Appendix A: Algorithmic Expressions of Probabilistic Geometry in 3D

A.1 Introduction

This appendix provides algorithmic expressions of the properties of Probabilistic Geometry (PG) in three-dimensional space (3D). By formalizing the axioms and definitions presented in the main document into computational algorithms, we aim to facilitate practical implementations and simulations of PG concepts. This approach bridges the gap between theoretical foundations and practical applications, enabling researchers and practitioners to experiment with PG in computational geometry, computer graphics, robotics, and related fields.

A.2 Notations and Conventions

Before presenting the algorithms, we define the notations and conventions used throughout this appendix:

- **Point**: Denoted by $\mathbf{p}$ or $\mathbf{q}$, represented as a tuple of coordinates in 3D space: $\mathbf{p} = (x, y, z)$.
- **Degree of Existence**: Denoted by $\mu(p)$, a real number in the interval $[0, 1]$.
- **Geometric Granularity Threshold**: Denoted by $\delta_{\text{VOID}}^{\text{geometric}}$, a positive real number representing the minimal meaningful distance.
- **Distance Function**: Denoted by $d(p, q)$, representing the probabilistic distance between points $\mathbf{p}$ and $\mathbf{q}$.
- **Expected Value**: Denoted by $E[\cdot]$, representing the expected value of a random variable.
- **Probability Density Function**: Denoted by $P_d(s)$, representing the probability density of distance s.
- **Degree of Geometric Distinguishability**: Denoted by $\mu_G(p, q)$, a real number in $[0, 1]$.
- **Axioms and Definitions**: References to the main document's axioms and definitions are included for consistency.

A.3 Algorithmic Representation of PG Properties

A.3.1 Points with Degrees of Existence

Definition 7.1 (Degree of Existence): In PG, each point $\mathbf{p}$ has a degree of existence $\mu(p) \in [0, 1]$, reflecting its probabilistic presence.

Data Structure:

```
Point3D:
    x: float
    y: float
    z: float
    mu: float  # Degree of existence, mu  [0, 1]
```

Assignment Strategies for $\mu(p)$:

- **Uniform Assignment**: Assign $\mu(p)$ randomly from a uniform distribution over $[0, 1]$.
- **Distance-Based Assignment**: Set $\mu(p)$ based on the point's distance from a reference point or region.
- **Application-Specific Factors**: Use sensor reliability, measurement uncertainty, or other criteria relevant to the application.

Algorithm:

Algorithm 1 Assign Degree of Existence

function ASSIGN_DEGREE_OF_EXISTENCE(p : Point3D)
    reference_point $\leftarrow (0, 0, 0)$ ▷ Origin
    max_distance $\leftarrow$ predefined_max_distance
    distance $\leftarrow$ euclidean_distance(p, reference_point)
    $\mu_p \leftarrow \max(0, 1 - (\text{distance}/\text{max\_distance}))$
    return μ_p
end function

A.3.2 Geometric Granularity Threshold

Definition 7.2 (Geometric Granularity Threshold $\delta_{\mathbf{VOID}}^{\mathbf{geometric}}$): A minimal meaningful distance $\delta_{\text{VOID}}^{\text{geometric}} > 0$, below which distinctions in spatial measurements become probabilistically indistinct.

Constant Declaration:

```
delta_void_geometric: float  # Positive real number, e.g., delta_void_geometric = 0.01
```

A.3.3 Probabilistic Distance Function

Axiom 6.3 (Probabilistic Distance Function): Distances between points are associated with probability distributions to reflect measurement uncertainty at small scales.

Clarifying Probability Distributions:

- **Uniform Distribution**: Assumes equal probability over the interval $[\delta_{\text{VOID}}^{\text{geometric}}, d_{\text{euclidean}}]$.

- **Normal (Gaussian) Distribution**: Appropriate when measurement errors are normally distributed around the Euclidean distance.
- **Exponential Distribution**: Suitable for modeling decay or attenuation effects in distances.

Algorithm to Compute Expected Distance:

Algorithm 2 Expected Distance

function EXPECTED_DISTANCE(p : Point3D, q : Point3D)
    $d_{\text{euclidean}} \leftarrow \sqrt{(p.x - q.x)^2 + (p.y - q.y)^2 + (p.z - q.z)^2}$
    if $d_{\text{euclidean}} < \delta_{\text{VOID}}^{\text{geometric}}$ **then**
        expected_d $\leftarrow \delta_{\text{VOID}}^{\text{geometric}}$
    else
        $\sigma \leftarrow$ measurement_uncertainty
        expected_d $\leftarrow$ integrate_normal_distribution(mean $= d_{\text{euclidean}}$, sigma $= \sigma$, lower $= \delta_{\text{VOID}}^{\text{geometric}}$)
    end if
    return expected_d
end function

Impact of Distribution Choice:

- **Uniform Distribution**: Simpler computation, assumes equal likelihood within the interval.
- **Normal Distribution**: Reflects real-world measurement uncertainties, provides a more accurate expected distance.

A.3.4 Degree of Geometric Distinguishability

Axiom 6.4 (Degree of Geometric Distinguishability):

$$\mu_G(p, q) = \frac{1}{1 + \frac{E[d(p,q)]}{\delta_{\text{VOID}}^{\text{geometric}}}}$$

Algorithm:

Algorithm 3 Degree of Distinguishability

function DEGREE_OF_DISTINGUISHABILITY(p : Point3D, q : Point3D)
    expected_d $\leftarrow$ expected_distance(p, q)
    $\mu_G \leftarrow 1/(1 + (\text{expected\_d}/\delta_{\text{VOID}}^{\text{geometric}}))$
    return μ_G ▷ $\mu_G \in (0, 1]$
end function

Interpretation:

- As $E[d(p,q)]$ approaches $\delta_{\text{VOID}}^{\text{geometric}}$, $\mu_G(p,q)$ approaches 0.5, indicating increased indistinguishability.
- For large expected distances, $\mu_G(p,q)$ approaches 0, meaning the points are easily distinguishable.

A.3.5 Probabilistic Metric Space

Axiom 6.5 (Probabilistic Metric Space): A geometric space **G** equipped with a probabilistic distance function forms a probabilistic metric space.

Properties:

1. **Non-negativity**: $P_d(s) = 0$ for $s < \delta_{\text{VOID}}^{\text{geometric}}$.
2. **Identity of Indiscernibles**: For any point $\mathbf{p}$, $P_d(p,p,s) = 0$ for $s < \delta_{\text{VOID}}^{\text{geometric}}$.
3. **Symmetry**: $P_d(p,q,s) = P_d(q,p,s)$ for all $\mathbf{p}, \mathbf{q} \in \mathbf{G}$.

Algorithmic Representation:

- **Distance Distribution Function**:

Algorithm 4 Distance Distribution

```
function DISTANCE_DISTRIBUTION(p : Point3D, q : Point3D)
    if p == q then
        distribution ← dirac_delta(s = δ_VOID^geometric)
    else
        distribution ← define_distribution(mean = d_euclidean, sigma = measurement_uncertainty, lower = δ_VOID^geometric)
    end if
    return distribution
end function
```

- **Symmetry Check**:

Algorithm 5 Symmetry Check

```
assert distance_distribution(p, q) == distance_distribution(q, p)
```

A.4 Implementation Considerations

A.4.1 Representation of Distributions

- **Discrete Representations**: Use histograms or probability mass functions for computational efficiency.
- **Continuous Representations**: Utilize analytical expressions or numerical methods for probability density functions (PDFs).

A.4.2 Handling $\delta_{\mathbf{VOID}}^{\mathbf{geometric}}$

- **Thresholding**: Enforce $\delta_{\text{VOID}}^{\text{geometric}}$ as the minimal distance in computations.
- **Normalization**: Ensure that PDFs integrate to 1 over their support.

A.4.3 Numerical Stability and Precision

- **Avoid Division by Zero**: Ensure $\delta_{\text{VOID}}^{\text{geometric}} > 0$ to prevent undefined operations.
- **Floating-Point Precision**: Use appropriate data types (e.g., 'double' precision) to maintain accuracy.

A.4.4 Algorithmic Complexity and Scalability

- **Computational Complexity**: The complexity of computing expected distances and degrees of distinguishability is $O(N^2)$ for N points.
- **Optimizations**:
  - **Spatial Partitioning**: Use spatial data structures like k-d trees or octrees to reduce the number of point comparisons.
  - **Thresholding Distances**: Ignore computations for points beyond a certain distance to focus on relevant interactions.
- **Parallelization**: Utilize parallel computing techniques to distribute computations across multiple processors.

A.5 Practical Applications and Use Cases

A.5.1 Robotics and Navigation

- **Probabilistic Localization**: Robots can use PG to model uncertainties in their position and the environment.
- **Sensor Fusion**: Combining data from multiple sensors with different degrees of reliability.

Example Algorithm:

A.5.2 Computer Graphics

- **Rendering with Uncertainty**: PG can model scenes where object boundaries are not sharply defined.
- **Level of Detail (LOD)**: Adjust object representations based on the observer's distance and viewing angle.

Example Algorithm:

Algorithm 6 Update Robot Position

```
function UPDATE_ROBOT_POSITION(sensor_data)
    for data_point ∈ sensor_data do
        point ← create_point(data_point.x, data_point.y, data_point.z)
        point.μ ← sensor_reliability(data_point.sensor_type)
                                ▷ Incorporate point into PG space for localization
    end for
end function
```

Algorithm 7 Render Scene

```
function RENDER_SCENE(observer_position, objects)
    for object ∈ objects do
        distance ← euclidean_distance(observer_position, object.position)
        object.μ ← max(0, 1 − (distance/max_render_distance))
        if object.μ > visibility_threshold then
            draw_object(object, opacity = object.μ)
        end if
    end for
end function
```

A.5.3 Spatial Data Analysis

- **Handling Imprecise Data**: PG provides a framework for analyzing spatial data with inherent uncertainties.
- **Probabilistic Clustering**: Grouping data points based on degrees of existence and probabilistic distances.

Example Algorithm:

Algorithm 8 Probabilistic Clustering

```
function PROBABILISTIC_CLUSTERING(data_points)
    for p ∈ data_points do
        p.μ ← assign_degree_of_existence(p)
    end for
    clusters ← initialize_clusters()
    for p ∈ data_points do
        assign_to_cluster(p, clusters, distance_function = expected_distance)
    end for
end function
```

A.6 Visualization and Diagrams

To aid understanding, visual representations can illustrate how PG concepts manifest in 3D space.

A.6.1 Points with Degrees of Existence

- **Visualization**: Represent points as spheres with sizes or opacities proportional to $\mu(p)$.
- **Diagram Description**: A 3D scatter plot where points vary in size and transparency based on their degrees of existence.

A.6.2 Probabilistic Distances

- **Visualization**: Show the probabilistic distance between two points as a shaded region or error bar.
- **Diagram Description**: Lines connecting points with varying thickness or color intensity representing the degree of distinguishability $\mu_G(p, q)$.

A.6.3 Effect of Granularity Threshold

- **Visualization**: Display regions where points are probabilistically indistinct due to the granularity threshold.
- **Diagram Description**: Overlapping spheres or blurred areas indicating indistinguishable points.

Implementation Note:

- Use visualization libraries (e.g., Matplotlib, VTK) to create interactive 3D plots.
- Adjust visual properties (e.g., color maps) to reflect probabilistic values.

A.7 Example Algorithms

A.7.1 Computing Expected Distance with Normal Distribution

Algorithm 9 Expected Distance with Normal Distribution

function EXPECTED_DISTANCE_NORMAL(p : Point3D, q : Point3D)
    $d_{\text{euclidean}} \leftarrow$ euclidean_distance(p, q)
    if $d_{\text{euclidean}} < \delta_{\text{VOID}}^{\text{geometric}}$ **then**
        expected_d $\leftarrow \delta_{\text{VOID}}^{\text{geometric}}$
    else
        mean $\leftarrow d_{\text{euclidean}}$
        $\sigma \leftarrow$ measurement_uncertainty
        expected_d $\leftarrow$ truncated_normal_mean(mean, σ, lower $= \delta_{\text{VOID}}^{\text{geometric}}$)
    end if
    return expected_d
end function

A.7.2 Efficient Degree of Distinguishability Computation

- **Optimization**: Precompute values for common distances or use lookup tables.
- **Algorithm**:

Algorithm 10 Efficient Degree of Distinguishability

```
function DEGREE_OF_DISTINGUISHABILITY_OPTIMIZED(p : Point3D, q : Point3D)
    distance_key ← quantize_distance(p, q)
    if distance_key ∈ lookup_table then
        μ_G ← lookup_table[distance_key]
    else
        expected_d ← expected_distance(p, q)
        μ_G ← 1/(1 + (expected_d/δ_VOID^geometric))
        lookup_table[distance_key] ← μ_G
    end if
    return μ_G
end function
```

A.8 Consistency with Main Document

- **Notations and Terminology**: All symbols and terms are consistent with those used in the main document.
- **References to Axioms and Definitions**: Each algorithm references the relevant axioms and definitions to reinforce the theoretical foundation.

A.9 Conclusion

The algorithmic expressions provided in this appendix offer a practical framework for implementing the concepts of Probabilistic Geometry in three-dimensional space. By translating the theoretical axioms and definitions into computational algorithms, we enable simulations and applications that explore the implications of PG in various fields.

Researchers and practitioners can use these algorithms to:

- **Model Spaces with Probabilistic Properties**: Incorporate uncertainties and degrees of existence into spatial models.
- **Handle Measurement Uncertainty**: Accommodate finite granularity and imprecision inherent in real-world data.
- **Develop Applications**: Implement PG concepts in robotics, computer graphics, spatial data analysis, and more.

By integrating these algorithmic expressions into software and systems, we move closer to realizing the full potential of Probabilistic Geometry in addressing complex spatial problems.