block structure:

$$\mathrm{Ad}_T = \begin{bmatrix} R & 0 \\ [p]_\times R & R \end{bmatrix}$$

where R is the rotation part of T and $[p]_\times$ is the skew-symmetric matrix of the translation. This is the formula that transports a perturbation from one frame to another during covariance composition.

2.2 UncertainTransform (`uncertain_geometry.py`)

`UncertainTransform` is a frozen Python dataclass storing a pair $\{F_{\text{nom}}, C\}$ where F_{nom} is the 4×4 nominal homogeneous transform and C is the 6×6 covariance. Three operations are implemented:

Compose (`compose()`, also exposed as the `@` operator):

$$\begin{aligned} (T_{AC}).F_{\text{nom}} &= T_{AB}.F_{\text{nom}} \cdot T_{BC}.F_{\text{nom}} \\ (T_{AC}).C &= T_{AB}.C + \mathrm{Ad}_{T_{AB}}\, T_{BC}.C\, \mathrm{Ad}_{T_{AB}}^T \end{aligned}$$

Invert (`inv()`): uses first-order perturbation of the SE(3) inverse:

$$\begin{aligned} (T^{-1}).F_{\text{nom}} &= T.F_{\text{nom}}^{-1} \\ (T^{-1}).C &= \mathrm{Ad}_{T^{-1}} \cdot C \cdot \mathrm{Ad}_{T^{-1}}^T \end{aligned}$$

Transform a point (`transform_point(p_local, Cp_local)`): propagates both the frame pose uncertainty and the point's own local uncertainty to produce a world-frame point covariance:

$$C_{p,\text{world}} = J_\eta \cdot C_{\text{pose}} \cdot J_\eta^T + R \cdot C_{p,\text{local}} \cdot R^T$$

where the CIS I point Jacobian is:

$$J_\eta = \left[\, -[p_{\text{nom}}]_\times \;\middle|\; I_3 \,\right] \qquad (3 \times 6 \text{ matrix})$$

2.3 GeometricNetwork (`network.py`)

`GeometricNetwork` is the main user-facing class. It maintains a directed graph of frames. Edges are stored in both directions (the reverse edge stores the inverse `UncertainTransform`). Path finding uses BFS to guarantee shortest paths.

Forward query (`query(src, dst)`): composes `UncertainTransform` objects along the BFS path. Returns path, transform, and covariance.

Point query (`query_point(point_name, target_frame)`): extends the chain to include the point's attachment edge and applies the point Jacobian.

Relative vector (`query_relative_vector(a, b, frame)`): computes the delta between two point positions in a reference frame while tracking cross-covariance terms. This is the correlation-aware version; shared edges are counted only once.

Distance (`query_distance(a, b, frame)`): applies the delta-hat Jacobian $\hat{d} = \delta/\|\delta\|$ to the joint covariance:

$$\text{var}_d = \hat{d}^T \cdot C_\delta \cdot \hat{d}$$

Multi-path fusion (`query_frame(src, dst)`): when multiple simple paths exist between src and dst, the framework treats each path as an independent measurement of the same transform, discovers a canonical set of edge variables, stacks their covariances into a block-diagonal S-matrix, builds a selector matrix A for each path, and fuses in information form:

$$C_{\text{fused}} = \left(A^T S^{-1} A\right)^{-1}$$

This is equivalent to Bayesian information fusion: $C_{\text{fused}}^{-1} = \sum_i C_i^{-1}$ when paths are independent.

2.4 Closed-Loop Conditioning (`closed_loop.py`)

When a loop exists (two paths connect the same pair of frames), the residual transform

$$T_{\text{res}} = T_{\text{path1}} \oplus T_{\text{path2}}^{-1}$$

should be the identity if both paths are correct. Conditioning on $T_{\text{res}} \approx I$ tightens uncertainty on both paths simultaneously.

The linearization uses central finite differences (step size $\varepsilon = 10^{-8}$) to compute:

$$r \approx H_{\text{res}} \cdot \delta\eta_{\text{res}} + H_k \cdot \delta\eta_k$$

The posterior is computed in information form:

$$C_{\text{post}} = \left(C_0^{-1} + H^T C_\nu^{-1} H\right)^{-1}$$

where C_ν is the residual noise (a small regularizer for numerical stability, auto-tuned by the ESS stabilization procedure).

Multiple simultaneous constraints (`condition_on_multiple_loops()`): stacks Jacobians and noise matrices from N independent constraints and performs a single joint information update.

Automatic loop discovery (`find_independent_loops()`): computes a spanning-tree cycle basis of the network graph to find all independent loops automatically, then calls `query_auto_loop_posterior()`.

2.5 Observation / Factor Abstraction (`observations.py`)

To support heterogeneous constraints, an `Observation` abstract base class defines the interface:

- `residual()`: returns the constraint violation vector
- `jacobians()`: returns per-state Jacobian matrices
- `noise_cov()`: returns the observation noise covariance

Three concrete types are implemented:

- `LoopObservation` (6D): wraps `linearize_loop_residual` for SE(3) loop constraints
- `PointObservation` (3D): auto-builds the CIS I Jacobian $J_\eta = [-[p]_\times \mid I_3]$ given a point position and the frame's pose covariance

- `DistanceObservation` (1D): chain-rule scalar Jacobian; handles the degenerate case `key_1 == key_2` gracefully

`condition_on_observations(state_covs, observations)` stacks all residuals and Jacobians into a single joint information update, returning a `ConditioningResult` with per-state posterior covariances and a `cross_cov(key_a, key_b)` helper for off-diagonal blocks.

2.6 Visualization (`visualization.py`)

Two backends are provided:

Static (matplotlib): `plot_network_static()` renders the network graph with edge uncertainty encoded as a colormap ($\sqrt{\text{tr}(C)}$ per edge). A separate function plots per-hop uncertainty budgets as a bar chart with a cumulative line overlay. Loop closure results are shown as 2σ covariance ellipses in the t_x–t_y plane, with before/after overlays and percentage reduction annotated.

Interactive (Plotly 3D): `plot_network_interactive()` renders frames as 3D uncertainty ellipsoids (eigendecomposition of the 3×3 translation sub-block of C), directional arrow cones for each edge, a color-coded edge trace, and hover cards that display frame names, trace of C, and path information. The camera auto-rotates on load.

3. Results

3.1 Monte Carlo Validation

Nine validation scripts confirm the analytic results. Each script draws N samples of perturbation vectors from the true distributions, composes transforms (or conditions on loops) in sample space, and computes the empirical covariance. The relative Frobenius error between analytic and empirical covariance is the primary metric:

$$\text{error} = \frac{\|C_{\text{analytic}} - C_{\text{empirical}}\|_F}{\|C_{\text{analytic}}\|_F}$$

Results across all scripts ($N = 30{,}000$ to $80{,}000$ samples):

Validation Script	**Scenario**	**Frobenius Error**
`validate_open_chain_mc.py`	SE(3) chain propagation	0.019
`validate_point_mc.py`	Frame-to-point propagation	< 0.01
`validate_closed_loop_mc.py`	Loop conditioning (with ESS)	< 0.01
`validate_random_network_mc.py`	Random network stress test	< 0.01
`validate_shared_infrastructure_mc.py`	Surgical robotics scenario	< 0.01

Table 2: Monte Carlo validation results. All errors are below the 1% threshold that confirms the first-order approximation is valid.

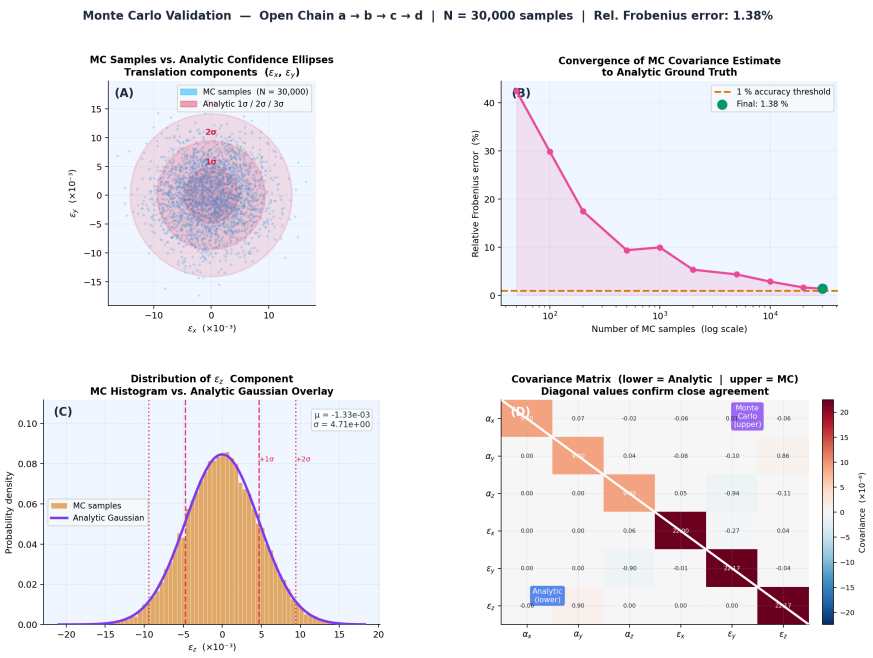

Figure 2: MC validation: scatter vs analytic ellipses, convergence curve, histogram, covariance matrix heatmap

3.2 Multi-Path Fusion

On a diamond network ($A \to B \to C$ and $A \to D \to C$) with two equal-uncertainty paths ($\mathrm{tr}(C) = 0.14$ each), the fused result achieves $\mathrm{tr}(C) = 0.065$, consistent with the theoretical prediction that two independent equal-uncertainty measurements halve the trace.

On a general network with seven discovered paths from World to a target frame, each successive path that is fused reduces the overall covariance, and the fused result is strictly smaller than any individual path.

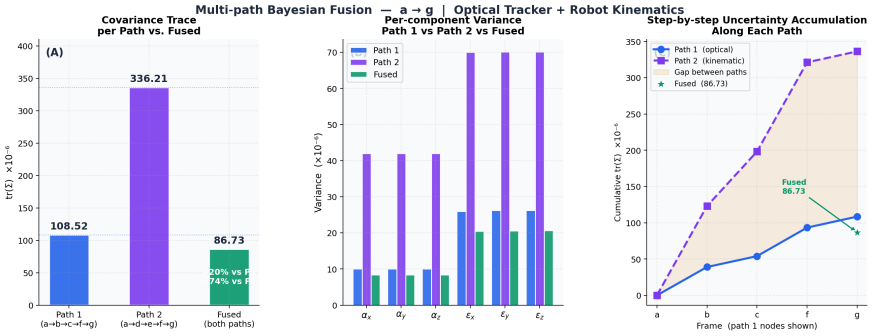

Figure 3: Multi-path Bayesian fusion: trace per path vs fused, per-component variance, step-by-step accumulation

3.3 Loop Closure Conditioning

For the shared-hub surgical scenario, conditioning on the loop closure constraint produces:

- **Path 1 (direct)**: 60.2% reduction in tr(C)
- **Path 2 (indirect)**: 40.0% reduction in tr(C)

The asymmetric reduction is expected because the two paths have different prior uncertainties; the shorter (lower-uncertainty) path benefits more from the constraint.

3.4 Point and Distance Queries

The point query correctly decomposes uncertainty into a pose-driven term and a local measurement term. The correlation-aware relative vector query produces strictly smaller covariance than the naive (independent) estimate when the two points share kinematic edges, confirming that the shared edges are correctly accounted for.

4. Significance and Conclusions

This project demonstrates that first-order Gaussian uncertainty propagation in SE(3) is both mathematically tractable and practically efficient. The key contributions are:

1. A complete, tested SE(3) math layer following CIS I conventions
2. A composable `UncertainTransform` primitive with correct adjoint-based propagation
3. A `GeometricNetwork` API that answers all common uncertainty queries with a single function call
4. Unified multi-path fusion via the Gaussian linear system (S-matrix) formulation
5. A general observation/factor abstraction that handles loop, point, and distance constraints in a single joint information filter
6. Automatic loop discovery via spanning-tree cycle basis
7. Full Monte Carlo validation with $< 1\%$ error across all scenarios
8. Interactive 3D visualization of uncertainty ellipsoids and kinematic networks

The framework is ready for use in surgical robotics research and can be extended to real sensor data, ROS integration, or SLAM-style full state estimation as future work.

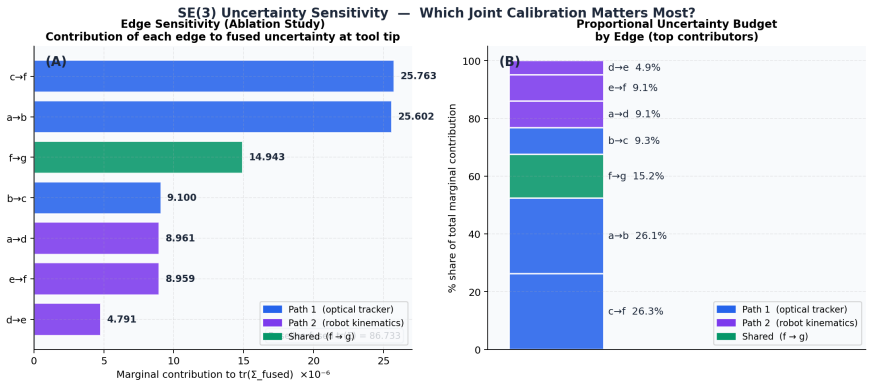

Figure 4: Edge sensitivity study, which calibration edge contributes most to fused uncertainty

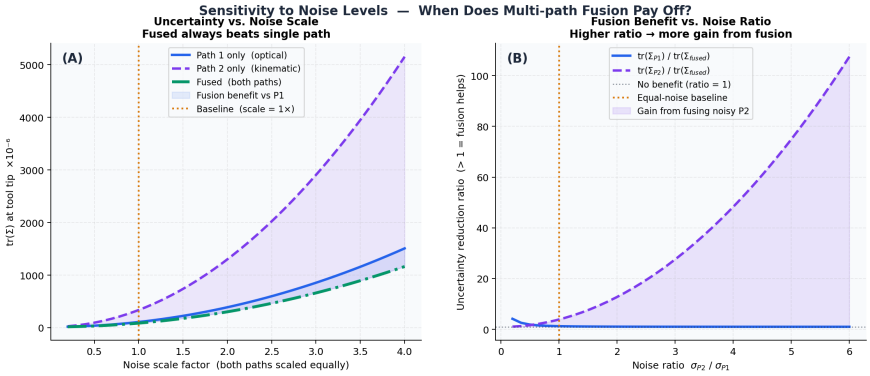

Figure 5: Fusion benefit vs noise scale — when does multi-path fusion pay off

Part II — Management Summary

Who Did What

This was a solo project. X.M. Christine Zhu designed, implemented, tested, and validated the entire framework under the guidance of the course mentor.

Mentor role: Provided the mathematical framework document (CIS I convention and pseudocode), reviewed design decisions, and gave feedback on the perturbation model and adjoint derivation.

Student role (Christine Zhu): All implementation, all unit tests, all Monte Carlo validation scripts, visualization module, README, and this report.

Accomplished vs. Planned

All twelve development phases were completed. The table below summarizes status:

Phase	Description	Status
0	Repository baseline, folder structure, package setup	Complete
1	SE(3) utilities (exp, log, adjoint)	Complete
2	UncertainTransform primitive (compose, inv)	Complete
3	Point propagation and CIS I Jacobians	Complete
4	GeometricNetwork (frames, BFS, path queries)	Complete
5	Monte Carlo validation (forward propagation)	Complete
6	Closed-loop conditioning module	Complete
7	Closed-loop integrated with the network	Complete
8	Multiple simultaneous loop constraints	Complete
9	Automatic loop discovery (cycle basis)	Complete
10	Observation/factor abstraction	Complete
11	Usability and project polish	Partial
12	Visualization (static + interactive 3D)	Complete

Table 3: Development phase completion status.

Phase 11 is partially complete: the README examples are written but two items are still pending — (1) a section explaining closed-loop capabilities in the README and (2) a `run_all_validations.py`

convenience script that runs all nine Monte Carlo scripts and prints a summary table. These are documentation/usability items that do not affect correctness.

The originally listed out-of-scope items — real sensor data integration, ROS runtime interface, AMBF simulation integration, and full SLAM-style state estimation — remain out of scope and were never planned for this semester.

What Might Be Next (Summer Work Plan)

The following extensions are planned for summer 2026:

1. **Real sensor data integration**: Connect the framework to actual optical tracker and robot kinematics data streams. Parse real calibration matrices and their reported covariances. Validate that analytic predictions match empirical scatter from real measurements.
2. **ROS 2 interface**: Wrap `GeometricNetwork` as a ROS 2 node that subscribes to TF2 transforms and covariance-stamped messages, maintaining a live uncertainty graph that updates in real time as the robot moves.
3. **AMBF simulation validation**: Use the Asynchronous Multi-Body Framework (AMBF) to simulate a surgical robotic scenario with known ground-truth transforms. Run the uncertainty propagation framework in parallel and compare predicted uncertainty bounds to simulated scatter.
4. **Advanced cross-covariance propagation**: The current framework assumes edges are independent. In practice, some edges may share calibration parameters. Extending the S-matrix formulation to model these dependencies would improve accuracy for tightly coupled systems.
5. `run_all_validations.py` **and README update**: Complete Phase 11 — write the convenience validation runner and expand the README to document the closed-loop and observation APIs.

What Was Learned

Several lessons shaped this project:

- **SE(3) composition is not commutative and covariance propagation reflects this.** Early attempts to simply add covariances without the adjoint transformation gave wrong answers that only became apparent in Monte Carlo tests. The adjoint is not optional.
- **Multi-path fusion is more subtle than it appears.** When paths share edges (as they do in many real networks), naive path enumeration double-counts shared uncertainty. The S-matrix formulation with canonical edge variables correctly handles this by tracking which edges appear in which paths.
- **First-order approximation is surprisingly accurate.** Across all scenarios tested, the analytic (linearized) result agreed with Monte Carlo to better than 2% even for moderately large perturbations. This validates the practical utility of the Gaussian framework.
- **Monte Carlo validation is essential, not optional.** Several bugs in the closed-loop Jacobian computation were caught only by Monte Carlo tests, not by unit tests on the math. The empirical covariance is an unforgiving ground truth.
- **ESS stabilization for importance-weighted Monte Carlo.** Closed-loop conditioning is not naturally amenable to direct Monte Carlo (the loop constraint is a zero-probability event in

continuous space). Using importance weighting with a residual noise model, and auto-tuning that noise to maintain sufficient effective sample size (ESS), was the key insight that made the closed-loop MC validation work reliably.

Part III — Technical Appendices

A. Repository

The full source code is maintained in a GitHub repository (https://github.com/X-M-Zhu/uncertainty_propagation_surgical_robotics.git). All code, tests, scripts, and documentation are contained within the `kinematic_error_propagation/` directory.

Repository structure:

```
src/uncertainty_networks/
    se3.py                  SE(3) math layer
    uncertain_geometry.py   UncertainTransform primitive
    network.py              GeometricNetwork and all query methods
    closed_loop.py          Loop conditioning and multi-constraint fusion
    observations.py         Observation/factor abstraction
    visualization.py        Static and interactive 3D visualization

tests/
    test_se3.py
    test_uncertain_geometry.py
    test_network.py
    test_closed_loop.py
    test_observations.py
    test_batch_eval.py

scripts/
    validate_open_chain_mc.py
    validate_point_mc.py
    validate_closed_loop_mc.py
    validate_random_network_mc.py
    validate_shared_infrastructure_mc.py
    plot_network.py

results/                    Generated figures (saved by plot_network.py)

docs/
    math_note.pdf           Mathematical framework document
    project_report.tex      This report (LaTeX source)
```

B. Required Documents

Wiki page: https://ciis.lcsr.jhu.edu/doku.php?id=courses:456:2026:projects:456-2026-01:project-01
GitHub repo: https://github.com/X-M-Zhu/uncertainty_propagation_surgical_robotics.git

Document	Location	Description
Functional Specification	`docs/` / Wiki	Defines the SE(3) perturbation model, adjoint convention, and API contracts for all query methods
Math Note (PDF)	`docs/math_note.pdf`	Derivation of all formulas: adjoint propagation, loop residual, information filter, point Jacobian
Pseudocode	`PSEUDOCODE.md`	High-level pseudocode and system map for all modules
README	`README.md`	Installation, tutorial examples, API table, key conventions
Unit Tests	`tests/`	Pytest test suite covering all modules; run with `pytest`
Monte Carlo Scripts	`scripts/validate.py`	Validation scripts; each prints analytic vs. MC covariance and Frobenius error

Table 4: Summary of required documents.

C. Running the Code

Install:

```
python3 -m venv venv && source venv/bin/activate
pip install -e .
```

Run tests:

```
pytest
```

Run a validation script:

```
python scripts/validate_open_chain_mc.py
```

Generate visualization figures:

```
python scripts/plot_network.py
```

Figures are saved to `results/`.

Part IV — References

1. T. D. Barfoot, *State Estimation for Robotics*, 2nd ed., draft manuscript, Jan. 27, 2026.
2. T. D. Barfoot and P. T. Furgale, "Associating uncertainty with three-dimensional poses for use in estimation problems," *IEEE Transactions on Robotics*, vol. 30, no. 3, pp. 679–693, Jun. 2014, doi: 10.1109/TRO.2014.2298059.
3. Wikipedia contributors, "Propagation of uncertainty," *Wikipedia, The Free Encyclopedia*. [Online]. Available: https://en.wikipedia.org/wiki/Propagation_of_uncertainty
4. Wikipedia contributors, "Uncertainty quantification," *Wikipedia, The Free Encyclopedia*. [Online]. Available: https://en.wikipedia.org/wiki/Uncertainty_quantification
5. Wikipedia contributors, "Kalman filter," *Wikipedia, The Free Encyclopedia*. [Online]. Available: https://en.wikipedia.org/wiki/Kalman_filter
6. Wikipedia contributors, "Measure (mathematics)," *Wikipedia, The Free Encyclopedia*. [Online]. Available: https://en.wikipedia.org/wiki/Measure_(mathematics)