Representación Métrica del Entorno en Robótica Móvil

Introducción

La **representación métrica del entorno** es fundamental en robótica móvil y en sistemas de inteligencia artificial que interactúan con espacios físicos. Los **mapas métricos** capturan las propiedades geométricas del entorno, proporcionando información detallada necesaria para tareas como navegación, evitación de obstáculos, localización y mapeo (SLAM).

Enfoques Principales

Existen dos enfoques principales para la representación métrica del entorno:

1. **Rejillas de Ocupación**: Dividen el espacio en una rejilla regular de celdas, donde cada celda almacena información sobre la ocupación del espacio.
2. **Representaciones Poliédricas**: Utilizan primitivas geométricas como líneas, planos y poliedros para representar el entorno de manera más compacta.

La elección del tipo de mapa depende del objetivo específico:

- **Navegación y Evitación de Obstáculos**: Se requiere información detallada del entorno, por lo que las **rejillas de ocupación** son más adecuadas.
- **Localización o Mapping**: Interesa identificar partes significativas del entorno que sean útiles para los objetivos, donde las **representaciones poliédricas** son más eficientes.

Rejillas de Ocupación

Las **rejillas de ocupación** representan el espacio como una rejilla de grano fino donde cada celda corresponde a una ubicación específica en el entorno.

Características

- **Estados de las Celdas**:
  - **Ocupado**: La celda contiene un objeto o está bloqueada.
  - **Libre**: La celda está despejada y puede ser atravesada.
  - **Desconocido**: No se ha obtenido información sobre la celda.
- **Información Probabilística**: La probabilidad de ocupación de cada celda se actualiza mediante observaciones y mediciones de los sensores.

Modelo Probabilístico

Asumiendo que la posición del robot $x_{1:t}$ y las lecturas de los sensores $z_{1:t}$ son conocidas, la probabilidad a posteriori del mapa m se obtiene como:

$$P(m|z_{1:t}, x_{1:t}) = \prod_i p(m_i|z_{1:t}, x_{1:t})$$

Donde m_i representa la celda i-ésima del mapa.

Ventajas

- **Conocimiento Detallado**: Proporcionan una representación exhaustiva del entorno, útil para navegación precisa y planificación detallada.
- **Simplicidad de Implementación**: Fácil de entender y programar.

Desventajas

- **Alto Consumo de Memoria**: Requiere almacenar información para cada celda, lo que es ineficiente en entornos de gran tamaño o dimensiones.
- **Escalabilidad Limitada**: No es práctico para áreas extensas o entornos tridimensionales complejos.

Para abordar estas limitaciones, se utilizan estructuras de datos más eficientes como **QuadTrees** y **OcTrees**.

QuadTrees

Los **QuadTrees** son estructuras de datos en forma de árbol que permiten una división no homogénea del espacio bidimensional. Aprovechan áreas contiguas con las mismas características para reducir el uso de memoria.

Funcionamiento

- **División Recursiva**: El espacio se divide en cuatro cuadrantes (de ahí "Quad") de forma recursiva.
- **Nodos**:
    - **Nodo Hoja**: Representa una región homogénea (todas las celdas tienen el mismo estado).
    - **Nodo Interno**: Tiene cuatro hijos correspondientes a los cuadrantes superior izquierdo, superior derecho, inferior izquierdo e inferior derecho.

Construcción del QuadTree

1. **Inicialización**: Se parte de una lista de puntos 2D adquiridos por los sensores del robot.
2. **Definición de Región**: Cada nodo está delimitado por dos puntos (esquina superior izquierda y esquina inferior derecha).
3. **Evaluación de Homogeneidad**:
    - Si todos los puntos dentro de la región tienen el mismo estado, el nodo es una hoja.
    - Si hay diversidad, se subdivide el nodo en cuatro cuadrantes y se repite el proceso para cada uno.

Algoritmo Simplificado

```
QuadTree(p1, p2, listaPuntos):
    Nodo = crearNodo(p1, p2)
```

```
si mismoTipo(Nodo, listaPuntos):
    Nodo.tipo = obtenerTipo(Nodo, listaPuntos)
sino:
    pMedio = puntoMedio(Nodo)
    dividir listaPuntos en cuatro listas según pMedio
    Nodo.nodoSI = QuadTree(p1, pMedio, puntosSI)
    Nodo.nodoSD = QuadTree((pMedio.x, p1.y), (p2.x, pMedio.y), puntosSD)
    Nodo.nodoII = QuadTree((p1.x, pMedio.y), (pMedio.x, p2.y), puntosII)
    Nodo.nodoID = QuadTree(pMedio, p2, puntosID)
retornar Nodo
```

Ventajas

- **Eficiencia en Memoria**: Reduce significativamente el uso de memoria en comparación con las rejillas de ocupación convencionales.
- **Adaptabilidad**: Se adapta dinámicamente a la complejidad del entorno, subdividiendo más en áreas con mayor detalle.

Aplicaciones

- **Mapas 2D**: Ideales para representar entornos bidimensionales en robótica móvil y gráficos por computadora.

OcTrees

Las **OcTrees** son la extensión tridimensional de los QuadTrees, utilizadas para representar entornos en 3D.

Características

- **División Espacial en 3D**: Cada nodo se divide en ocho octantes (de ahí “Oc”, de “Octree”).
- **Nodos**:
    - **Nodo Hoja**: Representa un volumen en el espacio con estado homogéneo.
    - **Nodo Interno**: Tiene ocho hijos correspondientes a los subvolúmenes.

Uso en Robótica Móvil

- **Mapeo 3D**: Permite almacenar mapas tridimensionales obtenidos de sensores de rango 3D, como LIDAR o cámaras estéreo.
- **Eficiencia**: Conserva memoria al representar áreas homogéneas sin necesidad de almacenar cada punto individualmente.

Representaciones Poliédricas

Las **representaciones poliédricas** utilizan primitivas geométricas (líneas, planos, poliedros) para modelar el entorno. Son útiles cuando se requiere una representación más abstracta y menos detallada.

Características

- **Primitivas Geométricas**:
  - **Líneas y Planos**: Para entornos 2D y superficies planas.
  - **Poliedros**: Representación de volúmenes en 3D.
  - **Círculos y Cilindros**: Para objetos curvos o redondeados.
- **Construcción**:
  - Puede ser construida manualmente o mediante algoritmos de ajuste a datos sensoriales.
  - Se utilizan métodos de ajuste como **mínimos cuadrados** para encontrar las primitivas que mejor se ajustan a los datos.

Proceso de Extracción de Primitivas

1. **Recolección de Datos**: Obtenemos una nube de puntos a partir de sensores como LIDAR o cámaras estéreo.
2. **Selección de Vecindad**: Para cada punto, se define una vecindad (puntos cercanos).
3. **Ajuste de Primitivas**: Se intenta ajustar una primitiva geométrica (por ejemplo, un plano) a los puntos de la vecindad.
4. **Validación**: Se verifica si el ajuste es adecuado según un criterio de error.
5. **Construcción del Mapa**: Se generan los elementos geométricos que representan el entorno.

Ventajas

- **Reducción de Datos**: Al representar grandes conjuntos de puntos con unas pocas primitivas geométricas, se reduce la cantidad de información a manejar.
- **Adecuado para Entornos Simples**: Es eficiente en entornos donde las estructuras geométricas son predominantes (edificios, habitaciones cuadradas).

Desventajas

- **Complejidad Computacional**: El proceso de ajuste puede ser costoso, especialmente en tiempo real, ya que la complejidad es $O(k \cdot n^2)$ (donde n es el número de puntos y k es el número de vecindarios).
- **Limitación en Entornos Complejos**: Dificultad para representar entornos con formas irregulares o gran variabilidad.

Kd-Trees

Los **Kd-Trees** (K-dimensional Trees) son estructuras de datos tipo árbol binario que permiten búsquedas eficientes en espacios multidimensionales.

Características

- **Nodos**:
  - Cada nodo representa un punto en un espacio de (k) dimensiones.
  - Los nodos dividen el espacio mediante hiperplanos perpendiculares a los ejes de coordenadas.
- **División del Espacio**:
  - En cada nivel del árbol, se elige un eje de división (ciclando entre las (k) dimensiones).
  - El hiperplano de división pasa por el punto asociado al nodo y divide el espacio en dos mitades.

Construcción del Kd-Tree

La construcción del árbol se realiza de forma recursiva:

1. **Inicio**: Comenzar con una lista de puntos y una profundidad inicial (\text{depth} = 0).
2. **Selección del Eje**:
   - El eje de división es (\text{axis} = \text{depth} \mod k).
3. **Ordenamiento de Puntos**: Ordenar los puntos según el eje seleccionado.
4. **Selección del Nodo**: Elegir el punto medio como el nodo actual.
5. **División de Puntos**:
   - Los puntos antes del punto medio forman el subárbol izquierdo.
   - Los puntos después forman el subárbol derecho.
6. **Recursión**: Aplicar el proceso recursivamente para los subárboles incrementando (\text{depth}).

Algoritmo Simplificado

```
KdTree(pointList, depth):
    si pointList está vacío:
        retornar null
    axis = depth mod k
    ordenar pointList según axis
    median = punto medio de pointList
    nodo = nuevo Nodo(median)
    nodo.left = KdTree(puntos antes de median, depth + 1)
    nodo.right = KdTree(puntos después de median, depth + 1)
    retornar nodo
```

Operaciones Comunes

- **Búsqueda del Vecino Más Cercano**: Encontrar el punto más cercano a un punto dado.
- **Búsqueda por Rango**: Encontrar todos los puntos dentro de un rango especificado.

Complejidad

- **Construcción**: (O(n \log n)), donde (n) es el número de puntos.
- **Búsqueda**: En promedio (O(\log n)), aunque en el peor caso puede ser (O(n)).

Ventajas

- **Eficiencia en Búsquedas**: Permite búsquedas rápidas en espacios de alta dimensionalidad.
- **Versatilidad**: Aplicable en diversas áreas como robótica, gráficos por computadora y reconocimiento de patrones.

Aplicaciones en Robótica

- **Localización y Mapeo**: Almacenar y acceder eficientemente a puntos de interés en el entorno.
- **Planificación de Trayectorias**: Encontrar rutas óptimas evitando obstáculos representados en el espacio multidimensional.

Conclusiones

La representación métrica del entorno es crucial para que los robots móviles puedan navegar, mapear y operar de manera autónoma en entornos desconocidos o dinámicos.

Resumen de Técnicas

- **Rejillas de Ocupación**: Ofrecen una representación detallada pero pueden ser ineficientes en términos de memoria.
- **QuadTrees y OcTrees**: Mejoran la eficiencia de almacenamiento aprovechando la homogeneidad en el espacio, ideales para entornos 2D y 3D respectivamente.
- **Representaciones Poliédricas**: Simplifican el entorno en primitivas geométricas, reduciendo la complejidad pero a costa de detalles.
- **Kd-Trees**: Proporcionan una estructura eficiente para manejar y buscar puntos en espacios multidimensionales, esencial para diversas operaciones en robótica.

Elección de la Representación

La elección de la representación depende de:

- **Objetivos Específicos**: Navegación detallada vs. localización global.
- **Características del Entorno**: Complejidad, tamaño y dimensionalidad.
- **Recursos Disponibles**: Capacidad de memoria y potencia de cómputo.

Retos y Oportunidades

- **Optimización de Recursos**: Buscar un equilibrio entre detalle y eficiencia.
- **Entornos Dinámicos**: Adaptar las representaciones para manejar cambios en el entorno.
- **Integración de Técnicas**: Combinar diferentes representaciones para aprovechar sus respectivas ventajas.