Notes from chap 5 RVC v2

NAVIGATION

- All about reactive navigation algorithms

La **navegación robótica** es el proceso de dirigir un robot hacia un objetivo. Este objetivo puede definirse con base en características detectables del entorno, como una fuente de luz, o mediante coordenadas geométricas en un mapa.  
A diferencia del enfoque humano —que implica crear mapas y colocar señales— en robótica no siempre es necesario representar explícitamente el entorno. De hecho, muchos robots pueden cumplir su tarea utilizando únicamente estrategias de **navegación reactiva**, que se basan en percibir el entorno y responder directamente a lo que se detecta.  
Un ejemplo clásico de este principio es **Elsie**, la tortuga robótica temprana que se movía en dirección a una fuente de luz y evitaba obstáculos sin tener una representación interna del entorno ni un plan definido.

  
Este tipo de comportamiento también se observa en robots modernos como el **iRobot® Roomba**, que navega dentro de habitaciones sin mapas, simplemente mediante respuestas locales a sensores, como se muestra en la fotografía de larga exposición de las sig figuras:   


Diagrama

El contenido generado por IA puede ser incorrecto.

Como un robot se mueve de manera inteligente en un entorno planificado, usando diferentes estrategias y métodos de navegación, tales como el reactivo, basado en mapas, planificación de caminos y mapeo con sensores.

- En navegación robótica se distinguen **dos grandes enfoques**:

* **Basado en mapa (map-based):** el robot tiene un modelo interno del entorno y planifica rutas.
* **Reactivo (reactive):** el robot responde de manera inmediata a sus sensores, sin necesidad de saber dónde está.

Como figura destacada en los inicios de la **cibernética**, Walter diseñó y construyó algunos de los primeros **robots autónomos**, con el objetivo de explorar cómo comportamientos complejos pueden surgir de simples redes neuronales.

Su robot más famoso, **Elsie** (especie *Machina Speculatrix*), fue creado en 1948. Era un robot de tres ruedas con capacidad de **fototaxis** (movimiento hacia una fuente de luz) y autonomía básica, ya que podía incluso buscar su estación de recarga.

- **Elsie** es un ejemplo clásico de sistema reactivo inspirado en la biología:

* Su arquitectura está conectada de tal forma que la entrada sensorial (luz) se traduce directamente en salida motora.
* No hay planificación deliberada ni localización, solo respuesta estímulo-acción

- Esto se llama **comportamiento emergente**: aunque el robot no planea, el resultado parece inteligente.

Por el contrario, el robot **Shakey** desarrollado en los años 60 (Fig. 5.1b), implementaba un enfoque radicalmente distinto: poseía percepción tridimensional y era capaz de crear un mapa del entorno. A partir de dicho mapa, razonaba para planificar un camino hasta su objetivo.

Este método basado en mapas representa un enfoque más cercano al que utilizan los seres humanos y que hoy en día emplean los **pilotos automáticos**, **vehículos autónomos** y **robots móviles inteligentes**. Este proceso se denomina **planificación de trayectorias** o **planificación del movimiento**.

Sin embargo, a pesar de que permite abordar tareas complejas, este método es también más exigente, ya que requiere:

* Un mapa del entorno.
* Conocer en todo momento la ubicación exacta del robot respecto a dicho mapa.

Ambas condiciones representan retos importantes, que se detallan posteriormente en el Capítulo 6.

5.1 reactive navigation

En resumen, la navegación reactiva y la navegación basada en mapas son dos extremos opuestos de un espectro de técnicas de navegación móvil:

* Los sistemas **reactivos** son rápidos y simples: vinculan directamente la percepción con la acción, sin necesidad de almacenamiento de mapas ni razonamiento.
* Los sistemas **basados en mapas** requieren mayor capacidad de cómputo y memoria, pero permiten resolver tareas más complejas.
* Esta división también se observa en la naturaleza: los **insectos** siguen estrategias reactivas simples, mientras que **mamíferos** (como humanos) construyen y razonan sobre representaciones mentales del entorno.

5.1.1 the braitenberg vehicle ( the most simple robot)

Los **vehículos de Braitenberg** representan una clase muy simple de robots orientados a objetivos. Se caracterizan por:

* Una **conexión directa entre sensores y motores**.
* La ausencia de un modelo interno del entorno.
* No utilizan un algoritmo de planificación explícito (aunque la conexión sensorial-actuadora puede considerarse un “plan implícito”).
* Uso de algoritmos clásicos como :
  + Potential Fields
  + Vector Field Hisogram
* Implemetancion del path of the braitenberg vehicle

Casos de uso :   
Se puede usar en un robot en 2D que busca el máximo de un campo escalar. Siendo este unb caso práctico de navegación reactiva hacia una fuente.

Se configura mediante:

* Dos sensores puestos de forma bilateral, uno a cada lado
  + Permitiendo que el robot gire en dirección de donde recibe el estímulo más fuerte.

Su modelo matemático se realiza mediante un campo escalar definido por la función sensorfield t.q

Texto

El contenido generado por IA puede ser incorrecto.

Esta función representa un **campo escalar** generado por un sensor (o una fuente) que está ubicado en el punto (60, 90). El campo disminuye con la distancia desde ese

punto.

De esta forma, el robot tendrá su velocidad y angulo de giro, definidos de la sigueinbte manera:

* \* v=2⋅sR​⋅sL​
* ω=Ks​⋅(sR​−sL​)

de donde inferimos que, si sR = sL, el robot debería avanzar en linea recta.

Interfaz de usuario gráfica, Aplicación

El contenido generado por IA puede ser incorrecto.

La gráfica muestra la trayectoria del vehículo Braitenberg en el plano XY, indicando cómo se mueve desde su posición inicial hacia el máximo del campo sensorial. Esta trayectoria es el resultado del comportamiento del vehículo al seguir el gradiente del campo, utilizando las lecturas de sus sensores izquierdo (sL) y derecho (sR).

Entendemos entonces que, el comportamiento generado depende completamente de la función de conexión entre sensores y motores.

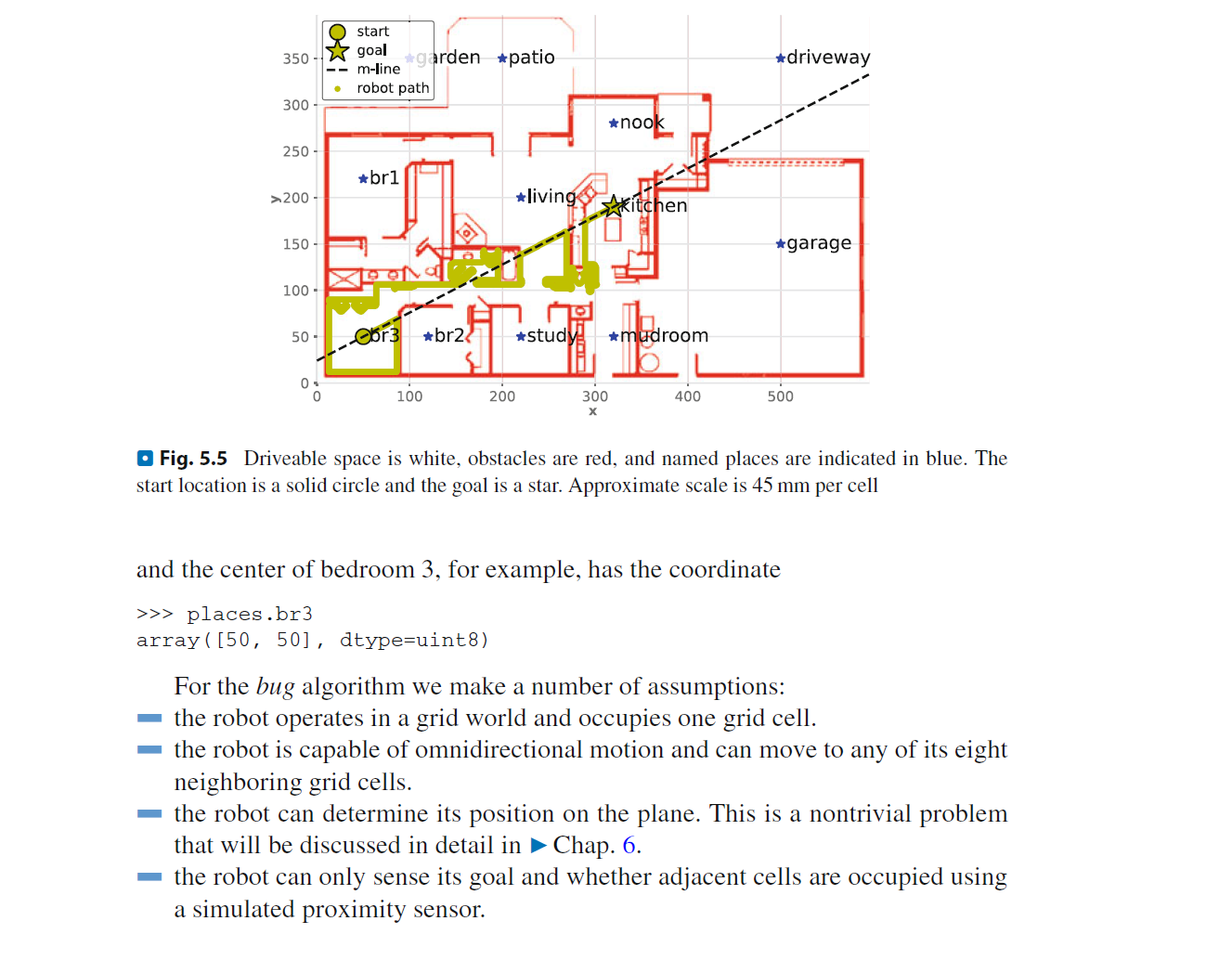
**Limitación:** El robot **no puede esquivar obstáculos**, ya que su única conducta es ir hacia el estímulo.

**Solución:** Añadir nuevas conductas, como evitar obstáculos o buscar aleatoriamente en ausencia de estímulo → esto nos lleva al concepto de **robótica basada en comportamientos** (behavior-based robotics).

5.1.2 Simple automata

En esta sección se introduce una **clase de robots reactivos** llamada **"bugs"**, que están diseñados para moverse hacia un objetivo mientras **esquivan obstáculos**, **sin tener un mapa** del entorno.

A diferencia de los vehículos de Braitenberg, los "bugs" **tienen una memoria mínima** y una **lógica de control más compleja**, por ejemplo, máquinas de estados.

example of an occupancy grid

* + Bug algorithms ; An reactive algorithm with strategy implemented

Texto

El contenido generado por IA puede ser incorrecto.

**Idea principal:**

1. **Dibuja una línea recta (llamada "m-line")** desde el punto de inicio al objetivo.

Diagrama

El contenido generado por IA puede ser incorrecto.

1. El robot intenta **moverse a lo largo de esa línea**.
2. Si encuentra un obstáculo, **rodea el obstáculo por la derecha**, siempre manteniéndose cerca del borde.
3. Cuando vuelve a intersectar la m-line **en un punto más cercano al objetivo que el anterior**, regresa a la m-line y sigue avanzando hacia el objetivo.
4. **Repite este proceso hasta llegar al destino**.

Diagrama

El contenido generado por IA puede ser incorrecto.

El robot parte desde el dormitorio 3 y se dirige hacia la cocina.

Se logra ver su recorrido como una línea de puntos verdes.

De esta manera, vemos que ;

El algoritmo no usa un mapa global, por lo que:

* Puede tomar rutas subóptimas como pasar por armarios o baños
* Toma decisiones locales sin saber si son contundentes a largo plazo

Esto es bueno porque; es imple y no necesita un mapa completo pero

Es malo, debido a que puede dar vueltas innecesarias y no garantiza la mejor ruta

Interfaz de usuario gráfica, Texto

El contenido generado por IA puede ser incorrecto.

Esto indica que, el robot tomó 1307 pasos para llegar a la meta.

**Bug2** es un ejemplo clásico de robot reactivo **con lógica**, capaz de **alcanzar objetivos sin un mapa**. Sin embargo, su falta de perspectiva global lo lleva a trayectorias **más largas y menos eficientes**.

Este algoritmo es útil para entender:

* Cómo se pueden diseñar robots que tomen decisiones simples basadas en sensores.
* Qué limitaciones existen al **no tener información completa** del entorno.
* Cómo funcionan las estrategias **de navegación local**.

- all about map-based navigation algorithms

Primero, vemos que en este capitulo tratamos :

- GRAFOS , que son redes de nodos y aristas interconectados.

Los bordes del gráfico son generalmente no la ruta real entre los vértices, simplemente representan el hecho de que hay es un camino manejable entre ellos

- rillas de ocupación , que representa una matriz binaria con celdas ocupadas o libres

- y mapas hoja de ruta, cuya red está planificada sobre el espacio libre.

5.2 map- based navigation

El objetivo de esta sección fue encontrar la **MEJOR ruta entre dos puntos (A y B)** usando una representación del entorno. “Mejor” puede significar:

* La **más corta** (menor distancia).
* La **más rápida** (aunque más larga).
* La **más segura** (evitando caminos estrechos o difíciles).

**También se pueden considerar:**

* Tamaño y capacidades del robot (ej. no puede girar en ángulos cerrados).
* Características del terreno (rugosidad, pendiente, etc.).

Hay que considerar, que el robot debe ser capaz de **sentir, planear y actuar** para ser efectivo en este contexto.

Veamos los tipos de representación de mapas :  
Gráfico, Gráfico de líneas

El contenido generado por IA puede ser incorrecto.

Grafos matemáticos y mapas basados en celdas

Observamos que los grafos representan lugares y rutas mediante:

* Vértices o nodos. (e.g ciudades o habitaciones)
* Aristas o edges : que representan los caminos entre los lugares
  + - A CADA ARISTA, podemos asignarle un costo, estos pueden representar:
      * Distancias , tiempos recorridos, velocidades promedio, costos monetarios , o incluso frecuencias de lugares
  + Tipos de grafos :
    - Dirigido (un solo sentido representado con flechas)
    - No dirigido o con doble sentido
* Los grafos pueden ESTAR emebebidos en uns sistema de coordenadas , ese decir ser métricos. (ubicaciones físicas) o ser TOPOLOGICOS (e.g un mapa de metro, cuya posición exacta es irrelevante)

El robot presenta un mapa del entorno, con el cual planifica rutas óptimas.

Tipos de mapas:

* Topológicos: Composiciones (nodos y conexiones)
* Métricos : como un mapa real de coordenadas

Conceptos de localización, (MAPPING + localization )

En el libro, los grafos son manejados usando el apquete pgraph, que permite crear y manipular grafos para navegación.

Veamos los tipos más a detalle:z

1. Grafos topológicos:
   1. Para este caso, podemos ver como se enfocan en lugares reconocibles, y conexiones lógicas entre estos.
      1. E.g :
         1. El robot no necesitará saber que está en la cocina
         2. Pero sabe que si atraviesa una puerta, estará en otro lugar, como puede ser la sala
         3. Tampoco necesita saber las coordenadas exactas, solo las relaciones
            1. Ergo, esto es útil debido a que, cuando un robot no tiene GPS o mapa preciso, puede usar sensores y reconocimiento para UBICARSE.
2. Mapas basados en celdas (Occupancy grid)
   1. Consiste en una **rejilla (grid)** donde cada celda:
      1. Tiene valor **0** si está libre.
      2. Tiene valor **1** si está ocupada (obstáculo).
      3. Este modelo es sencillo y ampliamente usado en navegación de robots.
      4. El tamaño de cada celda depende de:
         1. La precisión requerida.
         2. El tamaño del área total.
   2. Cuanto más pequeñas las celdas, **más precisa** será la navegación, pero también aumentará el uso de memoria y procesamiento.

On simulation :   
from RVC3 import VehiclePolygon, Unicycle, GridMap

Diagrama

El contenido generado por IA puede ser incorrecto.

| **Concepto** | **Explicación** |
| --- | --- |
| **Mapas en robótica** | Representaciones internas del entorno para tomar decisiones de navegación. |
| **Tipos de mapas** | 1. Grafos (embebidos o topológicos)  2. Occupancy grids |
| **Grafos** | Vértices (lugares) + Aristas (conexiones) + Costos |
| **Occupancy Grid** | Matriz donde 1 = obstáculo y 0 = libre |
| **Ejemplos de uso** | Planificación de rutas, evasión de obstáculos, optimización de tiempo o distancia. |
| **Ventajas del uso de mapas** | Permite planificación más compleja y global; mayor precisión en entornos grandes. |
| **Desventajas** | Mayor complejidad computacional; requiere conocer posición y entorno. |

5.3 planning with a graph – based map (GRAPH METHOD)

Uso de nodos y aristas (o grafo)

* Presenta el uso de algoritmos de búsqueda con el fin de estimar el camino
  + Dijktra
  + A\* (estimación heurística)

Esto nos presenta limitaciones reales de movimiento, y uso de sensores en la planificación del sistema

■ Fundamentos

* Un mapa basado en grafos representa el entorno mediante:
  + **Vértices (nodos):** ubicaciones (ciudades, habitaciones, puntos clave).
  + **Aristas (edges):** rutas transitables entre estos puntos.
  + Las aristas pueden tener atributos como: distancia, tiempo, velocidad permitida, o incluso calidad del terreno.
* El grafo puede ser:
  + **Dirigido:** aristas con sentido.
  + **No dirigido:** aristas bidireccionales.
  + **Embebido:** los nodos tienen coordenadas (gráfico sobre un plano).

■ Ejemplo con Toolbox:

* Se utiliza un archivo queensland.json con ciudades de Australia.
* Se crean los nodos con add\_vertex(name=name, coord=utm).
* Se agregan rutas con add\_edge(start, end, cost=distance).

■ Búsqueda de rutas:

1. **Breadth-First Search (BFS)**

* Prioriza caminos con **menos nodos**.
* No considera la distancia real.
* Devuelve una ruta rápida pero no óptima.

2. **Uniform-Cost Search (UCS)**

* Expande el nodo con menor **costo acumulado (g(v))**.
* Devuelve la ruta de **menor distancia real**.
* Usa una cola de prioridad y explora más nodos que BFS.

3. **A\* Search**

* Usa: f(v) = g(v) + h\*(v)
  + g(v): costo acumulado.
  + h\*(v): heurística admisible (ej: distancia Euclidiana).
* Explora menos nodos que UCS.
* Más eficiente si la heurística es buena.

■ Planificación por Tiempo Mínimo

* Se reemplaza la distancia por **tiempo de viaje (distancia / velocidad)**.
* Se ajusta heuristic para reflejar tiempo mínimo usando la **velocidad máxima**.

A\* se adapta sin perder optimalidad si la heurística es admisible.

5.4 Planning with an Occupancy-Grid Map

■ Fundamentos

* El mapa es una **matriz 2D**.
  + Celdas con valor **0 = libre**.
  + Celdas con valor **1 = ocupada** (obstáculo).
* Se asume:
  + Movimiento en 8 direcciones.
  + El robot ocupa una celda.

■ Distance Transform

* Se calcula la **distancia desde cada celda hasta la meta**.
* Estrategias:
  + **L2 (Euclidiana):** incluye diagonales.
  + **L1 (Manhattan):** solo ortogonales.
* La ruta óptima desde cualquier punto es seguir la pendiente descendente en el mapa de distancias.

■ Inflado de Obstáculos

* Se agrandan los obstáculos para representar el tamaño real del robot.
* Se usa **dilatación morfológica**.
* Se evita que el robot planee rutas por pasajes estrechos.

■ D\* (Dynamic A\*)

* Variante de A\* que **permite replanificar** si el entorno cambia.
* Usa una **lista de frontera** y **costo por celda**.
* Cuando una celda cambia su costo, solo se recalcula lo necesario.
* Muy útil en entornos parcialmente conocidos.

5.5 planning with roadmaps

■ Idea General

* Separar en dos fases:
  1. **Planificación:** construir una red de caminos libres (grafo).
  2. **Consulta:** buscar una ruta entre un punto de inicio y uno final.

■ Roadmap: Skeleton y Voronoi

* Se genera una red de caminos usando:
  + **Skeletonization:** reduce el espacio libre a una red de píxeles centrales.
  + **Voronoi Diagram:** distancias a obstáculos generan líneas centrales equidistantes.
* Se extraen **intersecciones (junctions)** y se crea el grafo.

■ PRM (Probabilistic Roadmap Method)

* Algoritmo por muestreo aleatorio:
  + Se generan puntos aleatorios en el espacio libre.
  + Se conectan si la línea recta entre ellos no intersecta obstáculos.
  + Se crea un **grafo embebido**.
* Se ajusta el número de puntos (npoints) para mejorar la conectividad.

■ Consulta de Ruta

* El robot se conecta al nodo PRM más cercano.
* Se usa A\* para encontrar la ruta entre nodos.
* La consulta es **rápida**, ideal para cambiar inicio y destino sin regenerar el grafo.

■ Consideraciones

* Puntos aleatorios pueden no cubrir todo el mapa: puede requerirse reintentar.
* Rutas pueden ser menos suaves.
* Puede fallar en pasajes estrechos.