Target Occlusion Detection for GPS-Denied Rover Systems

Table of Contents

Introduction

Target occlusion occurs when terrain features, buildings, or other obstructions interfere with direct line-of-sight (LOS) communication between drone anchors and a GPS-denied rover. This interference corrupts Time-of-Flight (ToF) distance measurements (see ToF), making accurate multilateration impossible (see multilateration). This document provides comprehensive methods for detecting occlusion without relying on visual sensors.

Problem Statement

In a GPS-denied environment, a rover relies on multiple drone anchors (minimum 4 for 3D positioning) to determine its location through multilateration. When one or more anchors become occluded:

- ToF measurements become invalid due to Non-Line-of-Sight (NLOS) conditions
- Distance measurements may be too long (signal reflection/diffraction) or unavailable
- Multilateration calculations produce incorrect or unstable position estimates
- System accuracy degrades below acceptable thresholds

Detection Requirements

Target occlusion detection must:

- Identify which specific anchors are occluded
- Distinguish between single and multiple occlusions

- Operate without visual confirmation
- Provide results within the system's measurement cycle timing
- Maintain computational efficiency for real-time operation

Fundamental Principles

Line-of-Sight (LOS) vs Non-Line-of-Sight (NLOS)

Line-of-Sight (LOS) Conditions:

- Direct radio path between anchor and rover
- ToF measurements represent true geometric distance
- Signal strength follows predictable path loss models
- All anchor distance circles/spheres converge at rover position

Non-Line-of-Sight (NLOS) Conditions:

- Signal path obstructed by terrain, buildings, or vegetation
- Signal arrives via reflection, diffraction, or scattering
- Measured distance typically exceeds true geometric distance
- Distance circles/spheres fail to converge at common point

Geometric Consistency Principle

Under LOS conditions, all anchor distance measurements must be geometrically consistent. This means:

- In 2D: Distance circles from all anchors intersect at the rover's position
- In 3D: Distance spheres from all anchors intersect at the rover's position
- Any deviation from perfect intersection indicates measurement error or occlusion

Triangle Inequality Constraint

For any three points (two anchors and rover), the triangle inequality must hold:

$$|d_1 - d_2| \leq d_{12} \leq d_1 + d_2$$

Where:

- d_1 = distance from anchor 1 to rover
- d_2 = distance from anchor 2 to rover
- d_{12} = distance between anchors 1 and 2

Violation of this constraint indicates NLOS conditions.

Geometric Occlusion Detection

Geometric detection examines whether distance measurements from all anchors are mutually consistent through geometric intersection analysis. The fundamental principle is that under ideal LOS conditions, all distance circles (2D) or spheres (3D) must intersect at exactly one point - the rover's position. This intersection point represents the multilateration solution where the number of intersecting geometric shapes equals the number of anchors.

2D Geometric Detection Algorithm

Step 1: Initialize Parameters

1. Set geometric tolerance: ϵ_{geo} (sample accuracy constraint set to 0.1m)
2. Collect anchor positions: (x_i, y_i) for $i = 1, 2, \ldots, n$ anchors
3. Collect ToF distance measurements: r_i for each anchor i

Step 2: Generate Distance Circles

1. For each anchor i at position (x_i, y_i) with measured distance r_i:

$$\text{Circle}_i : (x - x_i)^2 + (y - y_i)^2 = r_i^2$$

Step 3: Calculate Pairwise Circle Intersections

1. For each pair of anchors (i, j) where $i < j$:
2. Calculate center separation distance:

$$d_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2}$$

3. Check intersection feasibility:

- If $d_{ij} > r_i + r_j + \epsilon_{geo}$: Circles too far apart (no intersection)

- If $d_{ij} < |r_i - r_j| - \epsilon_{geo}$: One circle inside the other (no intersection)

- If $d_{ij} = 0$ and $r_i = r_j$: Identical circles (infinite intersections)

4. If intersection exists, calculate intersection points:

$$a = \frac{r_i^2 - r_j^2 + d_{ij}^2}{2d_{ij}}$$

$$h = \sqrt{\max(r_i^2 - a^2, 0)}$$

$$x_m = x_i + a \cdot \frac{x_j - x_i}{d_{ij}}$$

$$y_m = y_i + a \cdot \frac{y_j - y_i}{d_{ij}}$$

$$x_{int1} = x_m + h \cdot \frac{y_j - y_i}{d_{ij}}$$

$$y_{int1} = y_m - h \cdot \frac{x_j - x_i}{d_{ij}}$$

$$x_{int2} = x_m - h \cdot \frac{y_j - y_i}{d_{ij}}$$

$$y_{int2} = y_m + h \cdot \frac{x_j - x_i}{d_{ij}}$$

Step 4: Identify Rover Position Through Maximum Intersection Analysis

1. Collect all intersection points from Step 3 into set $P = \{p_1, p_2, \ldots, p_m\}$
2. For each candidate point $p_k = (x_k, y_k)$, determine how many circles intersect at this location
3. Calculate distance from candidate to each anchor:

$$d_{k,i} = \sqrt{(x_k - x_i)^2 + (y_k - y_i)^2}$$

4. Count intersecting circles at each candidate point:

$$\text{intersection\_count}_k = \sum_{i=1}^{n} \begin{cases} 1 & \text{if } |d_{k,i} - r_i| \leq \epsilon_{geo} \\ 0 & \text{otherwise} \end{cases}$$

5. **Rover Position Identification**: The point with maximum intersection count represents the rover's multilateration solution:

$$\text{rover\_position} = \arg\max_k(\text{intersection\_count}_k)$$

Step 5: Occlusion Detection Through Intersection Analysis

1. Find maximum intersection count: $\text{count}_{max} = \max_k(\text{intersection\_count}_k)$
2. **Case 1: No Occlusion Detected**

- If $\text{count}_{max} = n$: All n circles intersect at the rover position

- This indicates perfect geometric consistency with all anchors in LOS

- Rover position is precisely determined through multilateration

3. **Case 2: Single Anchor Occlusion**

- If $\text{count}_{max} = n - 1$: Only $n - 1$ circles intersect at the rover position

- One anchor is geometrically inconsistent, indicating occlusion

- System can still determine position using $n - 1$ valid anchors

- Proceed to Step 6 for occluded anchor identification

4. **Case 3: Multiple Anchor Occlusions**

- If $\text{count}_{max} < n - 1$: Fewer than $n - 1$ circles intersect at any point

- Multiple anchors are occluded, severely compromising multilateration

- Example: With 4 anchors, if only 2 circles intersect at maximum, then 2+ anchors are occluded

- Proceed to Step 7 for comprehensive occlusion analysis

Step 6: Single Occlusion Identification

1. Identify the rover position from Step 5 (point with $\text{count}_{max} = n - 1$ intersections)
2. For each anchor i, calculate distance from rover position to anchor:

$$d_{rover,i} = \sqrt{(x_{rover} - x_i)^2 + (y_{rover} - y_i)^2}$$

3. Compare with measured ToF distance:

$$\text{error}_i = |d_{rover,i} - r_i|$$

4. The anchor with $\text{error}_i > \epsilon_{geo}$ is the occluded anchor
5. **Multilateration Solution**: Use remaining $n - 1$ consistent anchors for final position estimate

Step 7: Multiple Occlusion Analysis

1. Since no point has sufficient circle intersections ($\text{count}_{max} < n - 1$), systematically test anchor combinations:

- Test exclusion of 2 anchors: $\binom{n}{2}$ combinations

- For each combination, use remaining $n - 2$ anchors

- Check if remaining circles achieve intersection count = $n - 2$

2. Continue with larger exclusion sets if necessary:

- Test exclusion of 3 anchors: $\binom{n}{3}$ combinations

- Ensure minimum positioning requirements maintained (≥3 anchors for 2D, ≥4 for 3D)

1. **Selection Criteria**: Choose combination that:

- Achieves maximum intersection count equal to number of remaining anchors

- Retains maximum number of anchors for robust multilateration

- Maintains positioning accuracy requirements

3D Geometric Detection Algorithm

Step 1: Initialize Parameters

1. Set geometric tolerance: ϵ_{geo} (recommended: 0.1 m)

2. Collect anchor positions: (x_i, y_i, z_i) for $i = 1, 2, \ldots, n$ anchors
3. Collect ToF distance measurements: r_i for each anchor i

Step 2: Generate Distance Spheres

1. For each anchor i at position (x_i, y_i, z_i) with measured distance r_i:

$$\text{Sphere}_i : (x - x_i)^2 + (y - y_i)^2 + (z - z_i)^2 = r_i^2$$

Step 3: Calculate Sphere-Sphere Intersections

1. For each pair of anchors (i, j) where $i < j$:
2. Calculate center separation distance:

$$d_{ij} = \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2 + (z_j - z_i)^2}$$

3. Check intersection feasibility:

- If $d_{ij} > r_i + r_j + \epsilon_{geo}$: Spheres don't intersect

- If $d_{ij} < |r_i - r_j| - \epsilon_{geo}$: One sphere inside the other

- If $d_{ij} = 0$: Spheres are concentric

4. If intersection exists, calculate intersection circle parameters:

$$a = \frac{r_i^2 - r_j^2 + d_{ij}^2}{2d_{ij}}$$

$$h = \sqrt{\max(r_i^2 - a^2, 0)}$$

5. Find intersection circle center:

$$\mathbf{p}_{ij} = \mathbf{c}_i + a \cdot \frac{\mathbf{c}_j - \mathbf{c}_i}{d_{ij}}$$

Where $\mathbf{c}_i = (x_i, y_i, z_i)$ and $\mathbf{c}_j = (x_j, y_j, z_j)$

6. Define intersection circle:

- Center: $\mathbf{p}_{ij}$

- Normal vector: $\mathbf{n}_{ij} = \frac{\mathbf{c}_j - \mathbf{c}_i}{d_{ij}}$

- Radius: h

Step 4: Multilateration Solution Through Sphere Intersection (see multilateration)

1. Convert sphere intersections to overdetermined system $\mathbf{Ax} = \mathbf{b}$:

$$\mathbf{A} = \begin{bmatrix} n_{1x} & n_{1y} & n_{1z} \\ n_{2x} & n_{2y} & n_{2z} \\ \vdots & \vdots & \vdots \\ n_{mx} & n_{my} & n_{mz} \end{bmatrix}, \quad \mathbf{b} = \begin{bmatrix} \mathbf{n_1} \cdot \mathbf{p_1} \\ \mathbf{n_2} \cdot \mathbf{p_2} \\ \vdots \\ \mathbf{n_m} \cdot \mathbf{p_m} \end{bmatrix}$$

2. Solve for rover position using least squares:

$$\mathbf{x}_{rover} = (\mathbf{A}^T\mathbf{A})^{-1}\mathbf{A}^T\mathbf{b}$$

Step 5: Sphere Intersection Validation and Occlusion Detection

1. For the calculated rover position, determine how many spheres intersect at this point
2. Calculate distance error for each sphere:

$$\text{error}_i = \left| \sqrt{(x_{rover} - x_i)^2 + (y_{rover} - y_i)^2 + (z_{rover} - z_i)^2} - r_i \right|$$

3. Count spheres intersecting at rover position:

$$\text{intersection\_count} = \sum_{i=1}^{n} \begin{cases} 1 & \text{if error}_i \leq \epsilon_{geo} \\ 0 & \text{otherwise} \end{cases}$$

Step 6: 3D Occlusion Classification

1. **Case 1: No Occlusion**

- If $\text{intersection\_count} = n$: All n spheres intersect at rover position

- Perfect multilateration solution with all anchors in LOS

2. **Case 2: Single Occlusion**

- If $\text{intersection\_count} = n - 1$: One sphere doesn't intersect at rover position

- Single anchor occlusion detected

- Proceed to Step 7 for identification

3. **Case 3: Multiple Occlusions**

- If $\text{intersection\_count} < n - 1$: Multiple spheres fail to intersect

- Example: With 4 anchors, if only 2 spheres intersect, then 2+ anchors are occluded

- Proceed to Step 8 for comprehensive analysis

Step 7: Single Occlusion Identification (3D)

1. The rover position from Step 4 represents the multilateration solution using $n - 1$ valid anchors
2. Identify occluded anchor as the one with $\text{error}_i > \epsilon_{geo}$
3. Verify occlusion by re-solving multilateration without the suspected anchor:

- Exclude anchor with highest error

- Recalculate position using remaining $n - 1$ anchors

- Confirm improved geometric consistency

Step 8: Multiple Occlusion Analysis (3D)

1. Since fewer than $n - 1$ spheres intersect at the rover position, systematically exclude anchors:

- Rank anchors by error magnitude: $\text{error}_1 \geq \text{error}_2 \geq \ldots \geq \text{error}_n$

- Exclude highest-error anchors iteratively

2. For each exclusion combination:

- Re-solve multilateration with remaining anchors

- Check if intersection count equals number of remaining anchors

3. **Selection Criteria**: Choose configuration where:

- Intersection count = number of remaining anchors (perfect multilateration)

- Maintains ≥4 anchors (minimum for 3D positioning)

- Maximizes number of valid anchors for positioning robustness

Odometry-Based Occlusion Detection

Odometry-based detection uses the rover's motion constraints to validate distance changes between consecutive measurements.

Theoretical Foundation

The triangle inequality provides a fundamental constraint: if a rover moves distance L, the range to any anchor cannot change by more than L under LOS conditions.

Mathematical Constraint:

$$|\Delta r_i| \leq L + \epsilon_{odom}$$

Where:

- $\Delta r_i = r_{i,new} - r_{i,old}$ (change in range to anchor i)
- L = rover displacement magnitude
- ϵ_{odom} = odometry tolerance (accounts for measurement uncertainty)

Odometry-Based Detection Algorithm

Step 1: Initialize Odometry System

1. Set odometry tolerance: ϵ_{odom} (sample accuracy constraint set to 0.05 m)
2. Initialize rover position tracking system
3. Identify offsets in timestamps between odometry system, ToF systems (see ToF), and base station / Parent Drone

Step 2: Collect Motion Data

1. Record rover displacement vector from odometry:

- Wheel encoders: $(\Delta x_{wheel}, \Delta y_{wheel})$

- Accelerometer: $(\Delta x_{accel}, \Delta y_{accel}, \Delta z_{accel})$

- Time interval: Δt

2. Calculate displacement magnitude:

$$L = \sqrt{(\Delta x)^2 + (\Delta y)^2 + (\Delta z)^2}$$

Step 3: Validate Odometry Consistency

1. **Cross-check wheel encoders vs accelerometer:**

- If wheel movement detected but no acceleration: rover likely stuck

- If acceleration detected but no wheel movement: rover likely slipped/fell

- Flag inconsistent odometry for further analysis

2. **Speed reasonableness check:**

$$v_{apparent} = \frac{L}{\Delta t}$$

- If $v_{apparent}$ exceeds maximum rover capability: flag as error

Step 4: Calculate Range Changes

1. For each anchor i, compute range difference:

$$\Delta r_i = r_{i,new} - r_{i,old}$$

2. Apply triangle inequality test:

$$\text{violation}_i = \begin{cases}

\text{True} & \text{if } |\Delta r*i*| > *L* + *epsilon*{odom} \\

\text{False} & \text{otherwise}

\end{cases}$$

Step 5: Enhanced Geometric Validation (Optional)

1. If rover heading θ and anchor bearing ϕ_i are known, use law of cosines:

$$r_{i,expected} = \sqrt{r_{i,old}^2 + L^2 - 2 \cdot r_{i,old} \cdot L \cdot \cos(\phi_i - \theta)}$$

2. Compare with measured value:

$$\text{error}_i = |r_{i,new} - r_{i,expected}|$$

3. Flag if $\text{error}_i > \epsilon_{odom}$

Step 6: Identify Occluded Anchors

1. **Single Violation:**

- If only one anchor violates triangle inequality: likely occluded

- Confidence level: High (assuming odometry is reliable)

2. **Multiple Violations:**

- Check odometry reliability (Step 3 validation)

- If odometry is reliable: multiple occlusions likely

- If odometry is unreliable: suspend occlusion detection until odometry recovers

3. **No Violations:**

- All anchors consistent with motion

- High confidence in LOS conditions

Step 7: Temporal Filtering

1. Apply temporal smoothing to reduce false positives:

- Maintain sliding window of last N measurements (recommended: $N = 5$)

- Require violation persistence across multiple time steps

- Use majority vote across temporal window

2. **Occlusion confirmation criteria:**

- Anchor must violate triangle inequality in ≥60% of recent measurements

- Violation magnitude must exceed $2 \times \epsilon_{odom}$ for high confidence

Combined Detection Algorithms

Combining geometric and odometry-based detection provides robust occlusion identification with reduced false positive rates.

Fusion Detection Algorithm

Step 1: Parallel Detection

1. Execute geometric detection algorithm (Section 3)
2. Execute odometry-based detection algorithm (Section 4)
3. Record results for each anchor:

- geo_occluded_i: Boolean result from geometric test

- odom_occluded_i: Boolean result from odometry test

Step 2: Confidence Weighting

1. Assign confidence weights based on system conditions:

- w_{geo}: Geometric detection weight (0.7 for good anchor geometry, 0.3 for poor)

- w_{odom}: Odometry detection weight (0.8 for reliable odometry, 0.2 for unreliable)

2. Ensure weights sum to unity: $w_{geo} + w_{odom} = 1.0$

Step 3: Fusion Decision

1. Calculate combined occlusion probability for each anchor:

$$P(\text{occluded}_i) = w_{geo} \cdot \text{geo\_occluded}_i + w_{odom} \cdot \text{odom\_occluded}_i$$

2. Apply threshold decision:

$$\text{final_occluded}_i = \begin{cases}

\text{True} & \text{if } P(\text{occluded}_i) > 0.6 \\

\text{False} & \text{otherwise}

\end{cases}$$

Step 4: Consistency Validation

1. **Agreement cases** (both methods agree):

- High confidence in result

- Proceed with anchor exclusion/inclusion as indicated

2. **Disagreement cases** (methods disagree):

- Lower confidence threshold to 0.4 for conservative approach

- Implement additional validation checks

- Consider environmental factors (multipath, terrain complexity)

Step 5: System Health Assessment

1. **Monitor detection system performance:**

- Track false positive rates across operational time

- Adjust confidence weights based on historical accuracy

- Flag systematic issues (e.g., consistent odometry failures)

2. **Adaptive threshold adjustment:**

- Increase sensitivity in complex terrain

- Decrease sensitivity in open environments

- Adjust based on rover speed and mission criticality

Mathematical Foundations

Geometric Convergence Theory

For n anchors in perfect LOS conditions, the geometric intersection should satisfy:

2D Case:

$$\max_{i,j} \left| \sqrt{(x_{convergence} - x_i)^2 + (y_{convergence} - y_i)^2} - r_i \right| \leq \epsilon_{geo}$$

3D Case:

$$\max_{i} \left| \sqrt{(x_{convergence} - x_i)^2 + (y_{convergence} - y_i)^2 + (z_{convergence} - z_i)^2} - r_i \right| \leq \epsilon_{geo}$$

Error Propagation in Occlusion Detection

Geometric Detection Error Sources:

1. **ToF measurement uncertainty:** σ_{ToF} (see ToF documentation)
2. **Anchor position uncertainty:** σ_{anchor}
3. **Intersection calculation precision:** σ_{calc}

Combined geometric error:

$$\sigma_{geo} = \sqrt{\sigma_{ToF}^2 + \sigma_{anchor}^2 + \sigma_{calc}^2}$$

Odometry Detection Error Sources:

1. **Wheel encoder uncertainty:** σ_{wheel}
2. **Accelerometer uncertainty:** σ_{accel}
3. **Time synchronization error:** σ_{time}

Combined odometry error:

$$\sigma_{odom} = \sqrt{\sigma_{wheel}^2 + \sigma_{accel}^2 + (v \cdot \sigma_{time})^2}$$

Statistical Detection Thresholds

False Positive Rate Minimization:

$$\epsilon_{geo} = k_1 \cdot \sigma_{geo}$$

$$\epsilon_{odom} = k_2 \cdot \sigma_{odom}$$

Where k_1, k_2 are chosen based on desired false positive rate:

- $k = 2$: ~5% false positive rate
- $k = 3$: ~0.3% false positive rate
- $k = 4$: ~0.006% false positive rate

Implementation Guidelines

Real-Time Performance Requirements

Computational Complexity:

- **2D Geometric Detection:** $O(n^2)$ for n anchors
- **3D Geometric Detection:** $O(n^3)$ for least-squares solution
- **Odometry Detection:** $O(n)$ for n anchors
- **Combined Detection:** $O(n^3)$ dominated by geometric component

Timing Budget Allocation:

- Maximum allowable detection time: 10% of measurement cycle
- For 1 Hz position updates: ≤100 ms detection time
- Recommended: <50 ms for real-time responsiveness

Anchor Configuration Optimization

Geometric Dilution of Precision (GDOP) Considerations:

1. **Optimal anchor spacing:** Avoid clustering anchors
2. **Altitude diversity:** In 3D systems, vary anchor heights
3. **Redundancy planning:** Deploy >minimum required anchors when possible
4. **Dynamic repositioning:** Move anchors when persistent occlusion detected

System Integration

Interface with Multilateration System:

1. Provide binary occlusion flags for each anchor
2. Supply confidence levels for decision weighting
3. Maintain anchor exclusion/inclusion lists
4. Update position calculations in real-time

Interface with ToF System:

1. Access raw distance measurements and timestamps
2. Monitor measurement quality indicators
3. Coordinate timing with precision clock systems
4. Handle communication system synchronization

Interface with Navigation System:

1. Receive odometry data streams
2. Synchronize timing across subsystems
3. Validate motion consistency
4. Provide occlusion status for path planning

Validation and Testing

Simulation Testing:

1. **Synthetic occlusion scenarios:** Test known occlusion patterns
2. **Noise injection:** Validate robustness to measurement errors
3. **Edge case analysis:** Test minimum anchor configurations
4. **Performance benchmarking:** Verify real-time requirements

Field Testing:

1. **Controlled occlusion tests:** Use artificial obstacles
2. **Natural terrain validation:** Test in target environments
3. **Multi-scenario testing:** Validate across different operational conditions
4. **Long-duration reliability:** Test system stability over extended operations

Performance Metrics:

1. **Detection accuracy:** True positive and false positive rates
2. **Response time:** Time from occlusion onset to detection
3. **System availability:** Percentage of time with valid position estimates
4. **Mission success rate:** Overall navigation performance with occlusion handling

---

This comprehensive framework provides the mathematical foundation and algorithmic structure needed to implement robust target occlusion detection in GPS-denied rover systems. The combination of geometric and odometry-based approaches ensures reliable performance across diverse operational environments while maintaining real-time processing requirements.