Tema 9. Representación métrica del entorno

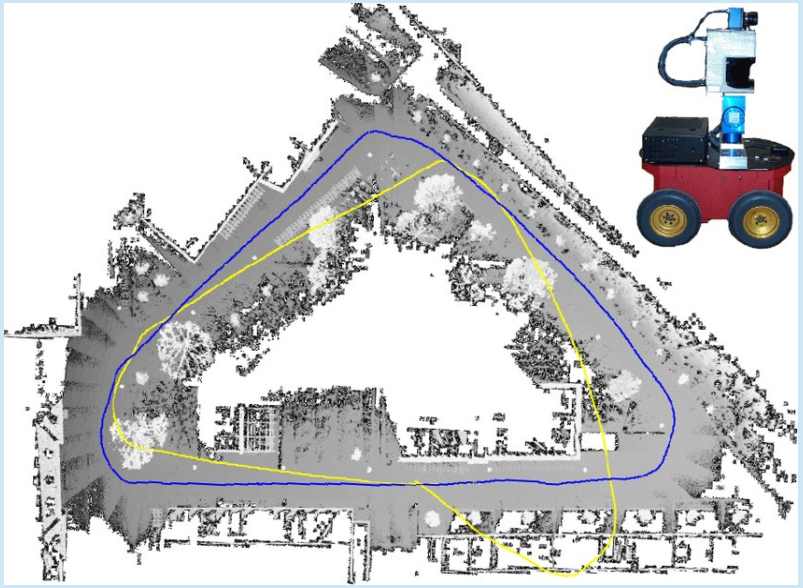
Razonamiento y Representación del Conocimiento

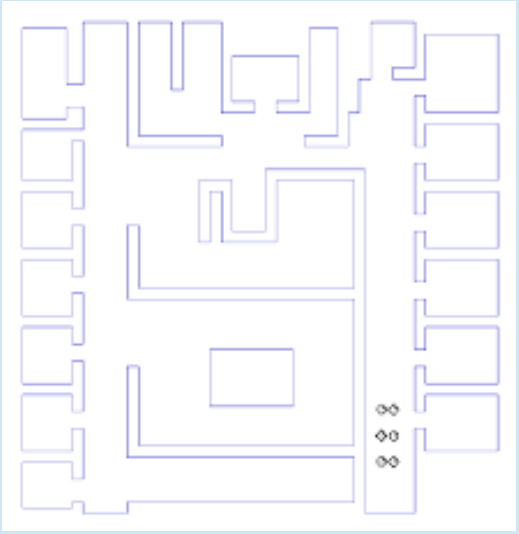
Índice

- Introducción
- Rejillas de ocupación
  - QuadTrees y OcTrees
- Representaciones Poliédricas
  - Kd-Trees

Introducción

- Mapas métricos
  - Capturan las propiedades geométricas del entorno
  - Dos enfoques
    - Rejillas de ocupación
    - Representaciones poliédricas



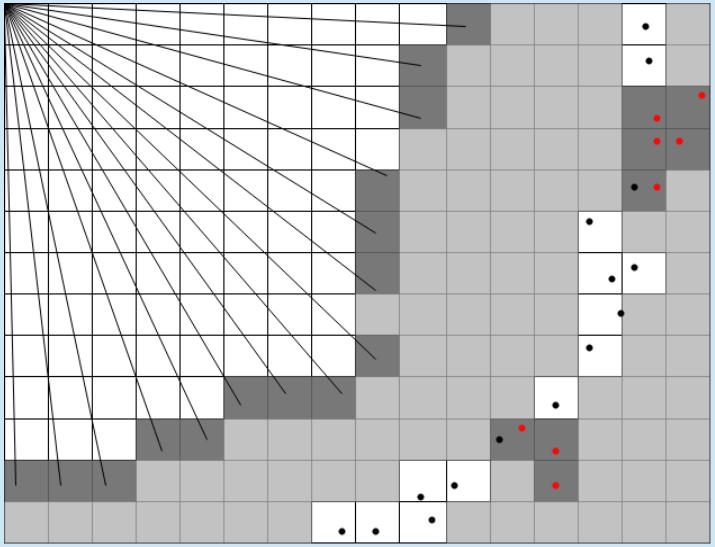


Introducción

- El tipo de mapa a utilizar depende del objetivo que busquemos:
  - Navegación y evitación de obstáculos
    - Toda la información del entorno es necesaria
    - Se utilizan principalmente rejillas de ocupación
  - Localización o mapping
    - Partes del entorno que sean útiles para el objetivo
    - Representaciones poliédricas

Rejillas de ocupación

- Representan el mapa del entorno como una rejilla de grano fino sobre las localizaciones en las que puede encontrarse un robot



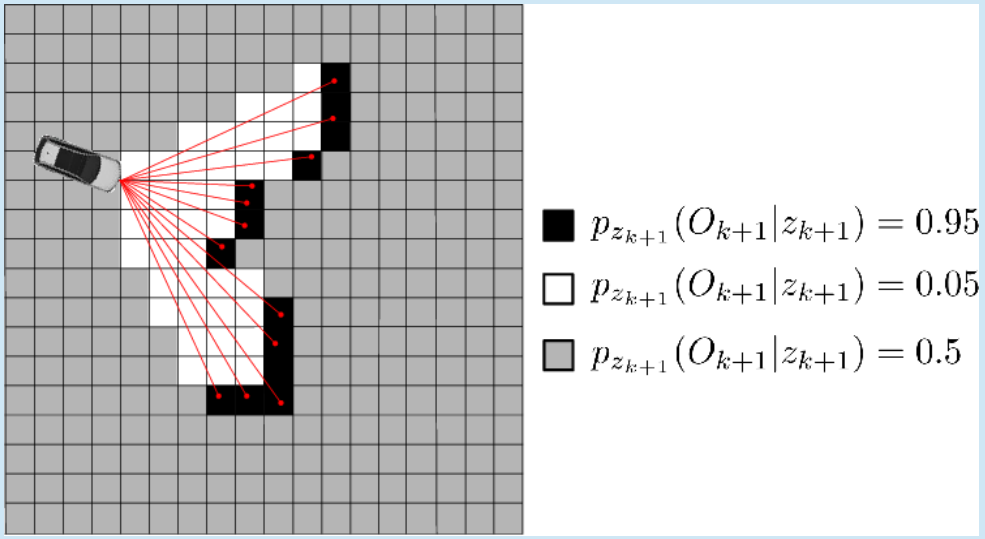
Rejillas de ocupación

- De cada posible localización:
  - Ocupado
  - Libre
  - Desconocido
- La información de si la posición en el mapa está ocupada o no es de tipo probabilístico

Rejillas de ocupación

- Representan la probabilidad a posteriori del mapa dados los datos
  - La posición del robot es conocida
  - Se conocen las lecturas de sus sensores

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



Rejillas de ocupación

- Ventajas:
  - Conocimiento detallado del entorno:
    - Navegación, planificación, localización, mapping
- Desventajas:
  - Consume enorme cantidad de memoria
  - Imposible de utilizar en entornos de gran tamaño → quadTrees u ocTrees

QuadTree

- División no homogénea del espacio
- Aprovecha grandes áreas contiguas de espacio con las mismas condiciones (libre, ocupado o desconocido) para reducir el uso de memoria
- Se utiliza para la representación de mapas 2D

QuadTree

- Funcionamiento:
  - Partimos de una lista de puntos 2D recogidos por los sensores del robot
  - Cada nodo del árbol vendrá delimitado por dos puntos
    - Esquina superior izquierda
    - Esquina inferior derecha
  - Un nodo del árbol será una hoja si todo el espacio delimitado por el nodo tiene el mismo estado (ocupado, libre o desconocido)
  - Un nodo genérico tendrá cuatro descendientes correspondientes a los cuartos superior izquierdo, superior derecho, inferior izquierdo, inferior derecho

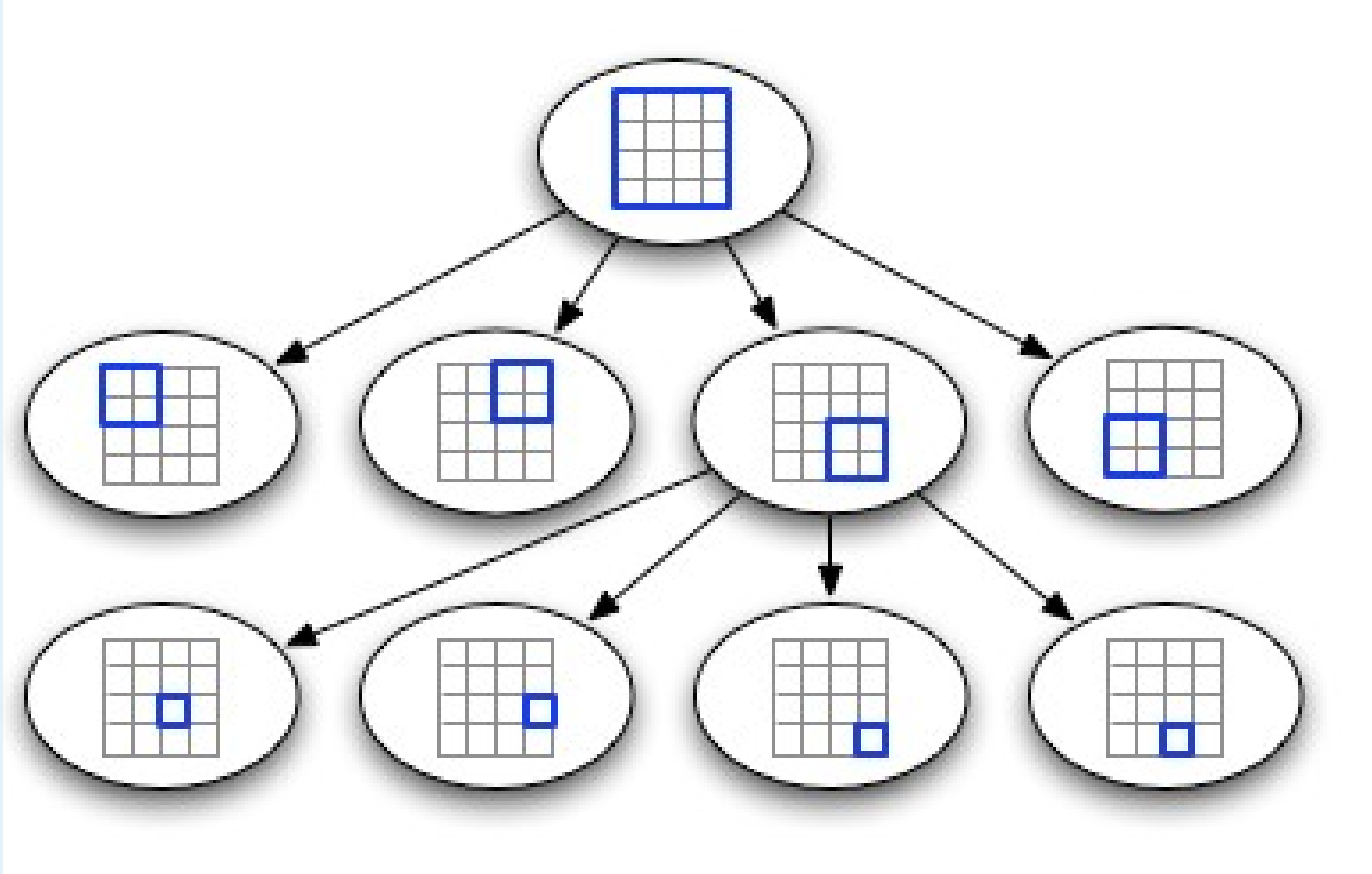
QuadTree

- Construcción

```
1.  QuadTree(p1, p2, listaPuntos): Nodo
2.    Nodo ret(p1, p2)
3.    si mismoTipo(ret, listaPuntos)
4.      ret.tipo=getType(ret, listaPuntos)
5.    sino
6.      pMedio = getMiddlePoint(ret)
7.      puntosSI, puntosSD, puntosII, puntosID = dividirPuntos(listaPuntos, pMedio)
8.      ret.nodoSI = QuadTree(p1, pMedio, puntosSI)
9.      ret.nodoSD = QuadTree(Punto(pMedio.x, p1.y), Punto(p2.x, pMedio.y), puntosSD)
10.     ret.nodoII = QuadTree(Punto(p1.x, pMedio.y), Punto(pMedio.x, p2.y), puntosII)
11.     ret.nodoID = QuadTree(pMedio, p2)
12.   return ret
```

QuadTree

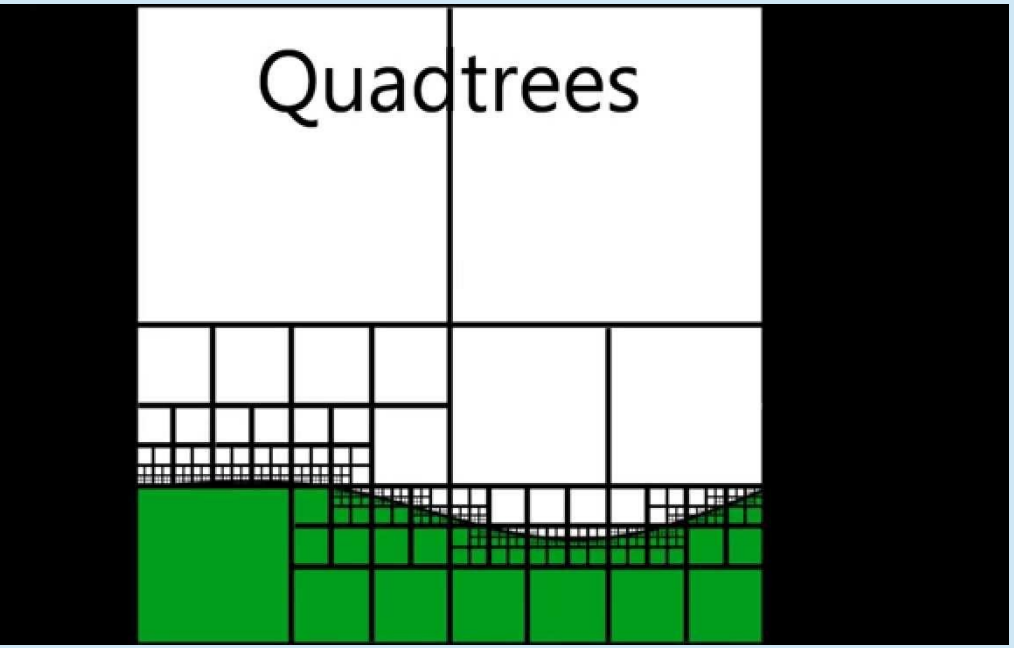
- Ejemplos

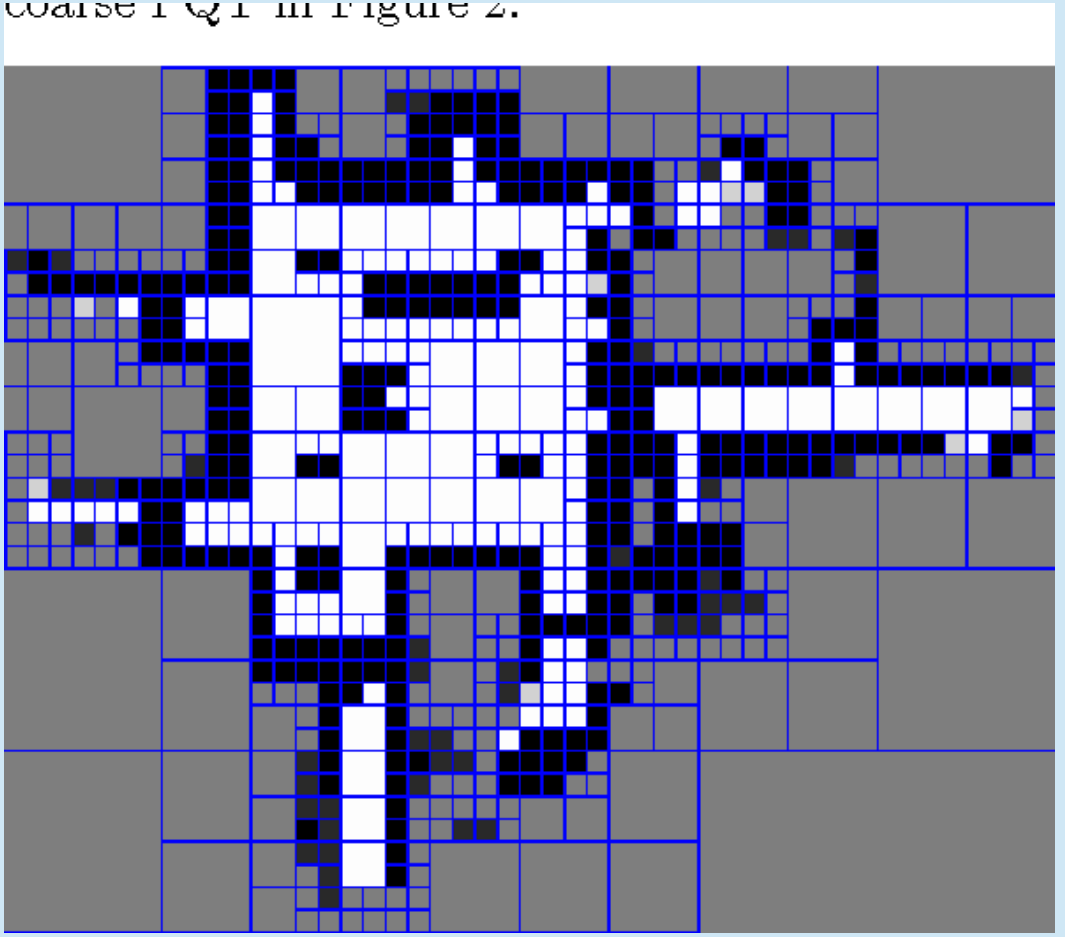


Fuente: Pradeep Pujari

QuadTree

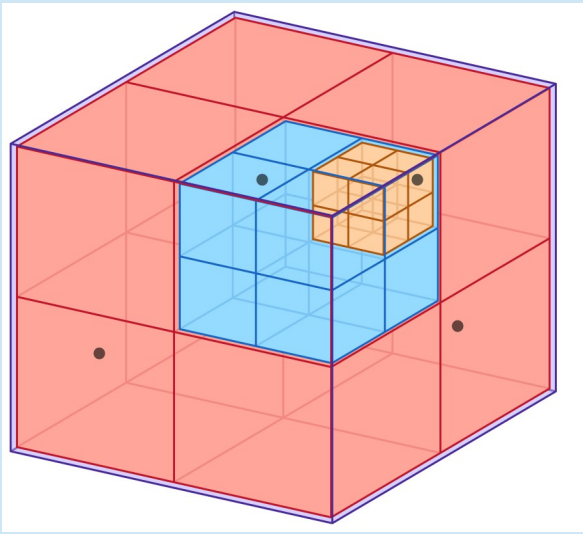
- Ejemplos





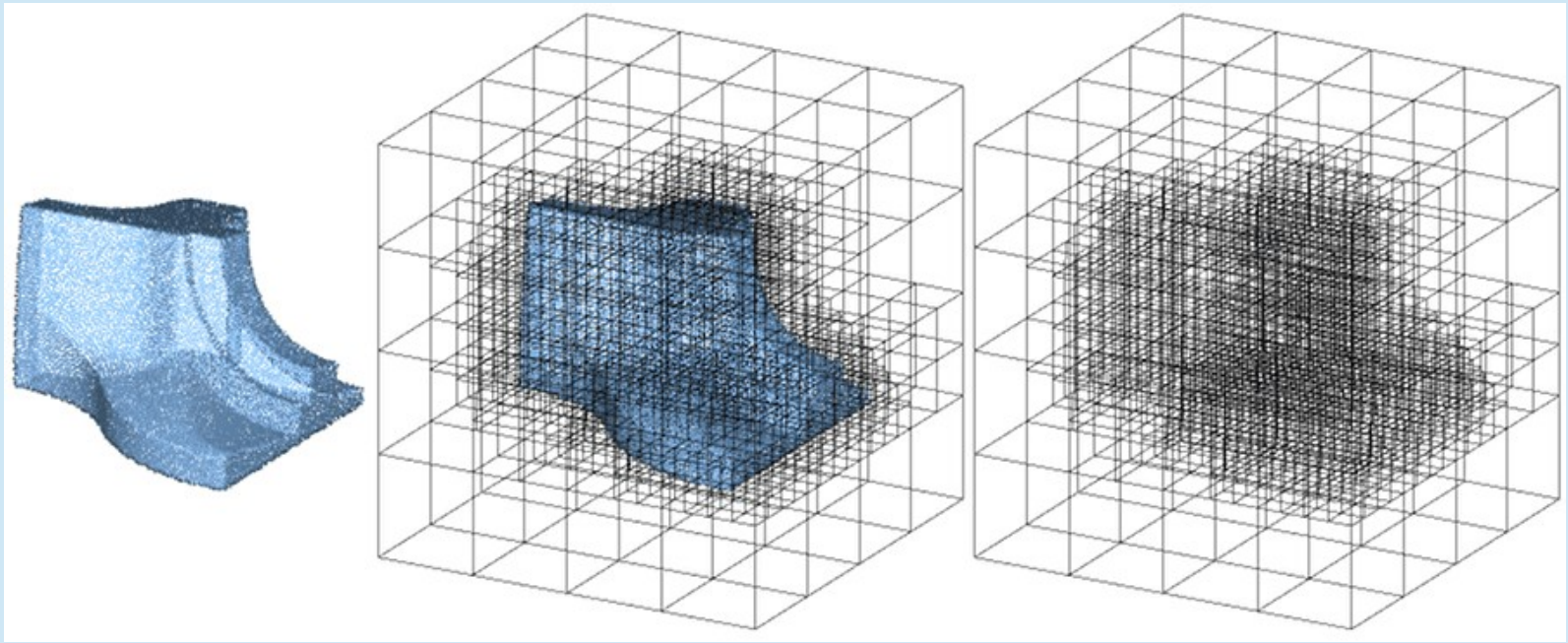
OcTree

- Estructura de datos tipo árbol
- Similar al QuadTree, pero en 3D
  - Cada nodo del árbol tendrá 8 descendientes en lugar de 4
- Permite almacenar una rejilla de ocupación con datos obtenidos mediante sensores de rango en 3D



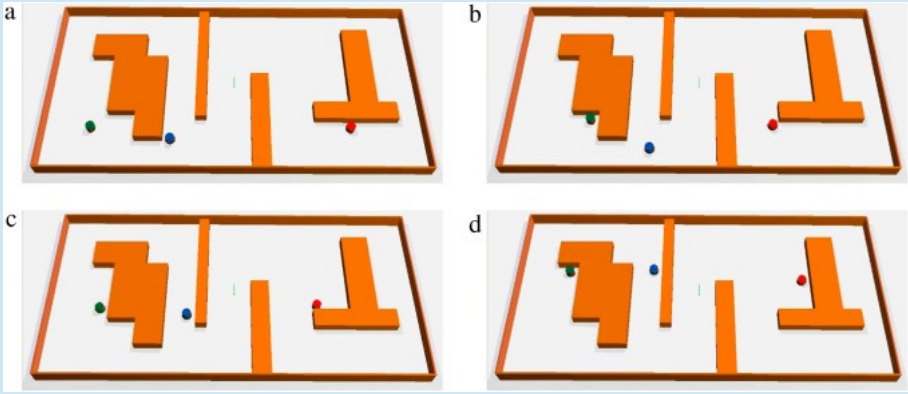
OcTree

- Ejemplo



Representaciones Poliédricas

- El mapa de entorno estará formado por:
  - Líneas
  - Planos
  - Poliedros
  - Círculos/cilindros



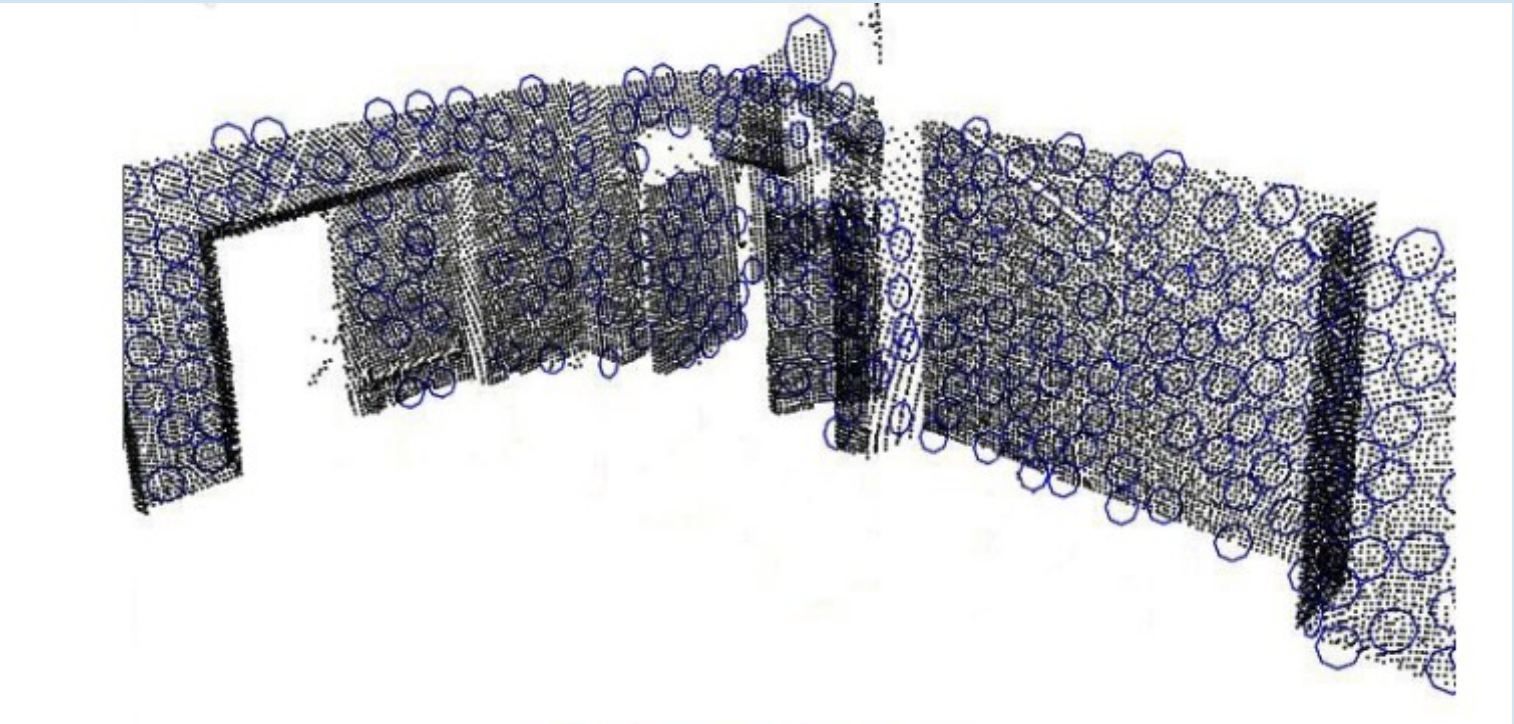
- Se construye manualmente o se obtiene mediante algoritmos que buscan estas primitivas a partir de las nubes de puntos obtenidas por los sensores del robot

Representaciones Poliédricas

- Extracción de primitivas
  - Ajuste de las primitivas a los datos utilizando mínimos cuadrados
  - Procedimiento:
    - Para cada punto
      - Obtener su vecindad
      - Comprobar si la primitiva se ajusta a los datos
    - Bastante costoso $O(k \cdot n^2)$ no es posible de ejecutarlo en tiempo real para valores de n grandes

Representaciones Poliédricas

- Ejemplo

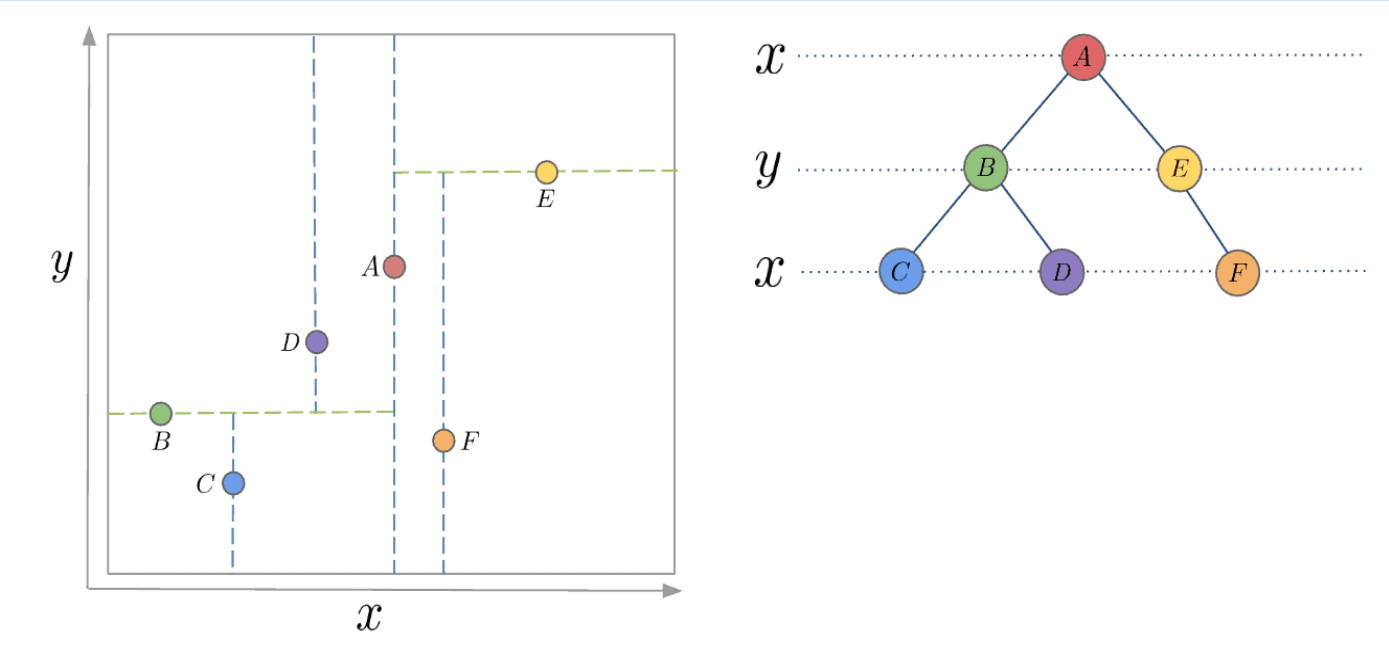


Kd-Tree

- Estructura de datos del tipo árbol binario
- Permiten la búsqueda eficiente de puntos en un espacio multidimensional:
  - Búsqueda del vecino más cercano
  - Búsqueda de todos los puntos en un rango

Kd-Tree

- ¿Qué es?
  - Cada nodo representa un punto en k-D
  - Cada nodo representa una partición del espacio en 2 mediante un hiperplano
  - Cada nivel del árbol realiza la división en un eje distinto



Kd-Tree

- Construcción

```
algorithm KdTree(pointList, depth): Node
    // INPUT
    //    pointList = a list of points
    //    depth = an integer indicating the current depth in the tree
    // OUTPUT
    //    The k-d tree rooted at the median point of pointList
    // Select the axis based on depth so that axis cycles through all valid values
    axis <- depth mod k
    Sort pointList
    // Choose median as pivot element
    median <- select median by axis from pointList
    // Create node and construct subtree
    node.location <- median
    node.leftChild <- KdTree(points in pointList before median, depth + 1)
    node.rightChild <- KdTree(points in pointList after median, depth + 1)
    return node
```

Kd-Tree

- Complejidad
  - Construcción: O(n·log(n))
  - Búsqueda: O(log(n))