SIMULATION-BASED UNCERTAINTY PROPAGATION IN GEOMETRIC NETWORKS FOR SURGICAL ROBOTICS

Functional Specification

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

Contents

Review and Major Update Log

Date	**Description of Changes**	**Person Changing**	**Person Reviewing**	**Comments**
03/09/2026	Initial draft of functional specification	X.M. Christine Zhu	–	Initial version
03/10/2026	Added supported object types, supported operations, operator overloading note, and review log per mentor feedback	X.M. Christine Zhu	Russ Taylor	Revised after feedback
03/13/2026	Added missing subroutines	X.M. Christine Zhu	Russ Taylor	Revised after feedback. OK to leave the algorithms here.
03/30/2026	Added Section 8: combining relationship chains, independence requirement, and observation abstraction (Gaussian fusion vs. information filter)	X.M. Christine Zhu	Russ Taylor	

1 System Overview

Modern surgical robotic systems rely on multiple sensing modalities and coordinate frames, including medical imaging systems, optical tracking systems, surgical tools, and anatomical models. Each transformation between coordinate frames introduces uncertainty arising from measurement noise, calibration error, and registration inaccuracies.

The goal of this project is to develop a simulation framework that models such systems as geometric networks and predicts how uncertainty propagates through these networks.

In the proposed framework:

- Nodes represent coordinate frames.
- Edges represent rigid transformations and related geometric quantities with associated uncertainty.
- Queries compute geometric relationships while propagating uncertainty.

2 Mathematical Representation

Rigid transformations are represented as

$$F = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$

Pose uncertainty is modeled using a 6-DOF perturbation vector

$$\vec{\eta} = \begin{bmatrix} \vec{\alpha} \\ \vec{\epsilon} \end{bmatrix}$$

where

- $\vec{\alpha}$ represents rotational error
- $\vec{\epsilon}$ represents translational error

The uncertainty is modeled as

$$\vec{\eta} \sim \mathcal{N}(0, C_{\eta\eta})$$

where $C_{\eta\eta}$ is a 6×6 covariance matrix.

3 System Architecture

The framework consists of five main modules:

1. SE(3) transformation utilities
2. uncertainty network representation
3. uncertainty propagation engine

4. query engine
5. simulation validation module

System workflow:

Network Definition → Uncertainty Propagation → Query Engine → Validation

4 Data Structures

4.1 Node

Nodes represent coordinate frames such as

- CT frame
- anatomy frame
- tracker frame
- tool frame
- drill tip frame
- single location, eg. marker, tool tip

4.2 Edge

Edges represent geometric objects and relationships used for propagation through the network. Depending on the application, an edge may represent:

- a deterministic rigid frame
- a rigid frame with uncertainty
- a deterministic rotation
- a rotation with uncertainty
- a deterministic translation vector
- a translation vector with uncertainty

Each edge stores the data required for the represented object, including mean value, associated covariance when applicable, and the parent and child frame labels needed for graph traversal and composition.

4.3 Supported Geometric Object Types

The framework shall support the following geometric object types as first-class entities in the uncertainty network and associated computational library:

- deterministic rigid frames
- rigid frames with uncertainty
- deterministic rotations
- rotations with uncertainty
- deterministic translation vectors
- translation vectors with uncertainty

A deterministic rigid frame is represented by a homogeneous transformation

$$F = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$

where R is a rotation matrix and t is a translation vector.

A frame with uncertainty is represented by a mean transformation together with an associated covariance on pose perturbation.

A deterministic rotation is represented by a rotation matrix R.

A rotation with uncertainty is represented by a mean rotation together with an associated rotational covariance.

A deterministic translation vector is represented by a vector $t \in \mathbb{R}^3$.

A translation vector with uncertainty is represented by a mean vector together with an associated covariance.

5 Functional Components

5.1 Network Path Search

Algorithm 1 Find Network Path

Require: Network graph, start frame A, target frame B

Ensure: Path connecting A and B

```
Initialize queue
Mark all nodes as unvisited
Enqueue A
while queue not empty do
    current ← dequeue node
    if current = B then
        return path
    end if
    for each neighbor do
        if neighbor not visited then
            record predecessor
            enqueue neighbor
        end if
    end for
end while
return failure
```

5.2 Build Uncertainty Network

Algorithm 2 Build Uncertainty Network

Require: Node list, transform list

Ensure: Network graph

```
Initialize empty network
for each node do
    add node to network
end for
for each transform do
    create edge with covariance
    add edge to network
end for
return network
```

5.3 Compose Transformations

Algorithm 3 Compose Transformations with Uncertainty

Require: $F_{ab}, C_{ab}, F_{bc}, C_{bc}$

$F_{ac} \leftarrow F_{ab} F_{bc}$

compute Jacobians J_1, J_2

$C_{ac} \leftarrow J_1 C_{ab} J_1^T + J_2 C_{bc} J_2^T$

return F_{ac}, C_{ac}

5.4 Frame–Frame Query

Algorithm 4 Frame–Frame Query

Require: Frames A, B

path $\leftarrow$ FindNetworkPath(A, B)

propagate transforms along path

return relative pose and covariance

5.5 Frame–Point Query

Algorithm 5 Frame–Point Query

Require: F_{AB}, C_{AB}, p_B

$p_A \leftarrow F_{AB} p_B$

compute Jacobian J

$C_p \leftarrow J C_{AB} J^T$

return p_A, C_p

5.6 Point–Point Distance

Algorithm 6 Point–Point Distance

Require: p_1, p_2, C_1, C_2

$\delta \leftarrow p_1 - p_2$

$d \leftarrow \|\delta\|$

compute Jacobian J

$\sigma^2 \leftarrow J(C_1 + C_2)J^T$

return d, σ^2

5.7 Closed Loop Fusion

Algorithm 7 Closed Loop Gaussian Fusion

Require: pose estimates μ_k, C_k

$I \leftarrow 0$
$w \leftarrow 0$
for $k = 1..n$ **do**
    $I \leftarrow I + C_k^{-1}$
    $w \leftarrow w + C_k^{-1}\mu_k$
end for
$C \leftarrow I^{-1}$
$\mu \leftarrow Cw$
return μ, C

6 Supported Operations

The framework shall provide a subroutine library supporting geometric computations involving deterministic and uncertain objects. At the functional specification level, the required capability is that the following categories of operations shall be supported.

6.1 Frame–Vector Operations

- $F * v$: deterministic frame acting on a deterministic vector
- $F * v_E$: deterministic frame acting on a vector with uncertainty
- $F_E * v$: uncertain frame acting on a deterministic vector
- $F_E * v_E$: uncertain frame acting on a vector with uncertainty

6.2 Frame–Frame Operations

- $F * F$: deterministic frame composed with deterministic frame
- $F_E * F$: uncertain frame composed with deterministic frame
- $F * F_E$: deterministic frame composed with uncertain frame
- $F_E * F_E$: uncertain frame composed with uncertain frame
- frame composition along multi-edge network paths
- inversion of deterministic and uncertain frame representations as required for path traversal

6.3 Rotation–Vector Operations

- $R * v$: deterministic rotation acting on a deterministic vector
- $R * v_E$: deterministic rotation acting on a vector with uncertainty
- $R_E * v$: uncertain rotation acting on a deterministic vector
- $R_E * v_E$: uncertain rotation acting on a vector with uncertainty

6.4 Translation Operations

- deterministic translation vector operations
- translation vector operations with uncertainty
- addition and subtraction of deterministic translation vectors
- addition and subtraction involving uncertain translation vectors
- propagation of covariance through translation addition and subtraction

6.5 Uncertainty Propagation Support

For all supported operations involving uncertain objects, the framework shall propagate uncertainty using the appropriate first-order covariance propagation rules. Exact implementation details, internal class design, and numerical strategies are part of the implementation specification.

6.6 Operator Overloading

For usability and clarity of expression, it is desirable that the software support operator overloading for core geometric operations wherever practical. In particular, expressions such as frame composition and frame–vector application should be expressible in a natural algebraic form. The exact implementation of operator overloading is part of the implementation specification rather than the functional specification.

6.7 Required Subroutine Library Support

The framework shall include a subroutine library supporting composition, application, and inversion operations for deterministic and uncertainty-bearing geometric objects. At the functional specification level, the required capability is that these subroutines shall be supported, while detailed mathematical derivations and implementation choices may be deferred to the implementation specification.

At a minimum, the subroutine library shall support the following classes of operations:

- frame acting on vector:

$$F * v, \qquad F * v_E, \qquad F_E * v, \qquad F_E * v_E$$

- frame composition:

$$F * F, \qquad F_E * F, \qquad F * F_E, \qquad F_E * F_E$$

- rotation acting on vector:

$$R * v, \qquad R * v_E, \qquad R_E * v, \qquad R_E * v_E$$

- deterministic and uncertain translation-vector operations
- inversion of deterministic and uncertainty-bearing frame representations as required for path traversal and network propagation

These requirements are stated here at the functional level only. Exact formulas, Jacobian definitions, covariance propagation details, and software design choices are part of the implementation specification.

7 Primitive Geometric Subroutines

This section describes the primitive subroutines required to support geometric operations involving deterministic and uncertainty-bearing frames, rotations, and vectors. These routines form the computational basis for higher-level path propagation, query evaluation, and closed-loop estimation within the uncertainty network framework.

Throughout this section, we use first-order linearization for uncertainty propagation. Covariances are represented using CIS-style notation.

7.1 Notation

Let

$$F = \begin{bmatrix} R & t \\ 0 & 1 \end{bmatrix}$$

denote a deterministic rigid transformation, where $R \in SO(3)$ and $t \in \mathbb{R}^3$.

Let

$$F_E = (F, C_{\eta\eta})$$

denote a frame with uncertainty, where

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

is a small pose perturbation.

Let

$$R_E = (R, C_{\alpha\alpha})$$

denote a rotation with uncertainty, and let

$$v_E = (v, C_{vv})$$

denote a vector with uncertainty.

7.2 Helper Routine: Skew-Symmetric Matrix

Algorithm 8 Construct Skew-Symmetric Matrix

Require: Vector $v = [v_1 \;\; v_2 \;\; v_3]^T$

Ensure: Skew-symmetric matrix $[v]_\times$

1:
$$[v]_\times \leftarrow \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix}$$

2: **return** $[v]_\times$

7.3 Frame–Vector Operations

Algorithm 9 Apply Deterministic Frame to Deterministic Vector ($F * v$)

Require: Frame $F = (R, t)$, vector v

Ensure: Transformed vector v'

1: $v' \leftarrow Rv + t$

2: **return** v'

Algorithm 10 Apply Uncertain Frame to Deterministic Vector ($F_E * v$)

Require: Mean frame $F = (R, t)$, pose covariance $C_{\eta\eta}$, vector v

Ensure: Mean transformed vector v', covariance $C_{v'v'}$

1: $v' \leftarrow Rv + t$

2: Construct $[v]_\times$

3: Compute Jacobian with respect to pose perturbation:
$$J_\eta \leftarrow \begin{bmatrix} -R[v]_\times & I \end{bmatrix}$$

4: $C_{v'v'} \leftarrow J_\eta C_{\eta\eta} J_\eta^T$

5: **return** $v', C_{v'v'}$

Algorithm 11 Apply Uncertain Frame to Uncertain Vector ($F_E * v_E$)

Require: Mean frame $F = (R, t)$, pose covariance $C_{\eta\eta}$, mean vector v, vector covariance C_{vv}

Ensure: Mean transformed vector v', covariance $C_{v'v'}$

1: $v' \leftarrow Rv + t$

2: Construct $[v]_\times$

3: Compute Jacobian with respect to pose perturbation:

$$J_\eta \leftarrow \begin{bmatrix} -R[v]_\times & I \end{bmatrix}$$

4: Compute Jacobian with respect to vector:

$$J_v \leftarrow R$$

5: $C_{v'v'} \leftarrow J_\eta C_{\eta\eta} J_\eta^T + J_v C_{vv} J_v^T$

6: **return** $v', C_{v'v'}$

7.4 Frame–Frame Operations

Algorithm 12 Compose Uncertain Frame with Deterministic Frame ($F_E * F$)

Require: Uncertain frame (F_1, C_1), deterministic frame F_2

Ensure: Composed frame F_{12}, covariance C_{12}

1: $F_{12} \leftarrow F_1 F_2$

2: Compute left-composition Jacobian:

$$J_1 \leftarrow \text{JacobianComposeLeft}(F_1, F_2)$$

3: $C_{12} \leftarrow J_1 C_1 J_1^T$

4: **return** F_{12}, C_{12}

Algorithm 13 Compose Deterministic Frame with Uncertain Frame ($F * F_E$)

Require: Deterministic frame F_1, uncertain frame (F_2, C_2)

Ensure: Composed frame F_{12}, covariance C_{12}

1: $F_{12} \leftarrow F_1 F_2$

2: Compute right-composition Jacobian:

$$J_2 \leftarrow \text{JacobianComposeRight}(F_1, F_2)$$

3: $C_{12} \leftarrow J_2 C_2 J_2^T$

4: **return** F_{12}, C_{12}

Algorithm 14 Compose Two Uncertain Frames ($F_E * F_E$)

Require: Uncertain frames (F_1, C_1) and (F_2, C_2)

Ensure: Composed frame F_{12}, covariance C_{12}

1: $F_{12} \leftarrow F_1 F_2$

2: Compute left-composition Jacobian:

$$J_1 \leftarrow \text{JacobianComposeLeft}(F_1, F_2)$$

3: Compute right-composition Jacobian:

$$J_2 \leftarrow \text{JacobianComposeRight}(F_1, F_2)$$

4: $C_{12} \leftarrow J_1 C_1 J_1^T + J_2 C_2 J_2^T$

5: **return** F_{12}, C_{12}

7.5 Rotation–Vector Operations

Algorithm 15 Apply Deterministic Rotation to Deterministic Vector ($R * v$)

Require: Rotation R, vector v

Ensure: Rotated vector v'

1: $v' \leftarrow Rv$

2: **return** v'

Algorithm 16 Apply Deterministic Rotation to Uncertain Vector ($R * v_E$)

Require: Rotation R, mean vector v, covariance C_{vv}

Ensure: Rotated vector v', covariance $C_{v'v'}$

1: $v' \leftarrow Rv$

2: $C_{v'v'} \leftarrow R C_{vv} R^T$

3: **return** $v', C_{v'v'}$

Algorithm 17 Apply Uncertain Rotation to Deterministic Vector ($R_E * v$)

Require: Mean rotation R, rotational covariance $C_{\alpha\alpha}$, vector v

Ensure: Rotated vector v', covariance $C_{v'v'}$

1: $v' \leftarrow Rv$

2: Construct $[v]_\times$

3: Compute Jacobian with respect to rotational perturbation:

$$J_\alpha \leftarrow -R[v]_\times$$

4: $C_{v'v'} \leftarrow J_\alpha C_{\alpha\alpha} J_\alpha^T$

5: **return** $v', C_{v'v'}$

Algorithm 18 Apply Uncertain Rotation to Uncertain Vector ($R_E * v_E$)

Require: Mean rotation R, rotational covariance $C_{\alpha\alpha}$, mean vector v, covariance C_{vv}

Ensure: Rotated vector v', covariance $C_{v'v'}$

1: $v' \leftarrow Rv$

2: Construct $[v]_\times$

3: Compute Jacobian with respect to rotation perturbation:

$$J_\alpha \leftarrow -R[v]_\times$$

4: Compute Jacobian with respect to vector:

$$J_v \leftarrow R$$

5: $C_{v'v'} \leftarrow J_\alpha C_{\alpha\alpha} J_\alpha^T + J_v C_{vv} J_v^T$

6: **return** $v', C_{v'v'}$

7.6 Translation–Vector Operations

Algorithm 19 Add Deterministic Translation to Deterministic Vector ($t + v$)

Require: Translation vector t, vector v

Ensure: Translated vector v'

1: $v' \leftarrow t + v$

2: **return** v'

Algorithm 20 Add Uncertain Translation to Deterministic Vector ($t_E + v$)

Require: Mean translation t, covariance C_{tt}, vector v

Ensure: Mean translated vector v', covariance $C_{v'v'}$

1: $v' \leftarrow t + v$

2: $C_{v'v'} \leftarrow C_{tt}$

3: **return** $v', C_{v'v'}$

Algorithm 21 Add Uncertain Translation to Uncertain Vector ($t_E + v_E$)

Require: Mean translation t, covariance C_{tt}, mean vector v, covariance C_{vv}

Ensure: Mean translated vector v', covariance $C_{v'v'}$

1: $v' \leftarrow t + v$

2: $C_{v'v'} \leftarrow C_{tt} + C_{vv}$

3: **return** $v', C_{v'v'}$

7.7 Supporting Composition Jacobians

The frame-composition routines above refer to the following abstract subroutines:

- JacobianComposeLeft(F_1, F_2)
- JacobianComposeRight(F_1, F_2)

These compute the first-order Jacobians that map perturbations in the left or right input frame, respectively, to perturbations in the composed output frame. Their exact matrix forms depend on the chosen local perturbation convention and are specified in the mathematical derivation section of the implementation specification.

7.8 Remarks

These primitive subroutines are intended to support:

- forward uncertainty propagation through transformation chains,
- query evaluation in the uncertainty network,
- closed-loop fusion and estimation routines,
- operator-overloaded implementations of geometric expressions.

Higher-level network algorithms may therefore treat these routines as basic computational building blocks.

8 Combining Relationship Chains and Observation Abstraction

8.1 General Formula

Given a set of observation equations, each linking a target perturbation η_0 to source perturbations $\{\eta_j\}$ via adjoint matrices $A_{i,j}$:

$$A_{i,0}\,\eta_0 \;=\; \sum_j A_{i,j}\,\eta_j, \qquad i = 1, \ldots, N \tag{1}$$

the combined information accumulated at node 0 from all N equations is:

$$B_j \;=\; \sum_i A_{i,j}^T A_{i,j} \tag{2}$$

giving posterior covariance $C_{\text{post}} = B_0^{-1}$ when the noise terms are identity-scaled.

8.2 Independence Requirement

The formula above is valid only when each perturbation η_j appears in *at most one* observation equation. If the same edge (i.e. the same η_j) appears in multiple equations, the $A_{i,j}^T A_{i,j}$ terms cannot simply be summed — doing so double-counts information and produces an overconfident posterior. In that case a joint information filter over the full stacked system is required.

8.3 Two Cases in Implementation

Case 1 — Independent paths (no shared edges). When two paths from A to B share no edges, their covariance estimates are independent. The posterior is the Gaussian fusion:

$$C_{\text{fused}}^{-1} \;=\; C_{\text{path 1}}^{-1} + C_{\text{path 2}}^{-1} \tag{3}$$

Implemented in `fuse_gaussians` (see `src/uncertainty_networks/closed_loop.py`).

Case 2 — Shared edges. When paths share one or more edges, a joint information update is required:

$$C_{\text{post}}^{-1} \;=\; C_0^{-1} + H^T C_\nu^{-1} H \tag{4}$$

where H is the stacked Jacobian matrix from all observations and C_ν is the block-diagonal noise covariance. Implemented in `condition_on_observations` (`src/uncertainty_networks/observations.py`, line 441).

8.4 Observation Abstraction

To support heterogeneous constraints, a unified `Observation` interface is defined with three concrete types:

- **LoopObservation** — SE(3) loop closure: $r = \log(T_{\text{res}}^{-1} T_k) \approx 0$, 6-dimensional residual.
- **PointObservation** — observed point position: residual $r = p_{\text{nom}} - z$, Jacobian $J_\eta = [-[p']_\times \;\; I_3]$ (CIS I convention).
- **DistanceObservation** — scalar distance factor: $r = \|p_1 - p_2\| - z$, 1-dimensional residual.

All three types expose the same interface (`residual()`, `jacobians()`, `noise_cov()`) and are consumed uniformly by `condition_on_observations`.

8.5 Remarks

The independence caveat applies directly to the surgical network: the Tracker–Tool edge is shared between the direct measurement path and any path routed through the Robot Base. Blindly stacking those observation equations would violate the independence assumption. The correct approach is `condition_on_observations`, which handles shared edges without double-counting.

9 Kalman Filter for Pose Estimation

9.1 Use of Kalman Filtering in the Proposed Framework

In addition to static uncertainty propagation through transformation chains, the proposed framework supports dynamic state estimation using a Kalman filter.

In practical surgical robotic systems, measurements from tracking systems arrive sequentially over time. For example, an optical tracker continuously observes the marker bodies

attached to surgical tools and anatomy. These measurements contain noise and may vary over time due to occlusion, tracking jitter, or sensor inaccuracies.

The Kalman filter is used to estimate the pose of tracked objects by combining previous state estimates with new measurements. This provides a recursive method for reducing measurement noise and maintaining a consistent estimate of the system state.

In the context of the proposed framework, the Kalman filter may be applied to estimate quantities such as:

- the pose of the surgical tool relative to the tracker frame,
- the pose of anatomy markers relative to the tracker,
- the time-varying pose of the drill tip during tool motion.

At each time step, the Kalman filter performs two operations:

Prediction Step

The system predicts the next state using a motion or state transition model.

Measurement Update

The predicted state is corrected using new sensor measurements obtained from the tracking system.

The filtered state estimate and covariance produced by the Kalman filter can then be used as inputs to the geometric uncertainty propagation framework. This allows the system to combine dynamic measurement filtering with analytic uncertainty propagation through the geometric network.

By integrating Kalman filtering into the framework, the system can support both:

- real-time pose estimation from sensor measurements, and
- analytic prediction of uncertainty propagation through the surgical navigation system.

Let the state vector be

$$x$$

with covariance

$$P$$

A measurement is represented as

$$z$$

with measurement model

$$z = Hx + v$$

where

- H is the measurement matrix
- $v \sim \mathcal{N}(0, C_z)$ is measurement noise

9.2 Prediction Step

The state prediction is

$$x^- = Ax$$

The predicted covariance is

$$P^- = APA^T + Q$$

where

- A is the state transition model
- Q is process noise covariance

9.3 Update Step

The Kalman gain is

$$K = P^- H^T (HP^- H^T + C_z)^{-1}$$

The state estimate is updated as

$$x = x^- + K(z - Hx^-)$$

The covariance update is

$$P = (I - KH)P^-$$

9.4 Kalman Filter State Estimation

Algorithm 22 Kalman Filter Update

Require: Previous state estimate x, covariance P

Require: measurement z

Predict state:

$x^- \leftarrow Ax$

Predict covariance:

$P^- \leftarrow APA^T + Q$

Compute Kalman gain:

$K \leftarrow P^- H^T (HP^- H^T + C_z)^{-1}$

Update state:

$x \leftarrow x^- + K(z - Hx^-)$

Update covariance:

$P \leftarrow (I - KH)P^-$

return x, P

10 Monte Carlo Validation

Algorithm 23 Monte Carlo Validation

Require: network, query, samples N

1: **for** $i = 1..N$ **do**
2: sample pose noise
3: perturb transformations
4: compute query result
5: store result
6: **end for**
7: compute covariance
8: **return** empirical covariance

11 Example Surgical Navigation Scenario

Typical coordinate chain:

$$F_{CT,tip} = F_{CT,anatomy} F_{anatomy,tracker} F_{tracker,tool} F_{tool,tip}$$

Each transformation contains uncertainty arising from

- registration error
- tracking noise
- calibration error

12 Scope of Implementation

12.1 Minimum

- transformation utilities
- uncertainty propagation
- network representation
- frame–frame, frame–point, point–point queries
- support for deterministic and uncertain frames, rotations, and translation vectors

12.2 Expected

- Monte Carlo validation
- closed loop fusion
- additional query functions
- broader support for mixed deterministic/uncertain geometric operations

12.3 Maximum

- visualization tools
- AMBF simulation integration
- teaching demonstrations
- operator-overloaded user interface for core geometric operations

13 Assumptions

- Gaussian pose uncertainty
- first order linearization
- small angle approximation
- rigid body transformations

14 Conclusion

This document describes the functional specification for a simulation framework that analyzes uncertainty propagation in geometric networks for surgical robotics. The system models sensing architectures as networks of geometric transformations and related quantities with uncertainty, and supports queries for analyzing the resulting positional accuracy. The framework is intended to support both deterministic and uncertainty-bearing geometric objects, together with the core operations required for uncertainty propagation in surgical navigation and robotics applications.

Notation Sheet

Symbol	Description
F_{ab}	Rigid transformation from frame b to frame a
F_E	Frame with uncertainty
R	Rotation matrix in a rigid transformation
R_E	Rotation with uncertainty
t	Translation vector in a rigid transformation
t_E	Translation vector with uncertainty
p_a	Coordinates of a point expressed in frame a
p_b	Coordinates of a point expressed in frame b
v	Deterministic vector or point
v_E	Vector or point with uncertainty
$\vec{\eta}$	6-DOF pose perturbation vector
$\vec{\alpha}$	Rotational error component of pose perturbation
$\vec{\epsilon}$	Translational error component of pose perturbation
C	Covariance matrix representing uncertainty
C_{ab}	Covariance associated with transformation F_{ab}
$C_{\eta\eta}$	Covariance associated with pose perturbation vector $\vec{\eta}$
C_p	Covariance of a propagated point estimate
C_z	Measurement noise covariance in Kalman filtering
J	Jacobian matrix used for uncertainty propagation
d	Distance between two points
σ^2	Variance of a scalar quantity
$\mathcal{N}(\mu, C)$	Multivariate Gaussian distribution with mean μ and covariance C
$\|\cdot\|$	Euclidean norm
$F_{CT,tip}$	Transformation from drill tip frame to CT frame
C_{tip}	Covariance of drill tip position in CT frame
N	Number of Monte Carlo samples
μ	Mean of a Gaussian distribution
I	Information matrix (inverse covariance)
x	State vector in Kalman filtering
P	Covariance of state estimate
A	State transition matrix
Q	Process noise covariance
z	Measurement vector
H	Measurement matrix
K	Kalman gain