

Path planning

Resumen

- Introducción**
- Dijkstra
- A*
- A*PS
- Theta*
- S-Theta*
- Conclusiones

Introducción

- El objetivo del path planning es seleccionar una ruta lo más “buena” posible
- La ruta debe ser válida
 - No atravesar obstáculos
 - Ni exceder los límites físicos del robot
- Hay que tener en cuenta las capacidades de cómputo
- Es una tarea fundamental tanto en videojuegos como en robótica móvil

Introducción

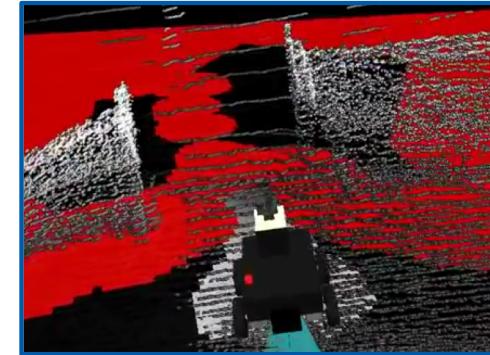
□ Navegación

- Rutas locales
- Guiado por sensores

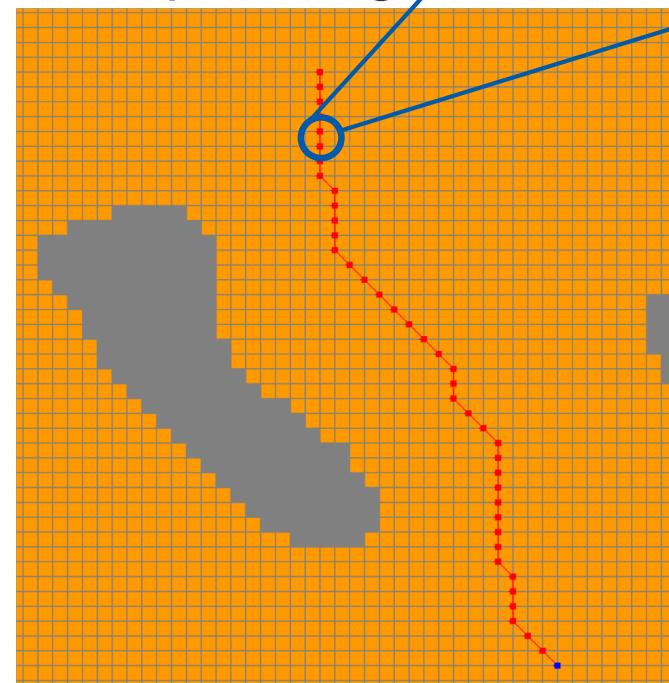
□ Planificación de rutas

- Rutas globales
- Terreno conocido
- Ligado a la IA

Navegación



Path planning



Introducción

□ El entorno:

- Rutas locales vs rutas de largo recorrido



- Totalmente observable vs parcialmente observable



- Discreto vs continuo

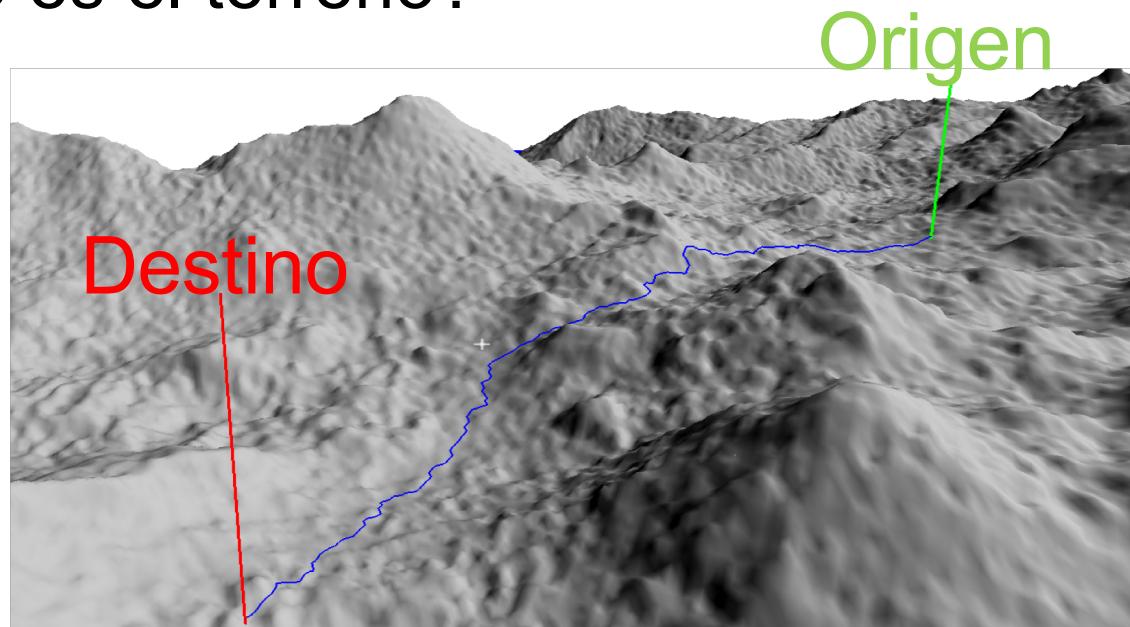


- Dinámico? Posiblemente

- ¿Información extra? Puede ser útil...

Introducción

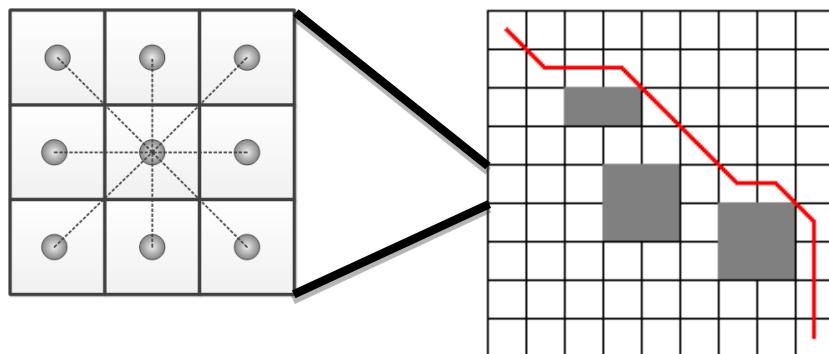
- Conseguir la ruta óptima requiere mucho esfuerzo
 - A veces, inviable
- En entornos de robótica móvil el path planning y el task planning están altamente ligados
 - ¿Planificación mixta?
- ¿Cómo es el terreno?



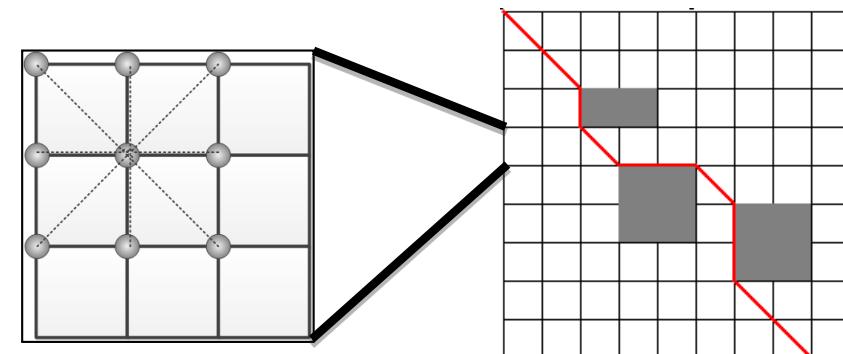
Introducción

- Entorno básico: tablero de ajedrez
 - 2D con casillas libres/bloqueadas
- Cada casilla (celda o nodo) típicamente está conectada con sus 8 vecinos
- 2 representaciones posibles:

Nodo central



Nodo vértice

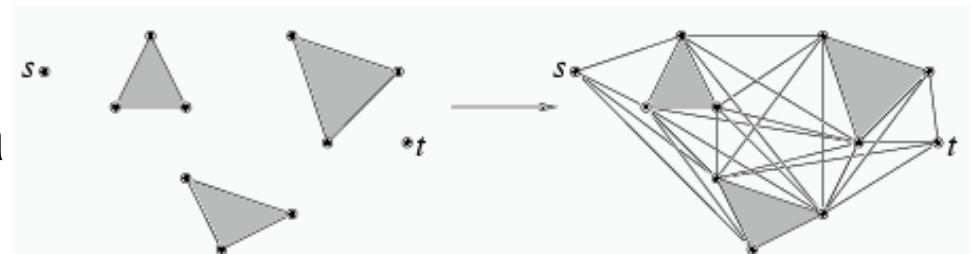
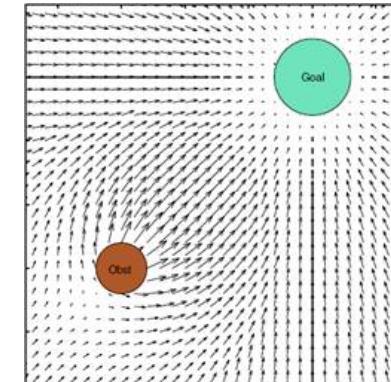
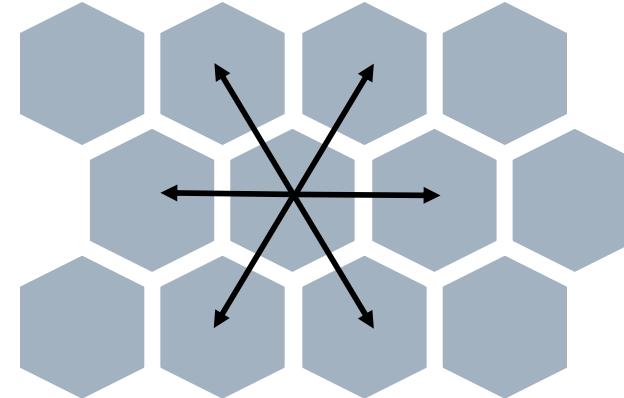


Introducción

- Celdas hexagonales
 - 6 vecinos
 - ¡Más complejas de tratar!

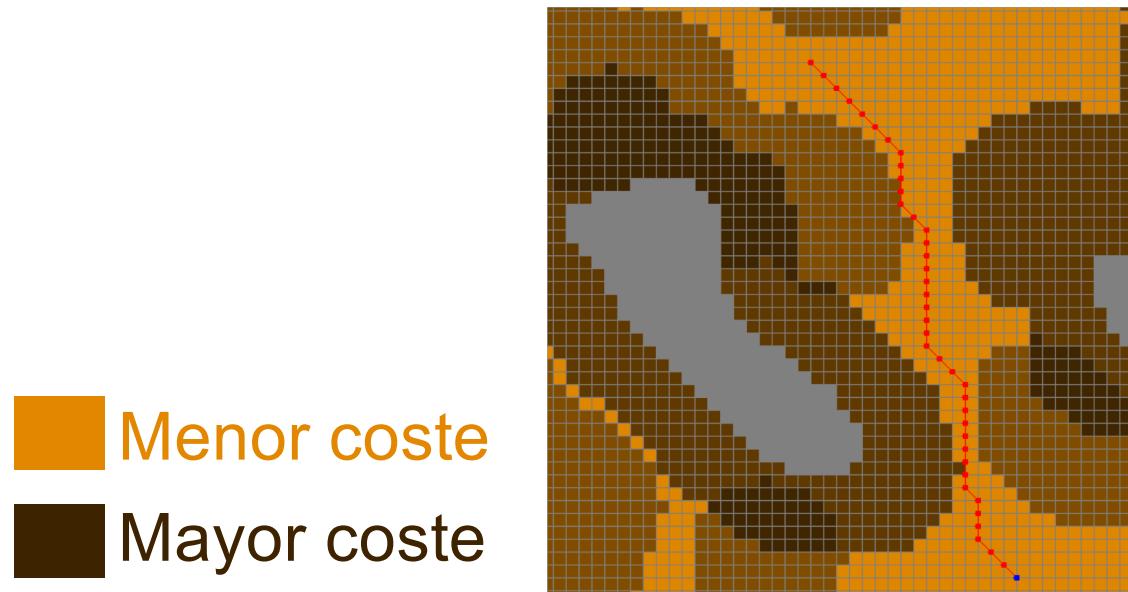
- Campos potenciales
 - Atracción/repulsión

- Grafos de visibilidad
 - Pares de puntos interconectados
 - Costosos de calcular
 - Solución rápida y óptima



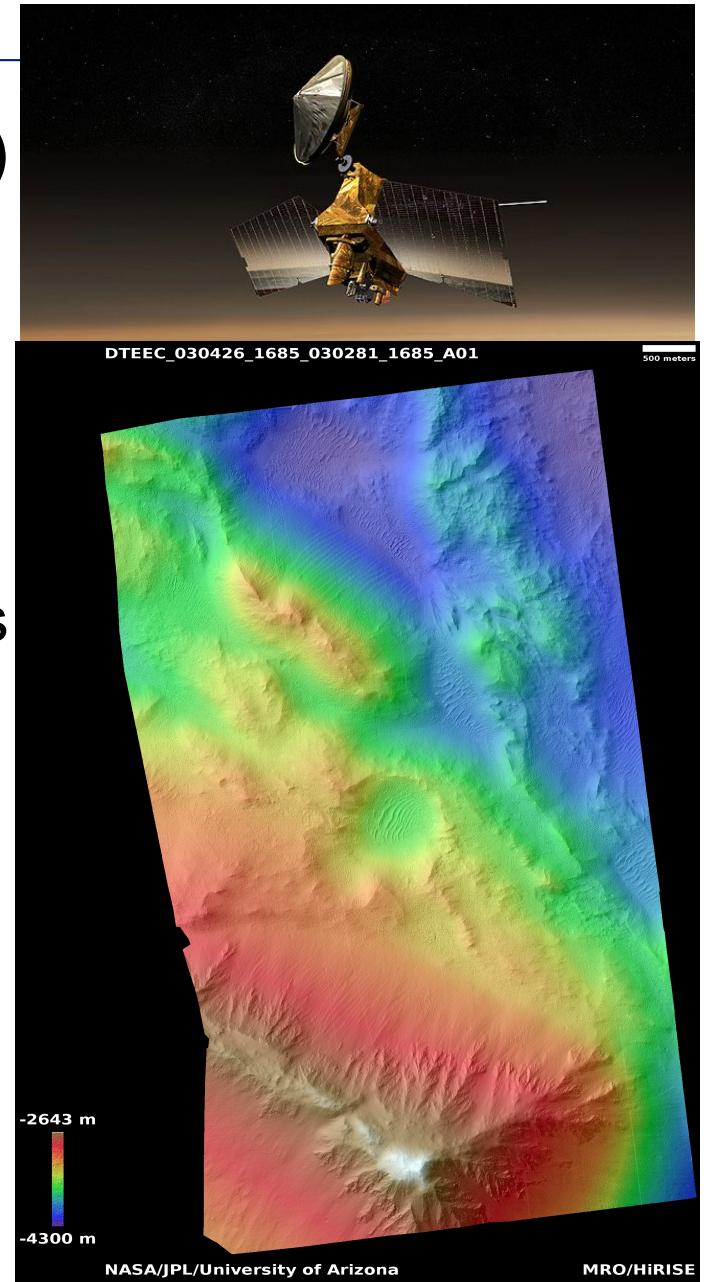
Introducción

- Mapas de costes
 - Extensión de mapas 2D
 - Añaden información
 - Típicamente combinación lineal de factores (rocas, pendiente, etc.)
- Objetivo: evitar zonas peligrosas



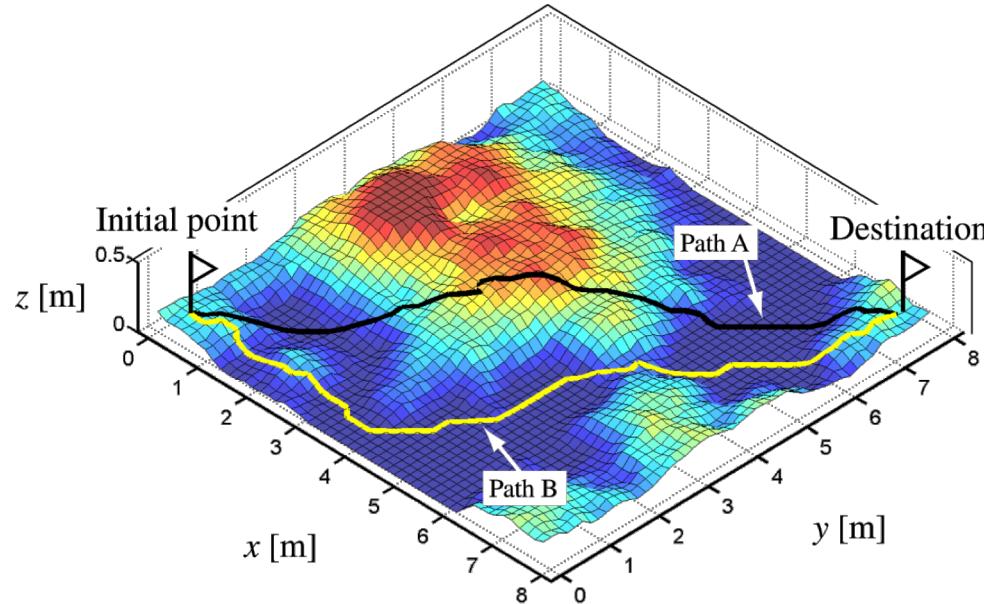
Introducción

- Mapas elevación digitales (DTM)
- Marte en alta resolución
 - Desde 2m de resolución horizontal
 - Hasta 25cm!
 - Resolución vertical de decímetros
- Usados para la planificación (MER/MSL)
- Libre descarga
www.uahirise.org/dtm



Introducción

- Combinación altitud + costes

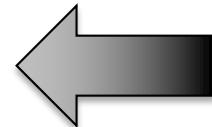


- ¿Qué hacer ante terrenos parcialmente conocidos?
 - Usualmente replanificar
 - Algoritmos específicos

Introducción

- Algoritmos deterministas

- Búsqueda no informada
- Búsqueda heurística



- Algoritmos estocásticos

- Búsqueda en árboles
- Algoritmos genéticos
- Colonia de hormigas

Resumen

- Introducción
- Dijkstra
- A*
- A*PS
- Theta*
- S-Theta*
- Conclusiones

Dijkstra

- Avanza al nodo no visitado más cercano al origen
- Calculando la distancia entre este y sus vecinos
- Si la distancia al vecino es menor, se actualiza
- Búsqueda no informada



Dijkstra

```
1  function Dijkstra(Graph, source):
2      dist[source] := 0                                // Initializations
3      for each vertex v in Graph:
4          if v ≠ source
5              dist[v] := infinity                     // Unknown distance from source to v
6              previous[v] := undefined                // Predecessor of v
7          end if
8          PQ.add_with_priority(v,dist[v])
9      end for
10
11
12     while PQ is not empty:                         // The main loop
13         u := PQ.extract_min()                      // Remove and return best vertex
14         for each neighbor v of u:                  // where v has not yet been removed from PQ.
15             alt = dist[u] + length(u, v)
16             if alt < dist[v]                      // Relax the edge (u,v)
17                 dist[v] := alt
18                 previous[v] := u
19                 PQ.decrease_priority(v,alt)
20             end if
21         end for
22     end while
23     return previous[]
```

Resumen

- Introducción
- Dijkstra
- A*
- A*PS
- Theta*
- S-Theta*
- Conclusiones

A*

- Búsqueda informada
 - Coste acumulado $G(t)$: coste para alcanzar t
 - Heurística $H(t)$: estimación del coste para alcanzar el objetivo desde t
- Heurísticas típicas
 - Distancia octil
 - Distancia euclídea
- Expande menos nodos que Dijkstra
- Sencillo, rápido y obtiene la ruta óptima en grafos de visibilidad
- Restringe los giros a 45°

A*



□ Posibles mejoras

- 16, 34, 64 vecinos
- Heurísticas
- Búsqueda bidireccional
- Multi-hilo

A*

Algorithm 1 A* search

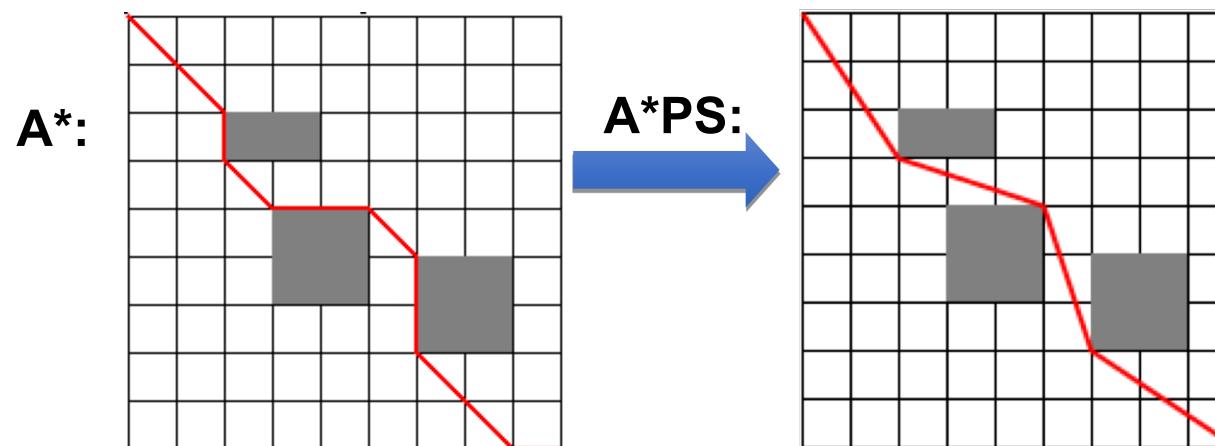
```
1  $G(s) \leftarrow 0$ 
2  $parent(s) \leftarrow s$ 
3  $open \leftarrow \emptyset$ 
4  $open.insert(s, G(s), H(s))$ 
5  $closed \leftarrow \emptyset$ 
6 while  $open \neq \emptyset$  do
7      $p \leftarrow open.pop()$ 
8     if  $p = g$  then
9         return  $path$ 
10    end if
11     $closed.insert(p)$ 
12    for  $t \in neighbours(p)$  do
13        if  $t \notin closed$  then
14            if  $t \notin open$  then
15                 $G(t) \leftarrow \infty$ 
16                 $parent(t) \leftarrow null$ 
17            end if
18             $UpdateVertex(p, t)$ 
19        end if
20    end for
21 end while
22 return  $fail$ 
```

Resumen

- Introducción
- Dijkstra
- A*
- A*PS
- Theta*
- S-Theta*
- Conclusiones

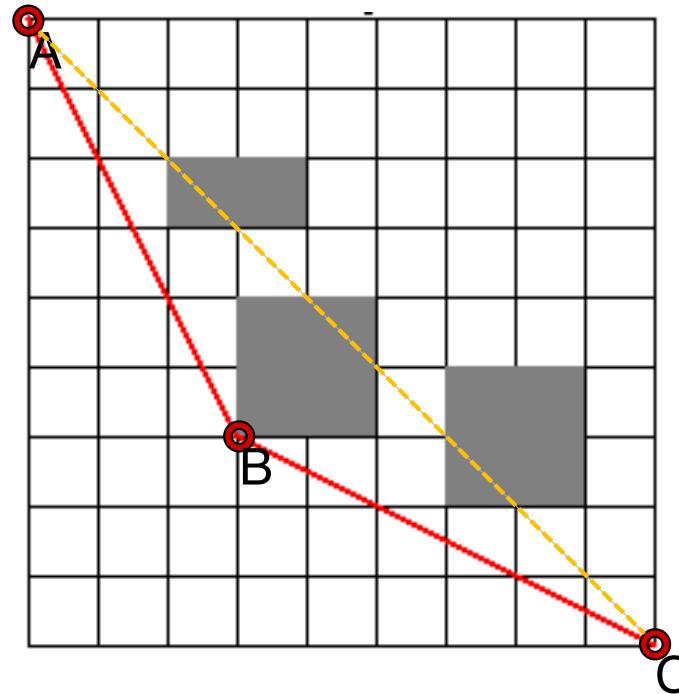
A*PS

- A*PS (A* Post Smoothed) reduce la ruta producida por A* eliminando nodos intermedios
- Introduce la línea de visibilidad
- Mejora la ruta → a costa de más cálculos



A*PS

- Línea de visibilidad
- El coste crece a medida que los nodos a comprobar se alejan entre sí

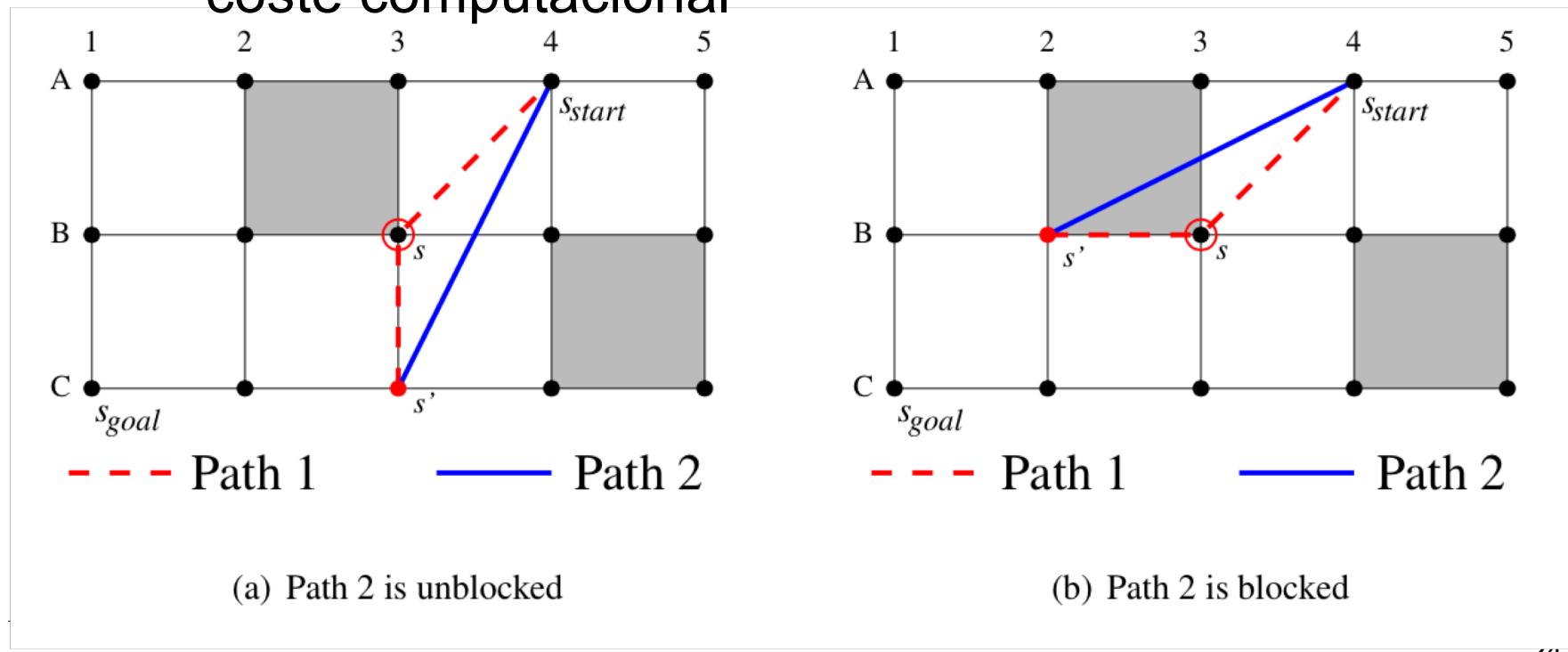


Resumen

- Introducción
- Dijkstra
- A*
- A*PS
- Theta*
- S-Theta*
- Conclusiones

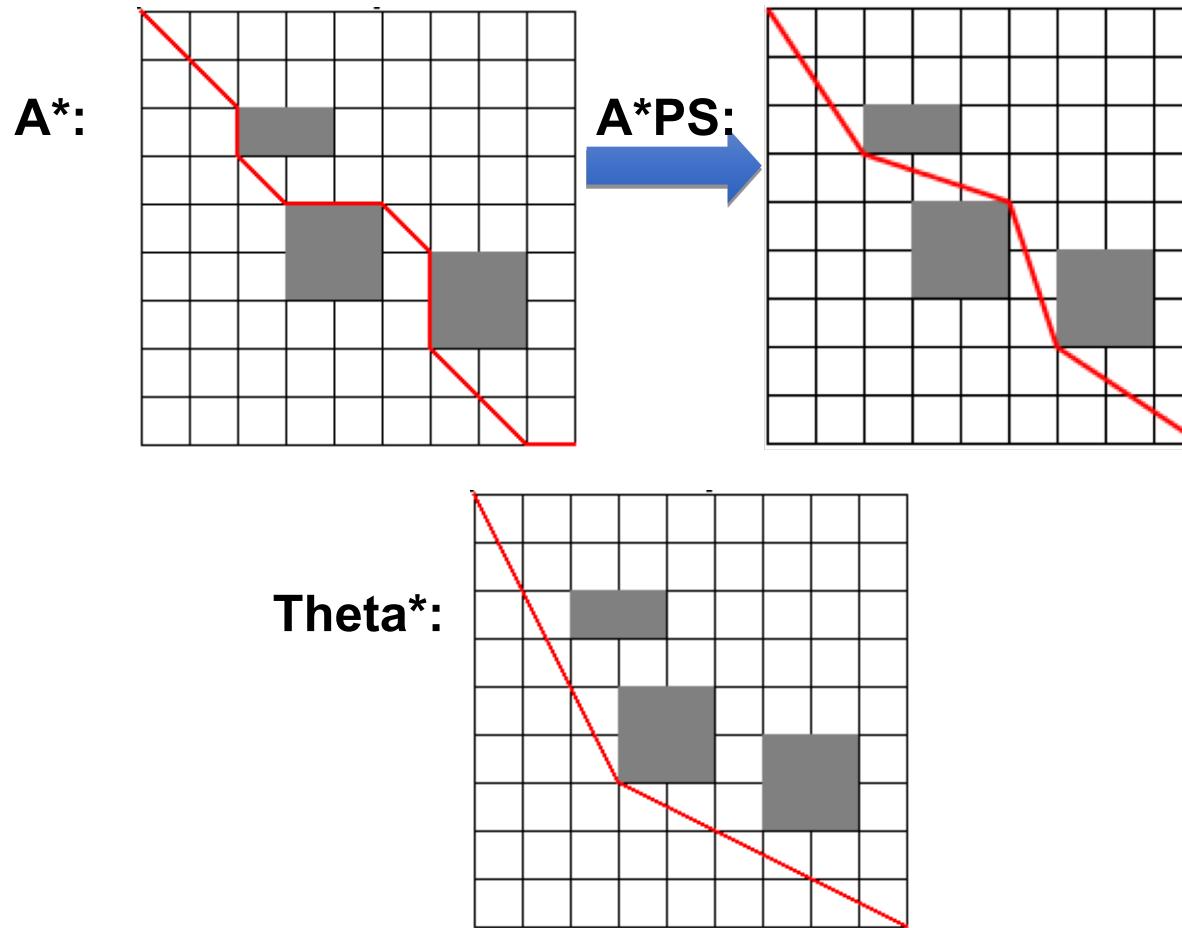
Theta*

- Variación de A* que integra la línea de visibilidad durante la búsqueda
 - Elimina la restricción de A*: los giros no están restringidos a $45^\circ \rightarrow$ rutas más naturales
 - Mejores rutas que A* (y que A*PS), pero con mayor coste computacional



Theta*

□ Comparativamente



Theta*

Algorithm 2 Update vertex function for Basic Theta*

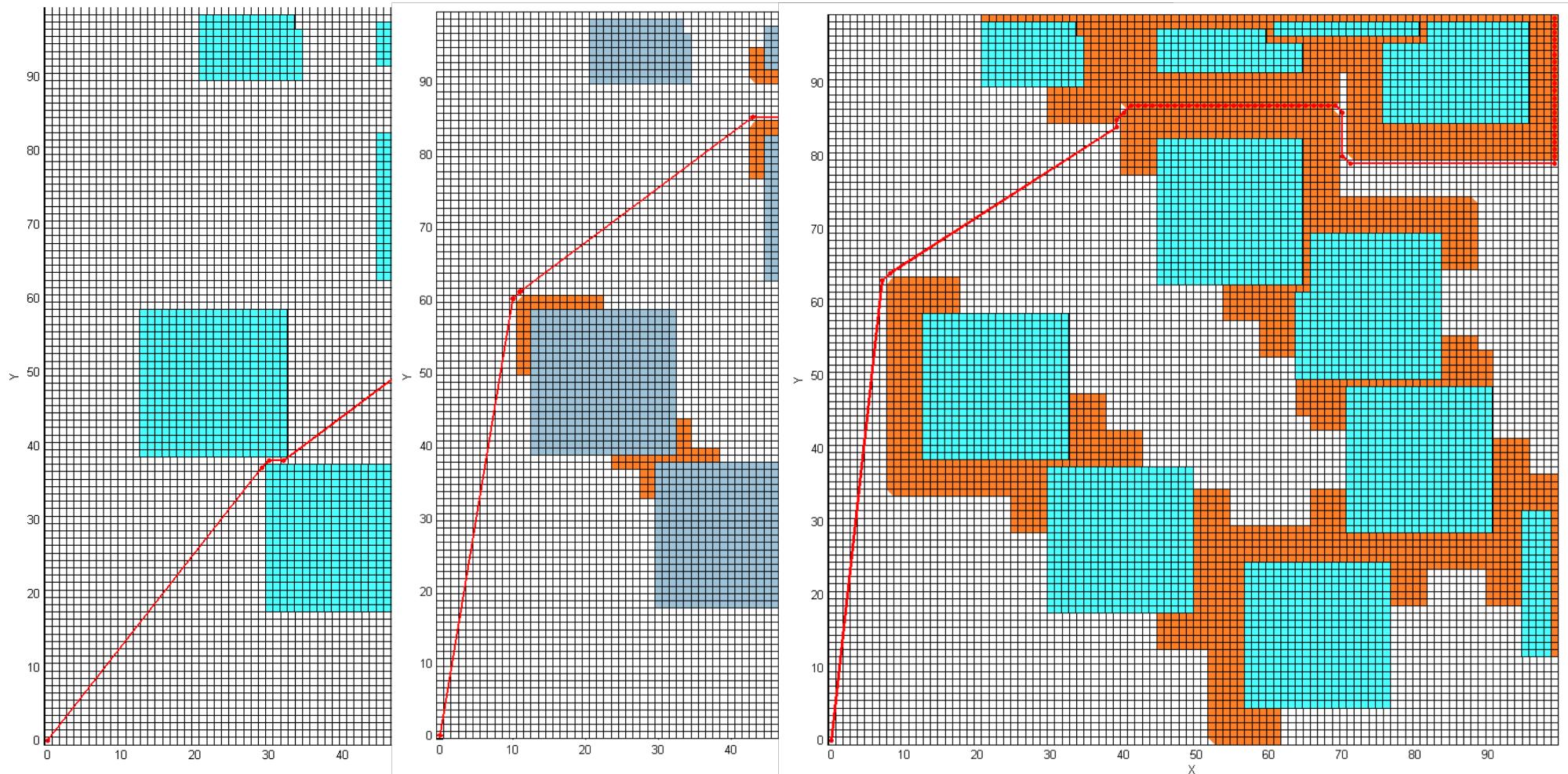
```
1 UpdateVertex(p, t)
2 if LineOfSight(parent(p), t) then
3     if G(parent(p)) + dist(parent(p), t) < G(t) then
4         G(t)  $\leftarrow$  G(parent(p)) + dist(parent(p), t)
5         parent(t)  $\leftarrow$  parent(p)
6         if t  $\in$  open then
7             open.remove(t)
8         end if
9         open.insert(t, G(t), H(t))
10    end if
11 else
12     if G(p) + dist(p, t) < G(t) then
13         G(t)  $\leftarrow$  G(p) + dist(p, t)
14         parent(t)  $\leftarrow$  p
15         if t  $\in$  open then
16             open.remove(t)
17         end if
18         open.insert(t, G(t), H(t))
19     end if
20 end if
```

Theta*

- Básicamente, Theta* es A*
- Cambia la función en la que se actualizan las conexiones entre los nodos
- Nodos no adyacentes pueden estar conectados
- Los cambios de dirección sólo se dan en los vértices de los obstáculos
 - ¿Puede ser un problema acercarse demasiado a un obstáculo?

Theta*

- Establecer márgenes de seguridad (aplicable a todos los algoritmos)



Resumen

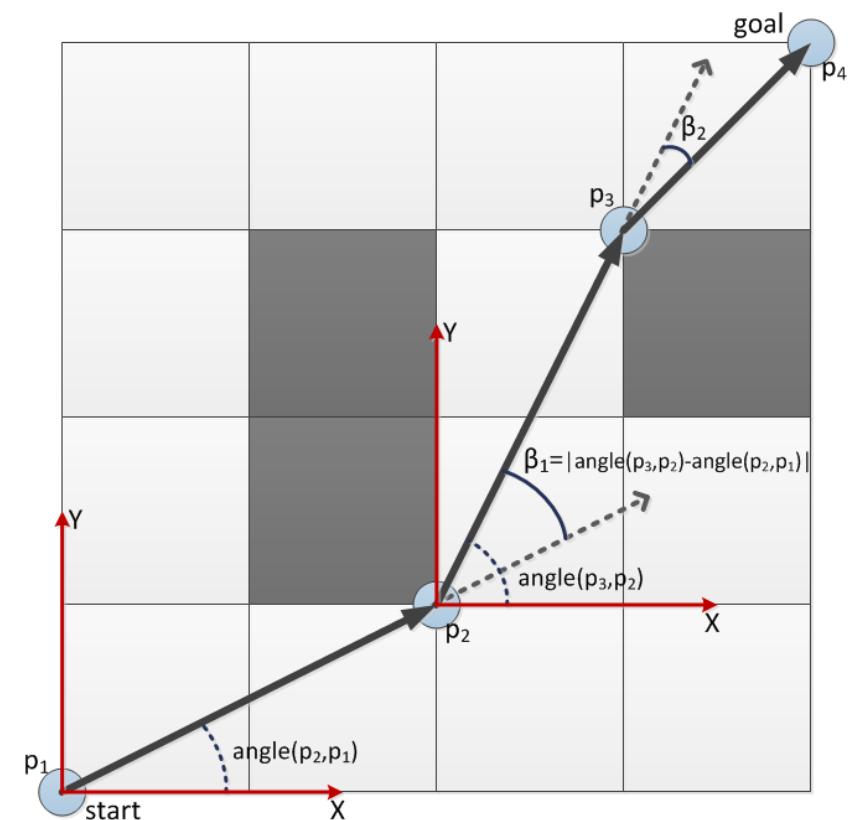
- Introducción
- Dijkstra
- A*
- A*PS
- Theta*
- S-Theta***
- Conclusiones

S-Theta*

- Theta* actualiza los nodos en función de la distancia en línea recta
- Clásicamente, la orientación no es tenida en cuenta
- S-Theta* es una modificación de Theta* que tiene en cuenta la orientación
 - Robots con limitaciones de giro
 - Rotar sobre el eje puede ser costoso
- El objetivo es reducir los giros, aunque implique recorrer una distancia algo mayor

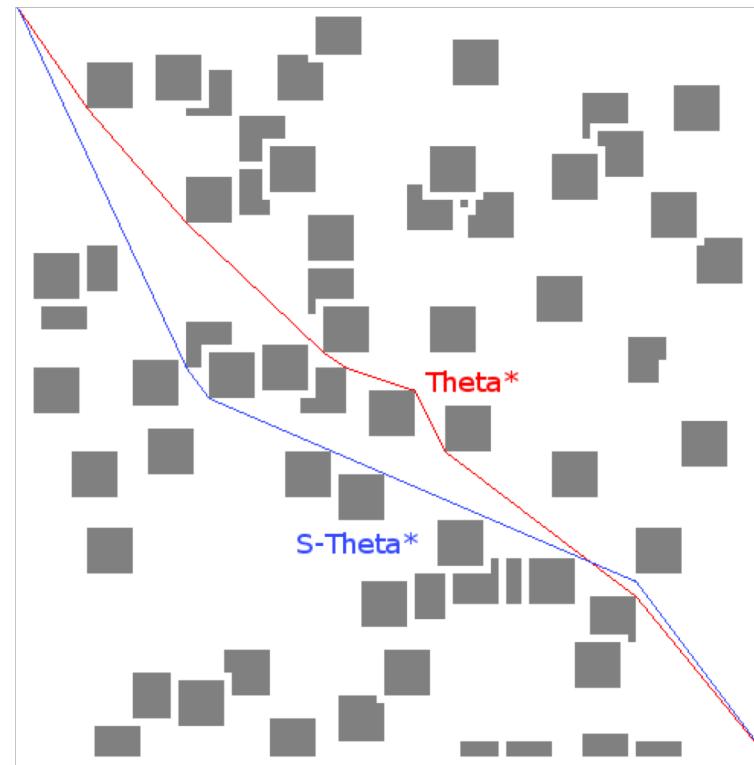
S-Theta*

- La modificación consiste en usar una función $\alpha(t)$ que representa el giro necesario para alcanzar t en relación al nodo padre y al objetivo
- $F(t) = G(t) + H(t) + \alpha(t)$
- Esto cambia la búsqueda
 - Reduce el número de giros
 - Así como su amplitud
- Rutas más suaves



S-Theta*

- $\alpha(t)$ “fuerza” a evitar los obstáculos sin acercarse tanto a ellos
- El margen de seguridad sigue siendo necesario
- Cambios de dirección en cualquier punto
- $\alpha(t) \in [0^\circ, 180^\circ]$



Resumen

- Introducción
- Dijkstra
- A*
- A*PS
- Theta*
- S-Theta*
- Conclusiones

Conclusiones

- El path planning clásico esta basado en búsqueda informada
- Fácil en entornos discretos y observables
- La heurística es muy importante
- Así como la representación del entorno
- Muchos trabajos publicados y margen de mejora
 - Field D* (JPL-NASA)
 - Lazy Theta* (¡útil para drones!)