

La planificación robótica

De la estrategia al movimiento.

Dr. Luis Felipe Marín Urías

Recordando paradigmas



Paradigma Reactivo



Paradigma Jerárquico

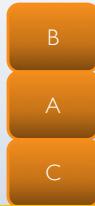
Tipos de planificación en robótica

- Planificación de acciones o tareas.
- Planificación de movimientos

Planificación de tareas o acciones

Razonamiento acerca de estados y acciones

Estado:



Sobre (B , A)
Sobre (A , C)
Sobre (C , Suelo)
Libre (B)
Libre (Suelo)

Razonamiento acerca de estados y acciones

Acción:

Mover (B , Suelo)



Estado:

Sobre (B , Suelo)
Sobre (A , C)
Sobre (C , Suelo)
Libre (B)
Libre (A)
Libre (Suelo)

Generación de un plan

S0



Sgoal



Plan de Acciones:

Estado S0:

Sobre (B , A)
Sobre (A , C)
Sobre (C , Suelo)
Libre (B)
Libre (Suelo)

Generación de un plan

S1



Sgoal



Plan de Acciones:

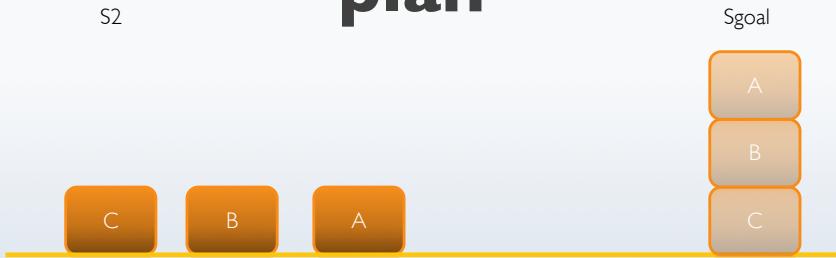
Mover (B , Suelo)

Estado S1:

Sobre (B , Suelo)
Sobre (A , C)
Sobre (C , Suelo)
Libre (B)
Libre (A)
Libre (Suelo)

Generación de un plan

S2



Sgoal



Plan de Acciones:

Mover (B, Suelo)
Mover (A, Suelo)

Estado S2:

Sobre (B , Suelo)
Sobre (A , Suelo)
Sobre (C , Suelo)
Libre (B)
Libre (A)
Libre (C)
Libre (Suelo)

S3



Sgoal



Plan de Acciones:

Mover (B, Suelo)
Mover (A, Suelo)
Mover (B, C)

Estado S3:

Sobre (B , C)
Sobre (A , Suelo)
Sobre (C , Suelo)
Libre (B)
Libre (A)
Libre (Suelo)

Generación de un plan

S4



Sgoal



Plan de Acciones:

Mover (B, Suelo)
Mover (A, Suelo)
Mover (B, C)
Mover (A, B)

Estado S4:

Sobre (B , C)
Sobre (A , B)
Sobre (C , Suelo)
Libre (A)
Libre (Suelo)

STRIPS

(Stanford Research Institute Problem Solver)

Es un generador de planes automatizado. El mismo nombre fue utilizado más tarde para referirse al lenguaje formal de las entradas de este generador de planes.

Una instancia de STRIPS se compone de:

- Un estado inicial.
- La especificación de estados de meta.
- Un conjunto de acciones, para cada una de las cuales se incluyen:
 - Precondiciones.
 - Postcondiciones.

STRIPS

Matemáticamente, una **instancia** de STRIPS es una tupla $\{P, O, I, G\}$, donde:

- **P** es un conjunto de condiciones (es decir, variables proposicionales).
- **O** es un conjunto de operadores (es decir, acciones); cada operador es también una tupla $\{\alpha, \beta, \gamma, \delta\}$, donde cada elemento es un conjunto de condiciones:
 - α representa a las condiciones que deben ser verdaderas para que la acción pueda ejecutarse.
 - β representa a las condiciones que deben ser falsas para que la acción pueda ejecutarse.
 - γ representa a las condiciones se hacen verdaderas si se ejecuta la acción.
 - δ representa a las condiciones se hacen falsas si se ejecuta la acción.
- **I** es el estado inicial, dado por un conjunto de condiciones que son inicialmente verdaderas (todas las demás se asumen falsas).
- **G** es la especificación del estado de meta, que está dada por una dupla, que especifica qué condiciones deben ser verdaderas y cuales falsas para considerar a un estado como meta.

Un **plan** para una instancia es una secuencia de operadores que puede ser ejecutada desde el estado inicial, y que lleva hasta un estado meta.

Un ejemplo simple

Un robot requiere ir de Xalapa a Tijuana:

Estado Meta: En Tijuana, B.C. (1000, 2828).
Estado inicial: En Xalapa, Ver. (0,0).

Diferencia: 1000

Diferencia	Operador
$d \geq 200$	Volar
$100 < d < 200$	Ir_en_tren
$d \leq 100$	Conducir
$d < 1$	Caminar

Precondiciones

Diferencia	Operador	Precondiciones
$d \geq 200$	Volar	
$100 < d < 200$	Ir_en_tren	
$d \leq 100$	Conducir_rentado	En aeropuerto
	Conducir_personal	En casa
$d < 1$	Caminar	

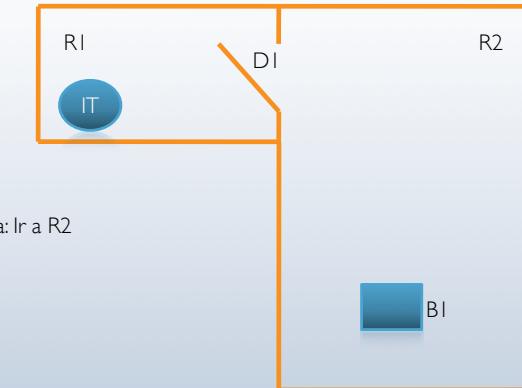
Listas

Diferencia	Operador	Precondicione	Lista a	Lista a borrar
$d \geq 200$	Volar		En Y En Aeropuerto	En X
$100 < d < 200$	Ir_en_tren		En Y En la estación	En X
$d \leq 100$	Conducir_rentado	En aeropuerto		
	Conducir_personal	En casa		
$d < 1$	Caminar			

Creando un ejemplo más realista

- Crear el modelo del mundo.
- El mundo debe estar representado por hechos o axiomas, en lógica de predicados.
- Los predicados son funciones que evalúan en Verdadero o Falso.
- Los predicados están en mayúsculas por convención de la programación en IA.
- Las variables están en minúsculas.
- Constantes también van en mayúsculas.

Modelo del mundo



Predicados

El robot puede:

- Saber si un objeto móvil está en un cuarto, cerca de una puerta o de otro objeto móvil
- Saber si una puerta está abierta o cerrada.
- Saber que cuartos están conectados.

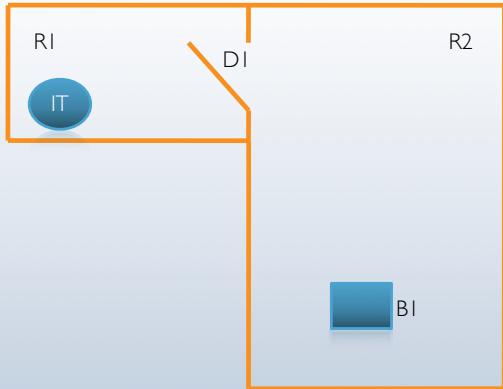
Tipos de cosas en el mundo:

- objeto_móvil: robot, cajas para transportar
- Cuartos (Rx)
- Puertas (Dx)

Conocimiento del robot

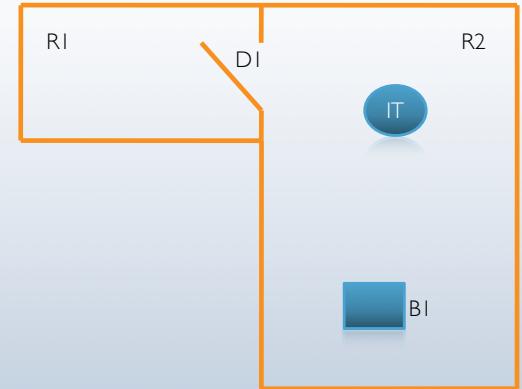
- INROOM (x, r)
x = objeto_móvil
r = cuarto
- NEXTO (x, t)
t = {objeto_móvil, puerta}
- STATUS (d, s)
d = puerta
s = estado = {OPEN, CLOSE}
rx, ry = cuarto
- CONNECTS (d, rx, ry)

Estados



Estado Inicial:
INROOM (IT, R1)
INROOM (BI, R2)
CONNECTS (DI, R1, R2)
CONNECTS (DI, R2, R1)
STATUS (DI, OPEN)

Estados



Estado Meta:
INROOM (IT, R2)
INROOM (BI, R2)
CONNECTS (DI, R1, R2)
CONNECTS (DI, R2, R1)
STATUS (DI, OPEN)

Tabla de diferencias

Operador	Precondiciones	Add-list	Delete-list
OPI: GOTODOOR(IT,dx)	INROOM(IT,rk) CONNECT(dx,rk,rm)	NEXTTO(IT,dm)	
OP2: GOTHUDOOR(IT,dx)	CONNECT(dx,rk,rm) NEXTTO(IT,dm) STATUS(dx,OPEN) INROOM(IT,rk)	INROOM(IT,rm)	INROOM(IT,rk)

El mundo cerrado y el problema del marco

Mundo cerrado:

El modelo funciona solo para un problema en particular y nada más.

Quién es marco?

The frame problem

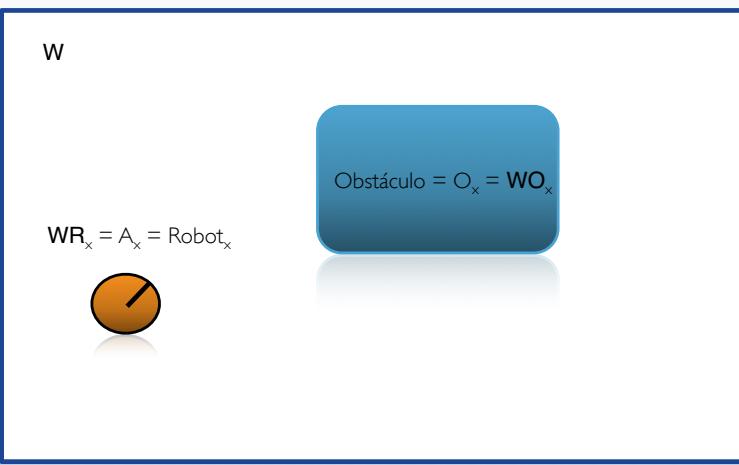
El tamaño de la descripción del mundo

Planificación de movimientos

Por qué hacer la planificación de movimientos?

- Ayudar al robot a desplazarse en un ambiente conocido o semi-conocido.
- Encontrar una forma para que un objeto vaya de un lugar a otro lugar (Piano Mover's Problem).
- Encontrar una serie de configuraciones con la que un robot puede ensamblar un auto.
- Aplicaciones: Animación, Video Juegos, Asistencia Automotriz, Navegación robótica, Manufactura, etc.

Representando el mundo



Configuración de un robot o cuerpo móvil

Robot Móvil:



Una configuración es una especificación de todos los puntos de un robot relativos a un sistema de coordenadas

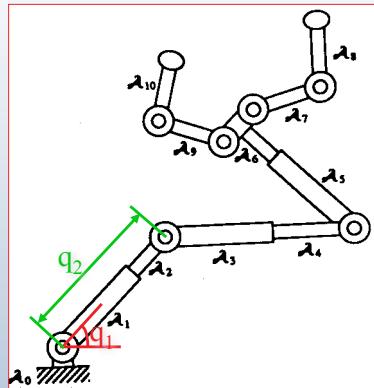


Generalmente se representa en forma de vector

$$q = \{ q_1, q_2, q_3 \}$$

Configuración de un Robot

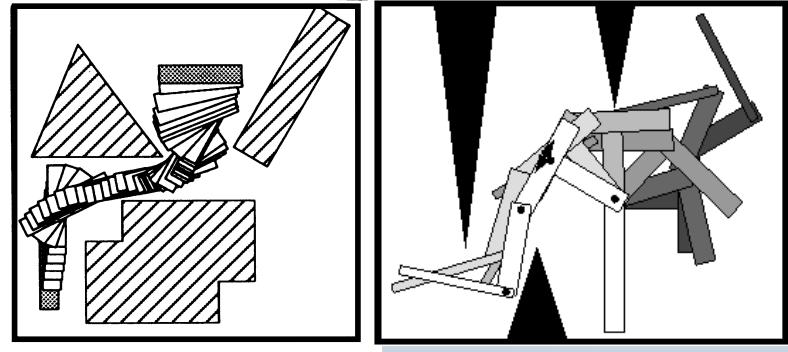
Robot Articulado:



$$\mathbf{q} = (q_1, q_2, \dots, q_{10})$$

Que es una trayectoria?

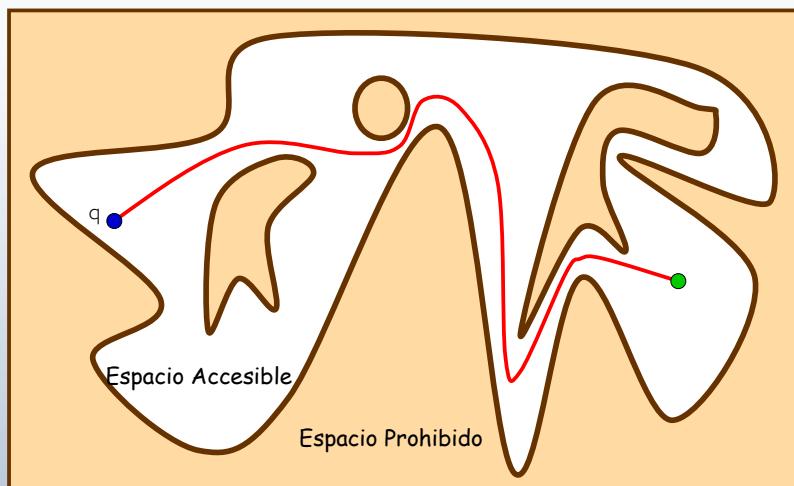
Una secuencia de configuraciones para llegar de un punto a un otro



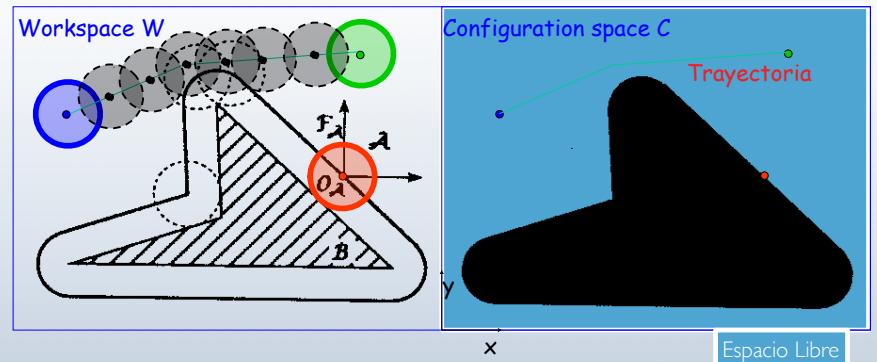
Depende de la morfología así como de su modelo cinemático

Encontrar una trayectoria

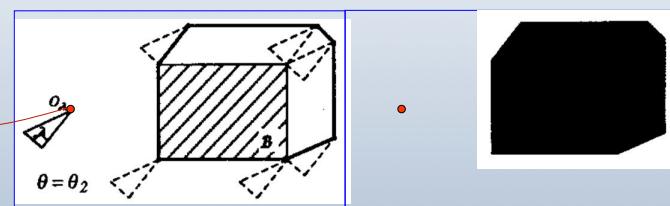
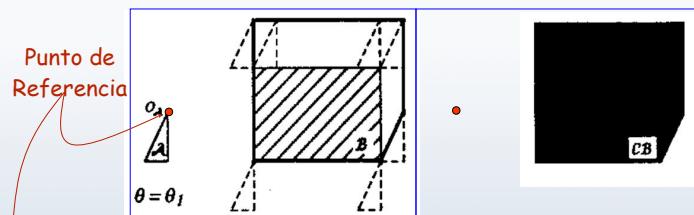
Se reduce la representación del robot a un punto:



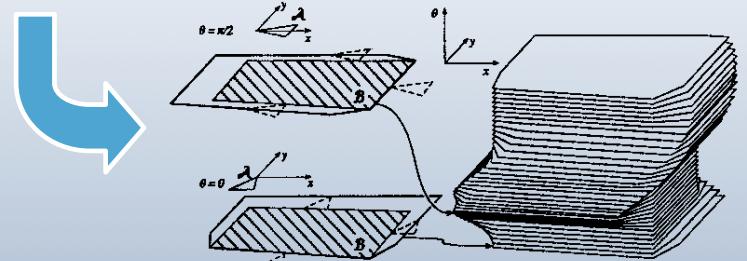
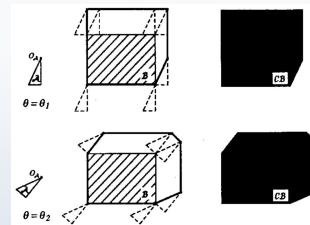
Espacio de configuración 2D



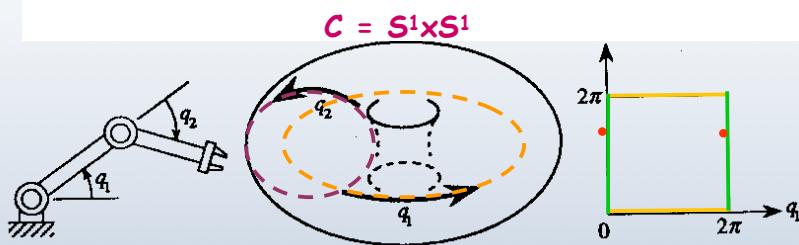
Diferentes morfologías



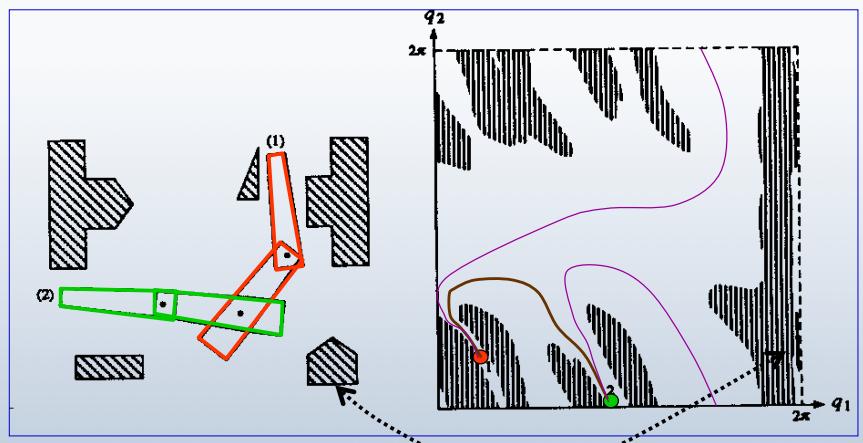
Trasladando y rotando un polígono en 2D



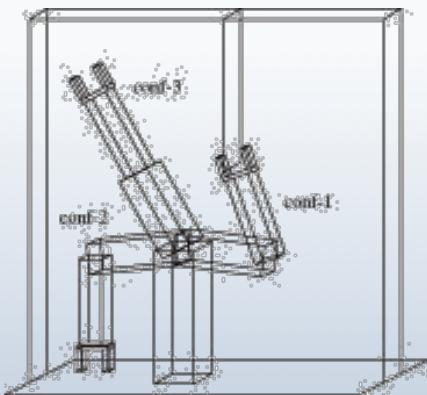
Espacio de configuración mas elaborado



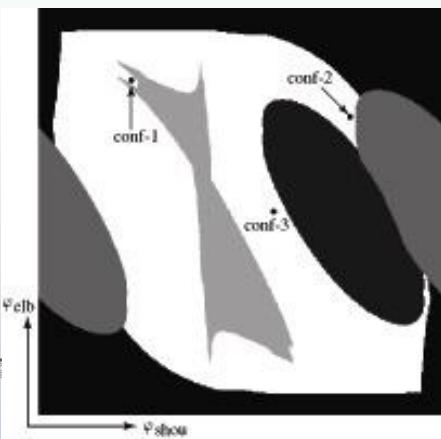
Robot articulado



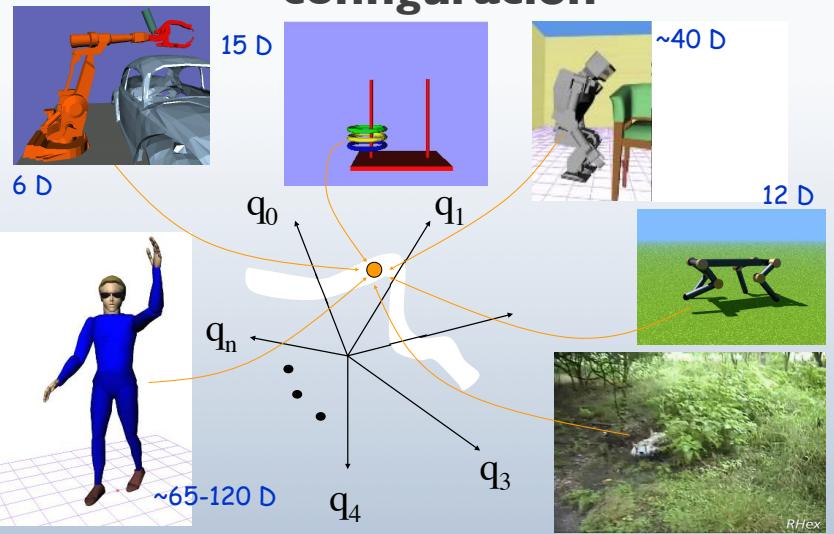
Robot articulado



Workspace



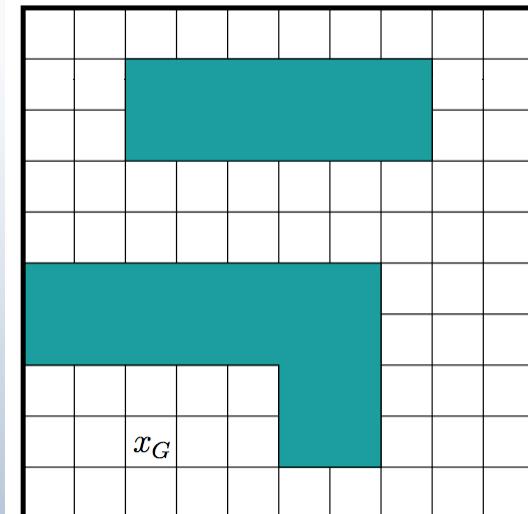
Todos tienen su espacio de configuración



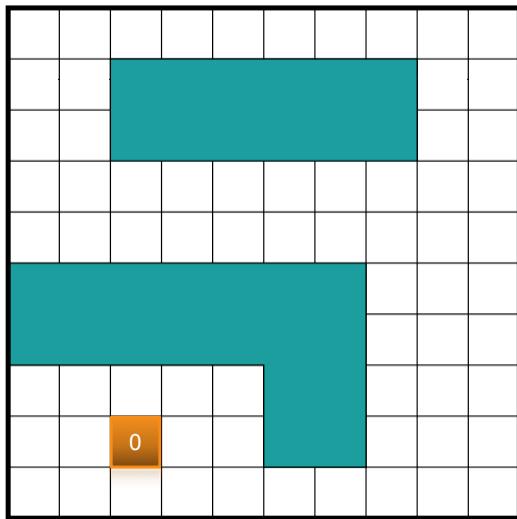
Objetivo: Llegar de un punto a un otro

Metodología: Grid

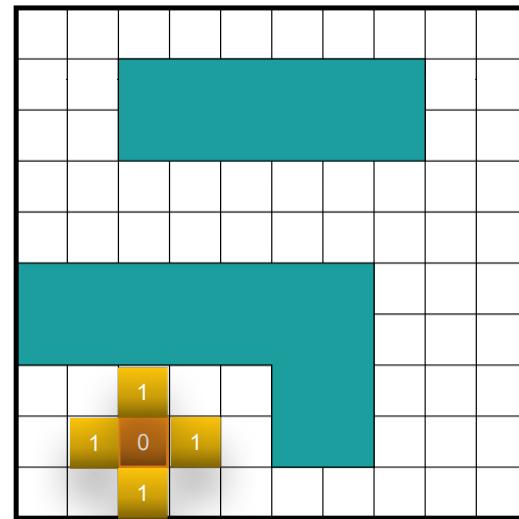
Grid de ocupación



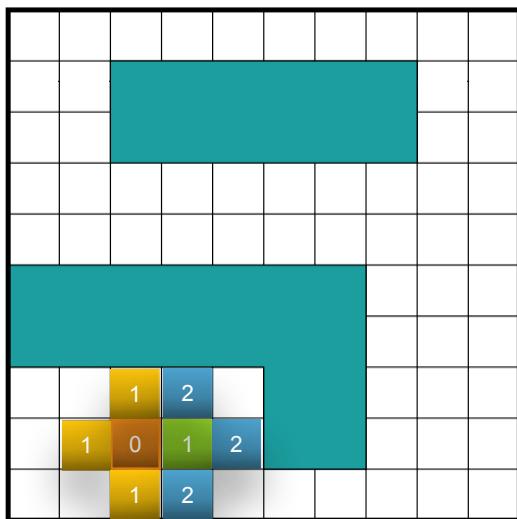
Wavefront Expansion Planning



Wavefront Expansion Planning



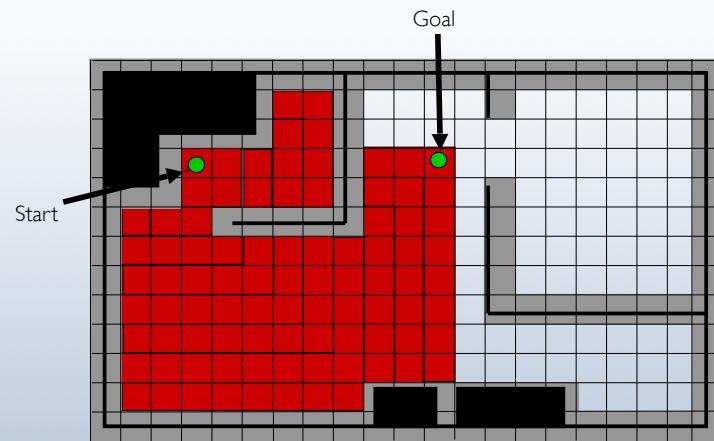
Wavefront Expansion Planning



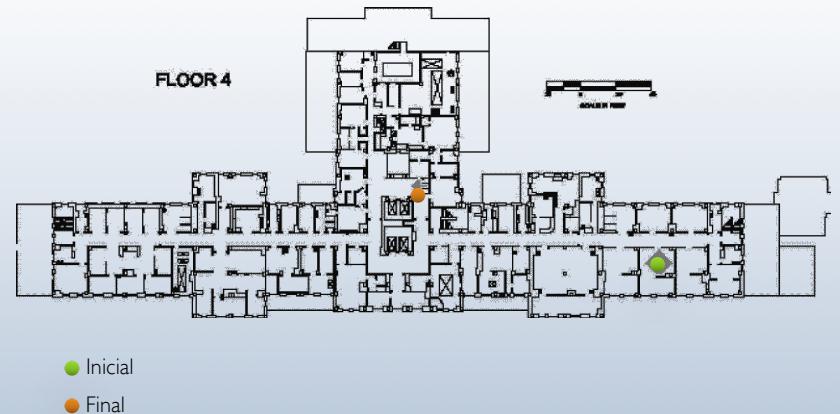
Wavefront Expansion Planning

22	21	22	21	20	19	18	17	16	17	
21	20							15	16	
20	19							14	15	
19	18	17	16	15	14	13	12	13	14	
18	17	16	15	14	13	12	11	12	13	
								10	11	12
								9	10	11
3	2	1	2	3				8	9	10
2	1	0	1	2				7	8	9
3	2	1	2	3	4	5	6	7	8	

Wavefront Expansion Planning



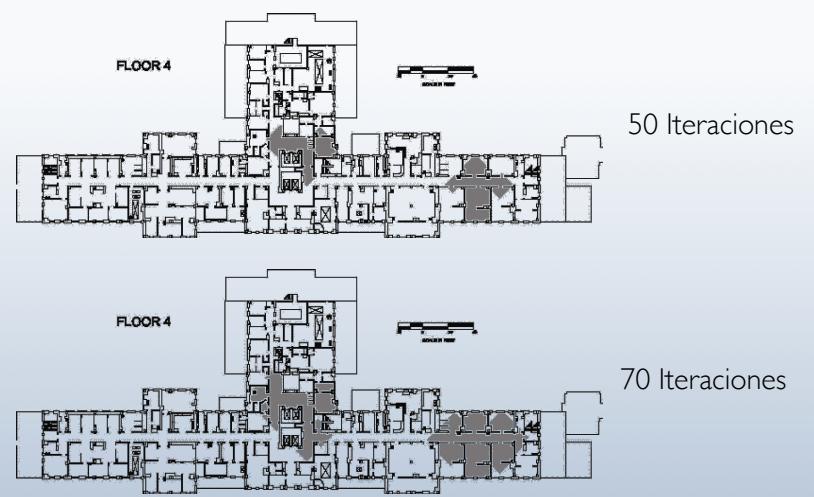
Wavefront Dual



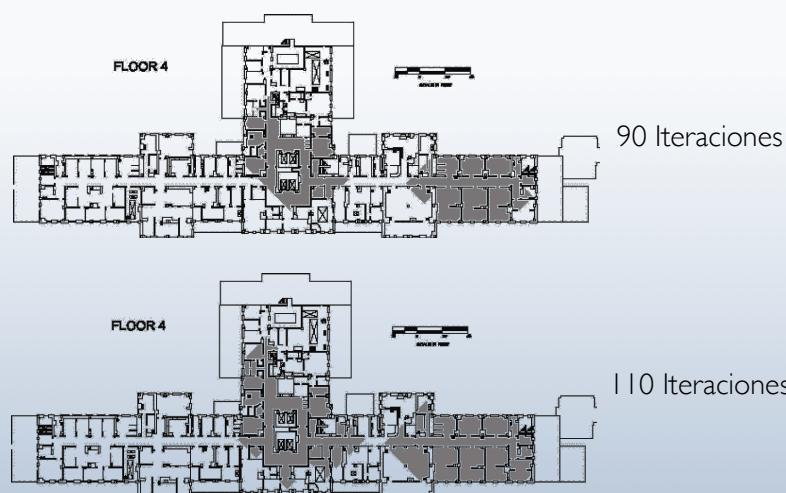
Wavefront Dual



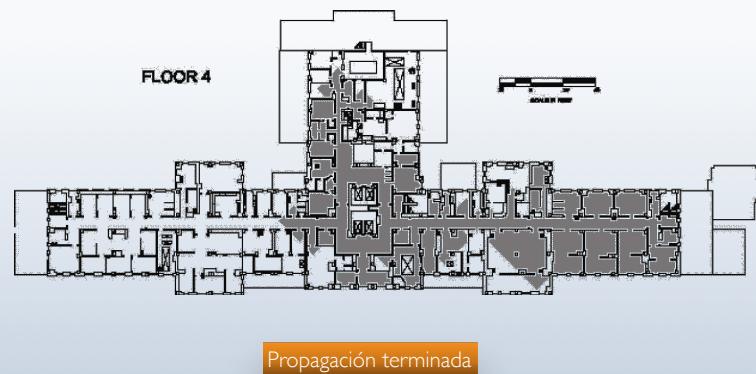
Wavefront Dual



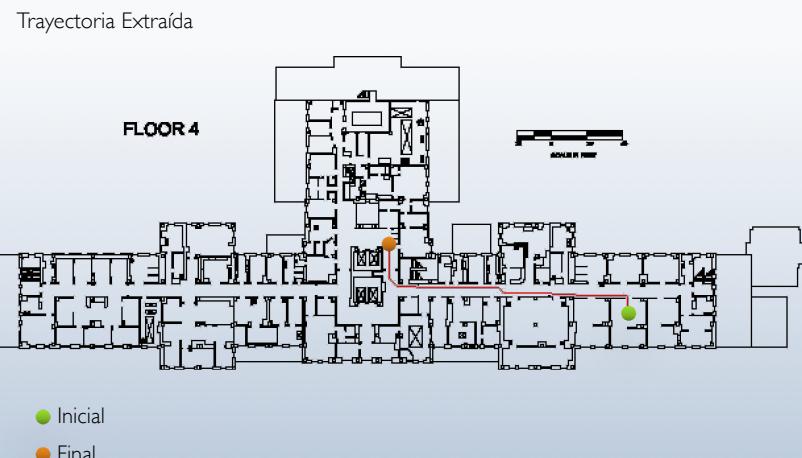
Wavefront Dual



Wavefront Dual



Wavefront Dual



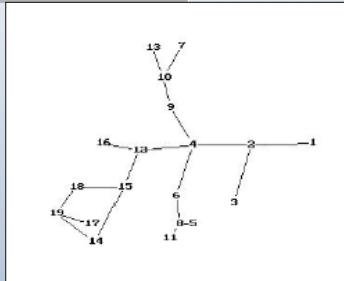
Objetivo: Llegar de un punto a un otro

Metodología: Mapas Topológicos

Mapas topológicos



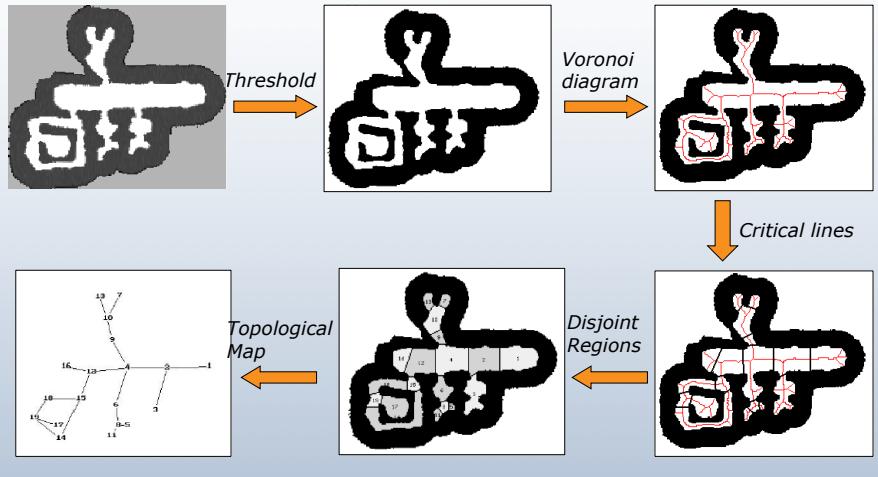
Mapa de grid presenta una precisión alta
Pero son complejos de calcular



Mapas topológicos facilitan la planificación de trayectorias

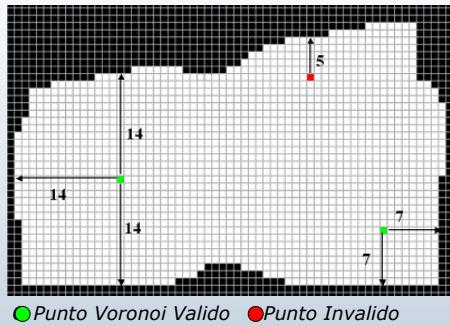
Representados por grafos

Pasando de grid a topológico

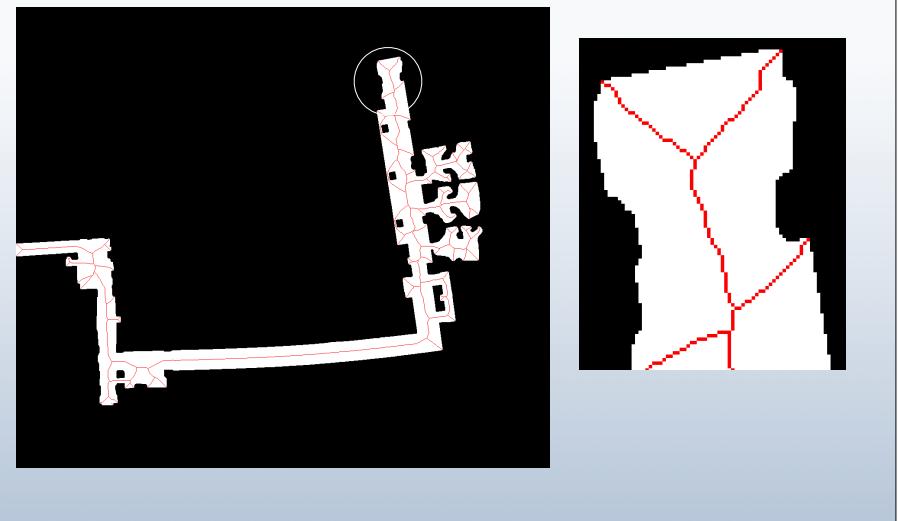


Diagramas de Voronoi

Conjunto de puntos que cumplen con la propiedad de que contienen dos puntos más cercanos a los obstáculos y que son equidistantes.



Ejemplo



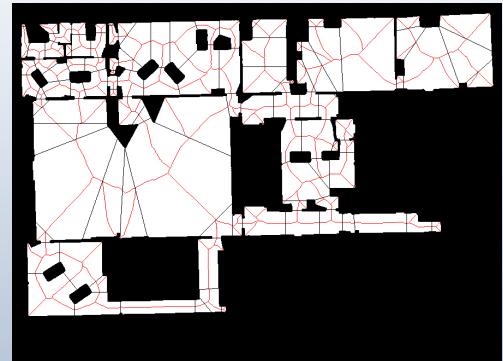
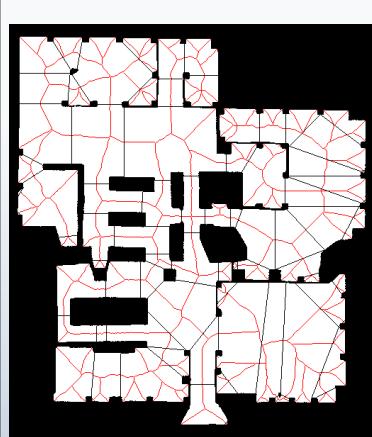
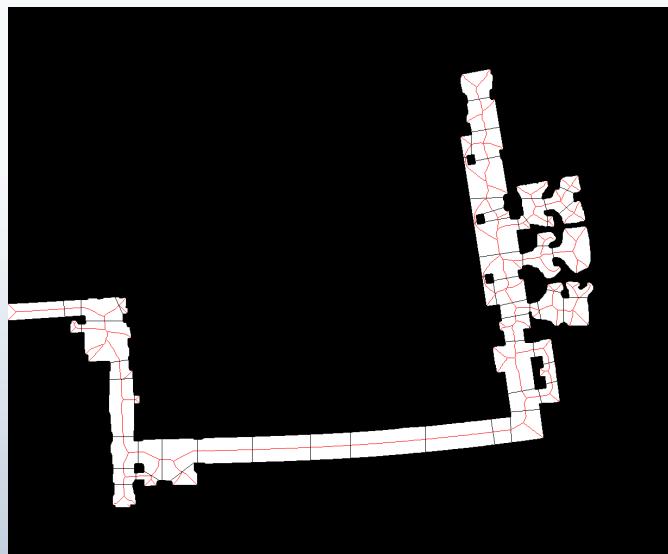
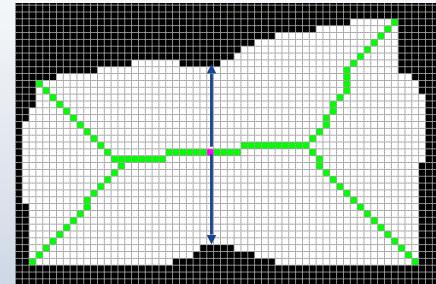
Más ejemplos



Puntos críticos

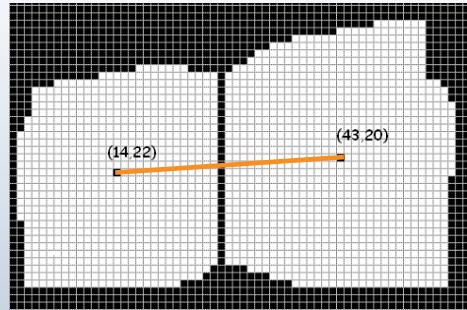
Un punto crítico es un punto dentro del diagrama de Voronoi que minimiza la distancia entre el espacio ocupado, de manera local

Se dibujan las líneas que lo atraviesan y se obtienen las líneas críticas



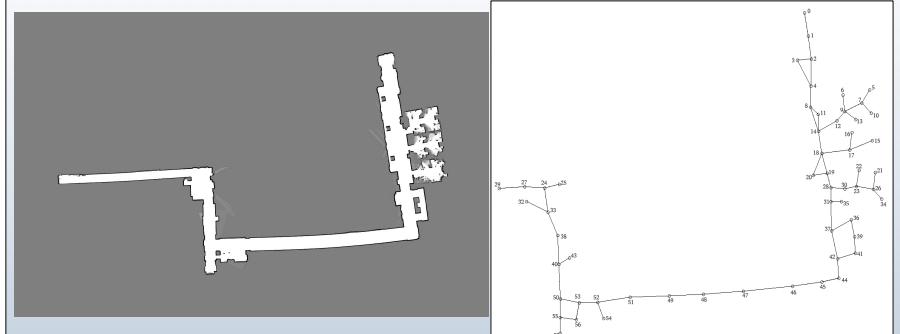
Extracción del Mapa topológico

Se obtiene el centro de masa de las regiones:

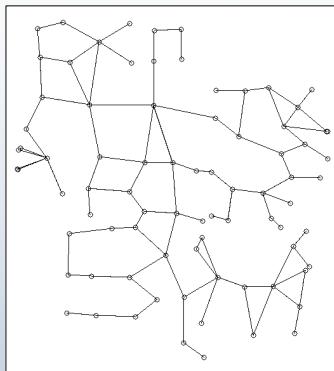
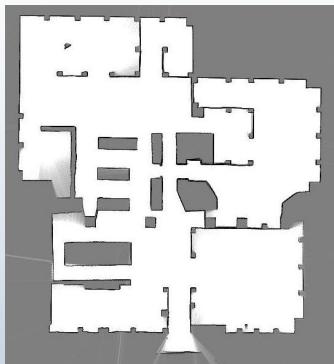


Se unen por una línea y se les proporcionan identificadores: ID 0 a ID 1

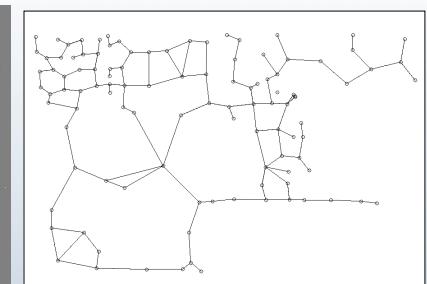
Mapa 1



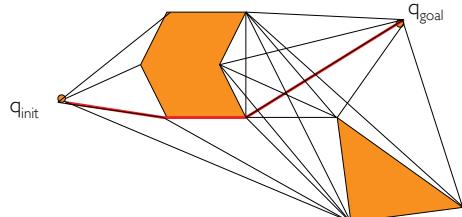
Mapa 2



Mapa 3



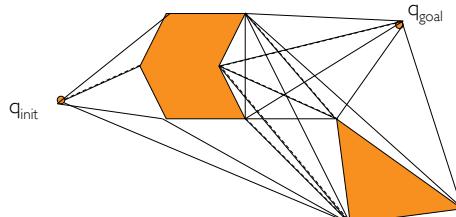
Visibility Graph



Algoritmo:

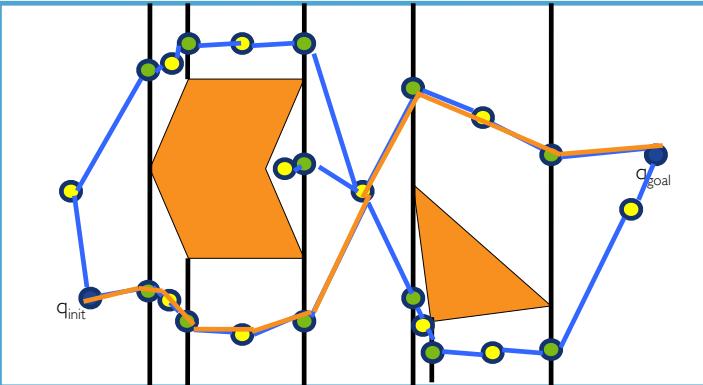
- Generar el grafo de visibilidad G
- Buscar en G una trayectoria de q_{init} a q_{goal}
- Si se encuentra: Regresar la ruta.
- Sino: Falla!!!

Visibility Graph Reducido



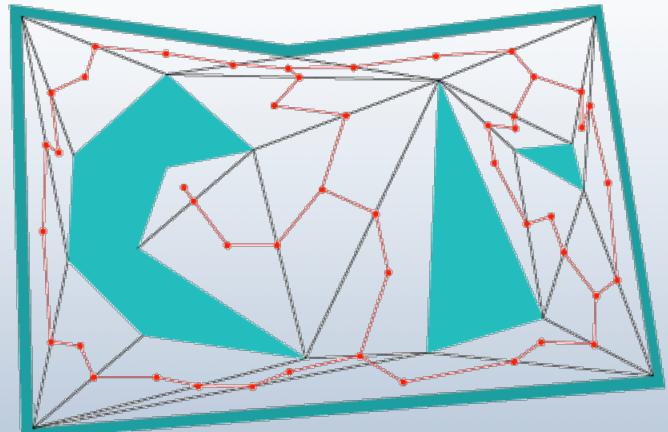
- Se eliminan las aristas o segmentos que pertenezcan a una región cóncava
- Se eliminan las aristas o segmentos que no sean tangenciales:
Ósea aquellos que en la ecuación de la recta toquen al obstáculo en 2 puntos o mas, exceptuando a aquellos segmentos que que se encuentren totalmente dentro de dicho obstáculo

Descomposición de celdas



- Líneas tangenciales y verticales del perímetro los vértices
- Puntos en medio de las celdas y en medio de los puntos
- Unir Puntos
- Encontrar camino

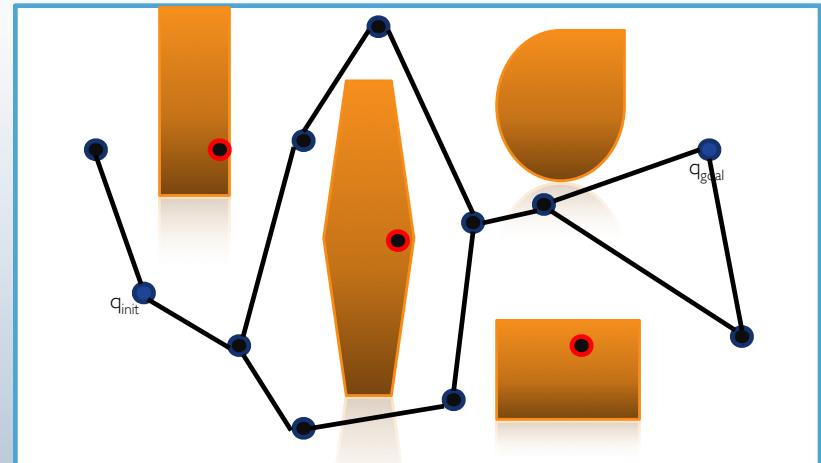
Descomposición en triangulos



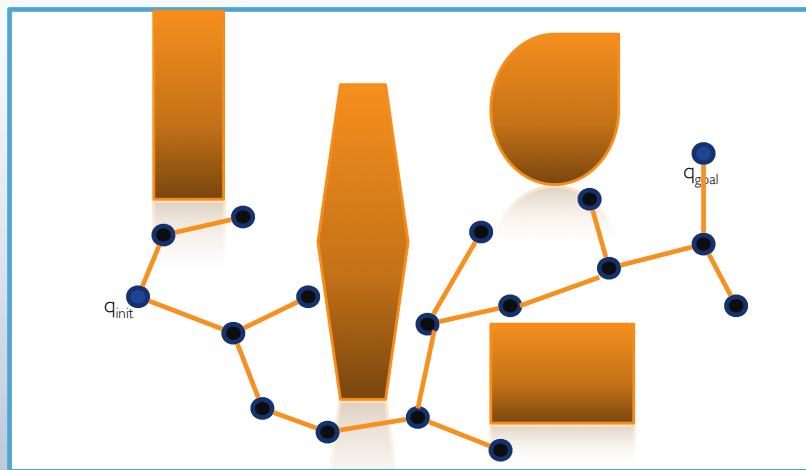
Objetivo: Llegar de un punto a un otro

Metodología: Mapas probabilísticos

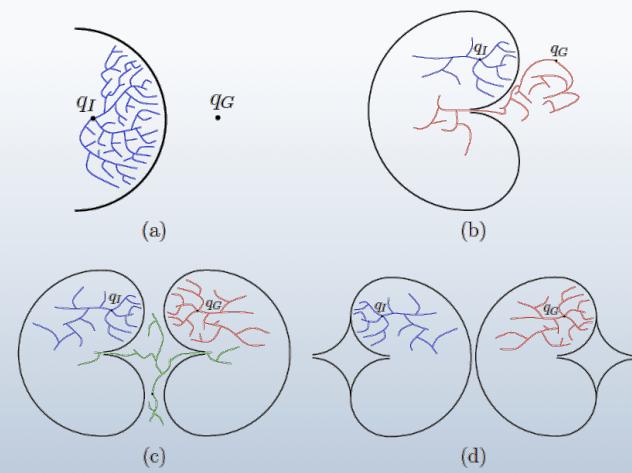
PRM – Probabilistic Road Maps



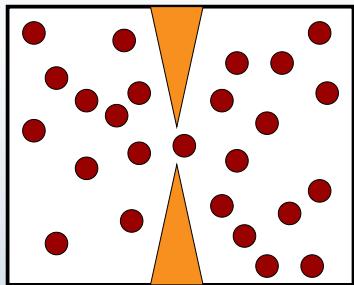
RRT's – Rapidly-Exploring Random Trees



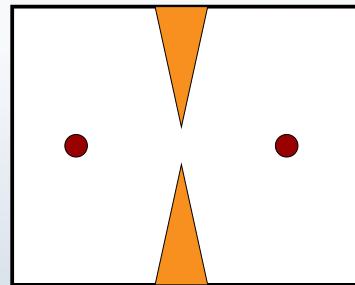
RRT's – Rapidly-Exploring Random Trees



Visibility - PRM



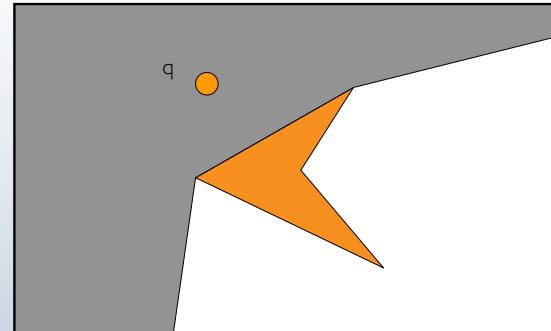
PRM Clásico



PRM con visibilidad

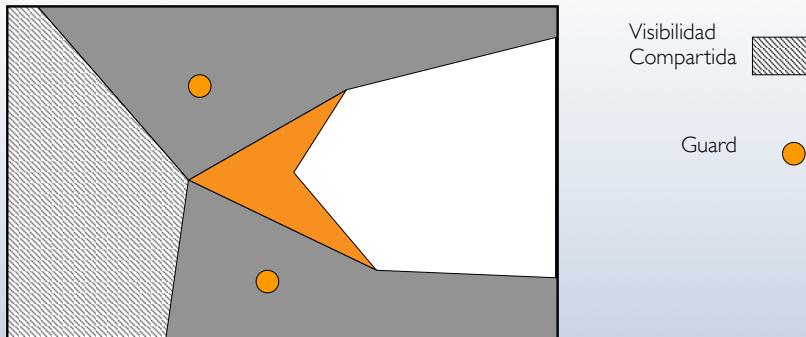
Asegurando cobertura

- Dominio de visibilidad de la configuración q :



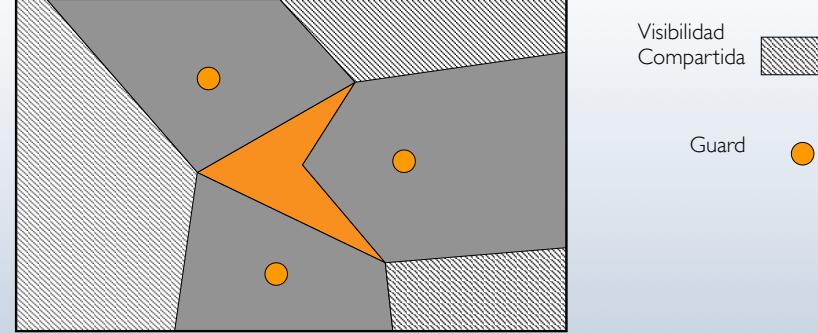
Ensuring Coverage

- Cobertura del espacio libre con nodos **Guard** (Guardias)



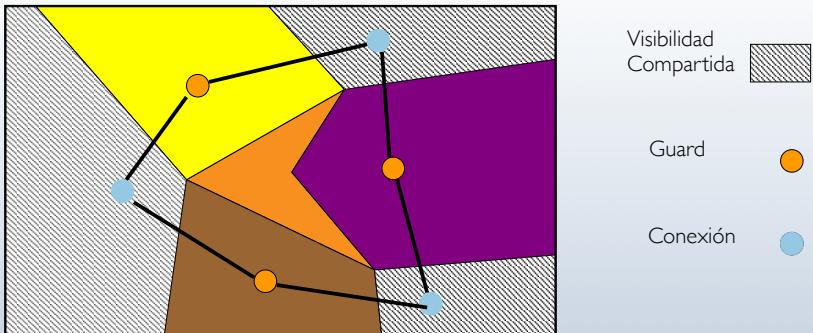
Ensuring Coverage

- Cobertura del espacio libre con nodos **Guard** (Guardias)



Creando conexiones

- Completando el roadmap con nodos de **Conexión**

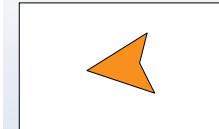


Algoritmo

```

Algorithm Visib-PRM
    Guard  $\leftarrow \emptyset$ ; Connection  $\leftarrow \emptyset$ ; ntry  $\leftarrow 0$ 
    While (ntry < M)
        Select a random free configuration q
         $g_{vis} \leftarrow \emptyset$ ;  $G_{vis} \leftarrow \emptyset$ 
        For all components  $G_i$  of Guard do
            found  $\leftarrow \text{FALSE}$ 
            For all nodes g of  $G_i$  do
                If(q belongs to  $Vis(g)$ ) then
                    found  $\leftarrow \text{TRUE}$ 
                    If ( $g_{vis} = \emptyset$ ) then  $g_{vis} \leftarrow g$ ;  $G_{vis} \leftarrow G_i$ 
                    Else /* q is a connection node */
                        Add q to Connection
                        Create edges (q, g) and (q,  $g_{vis}$ )
                        Merge components  $G_{vis}$  and  $G_i$ ;
            End
            Until found = TRUE
            If ( $g_{vis} = \emptyset$ ) then /* q is a guard node */
                Add {q} to Guard; ntry  $\leftarrow 0$ 
            Else ntry  $\leftarrow ntry + 1$ 
        End
    End

```

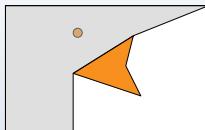


Algoritmo

```

Algorithm Visib-PRM
    Guard  $\leftarrow \emptyset$ ; Connection  $\leftarrow \emptyset$ ; ntry  $\leftarrow 0$ 
    While (ntry < M)
        Select a random free configuration q
         $g_{vis} \leftarrow \emptyset$ ;  $G_{vis} \leftarrow \emptyset$ 
        For all components  $G_i$  of Guard do
            found  $\leftarrow \text{FALSE}$ 
            For all nodes g of  $G_i$  do
                If(q belongs to  $Vis(g)$ ) then
                    found  $\leftarrow \text{TRUE}$ 
                    If ( $g_{vis} = \emptyset$ ) then  $g_{vis} \leftarrow g$ ;  $G_{vis} \leftarrow G_i$ 
                    Else /* q is a connection node */
                        Add q to Connection
                        Create edges (q, g) and (q,  $g_{vis}$ )
                        Merge components  $G_{vis}$  and  $G_i$ ;
            End
            Until found = TRUE
            If ( $g_{vis} = \emptyset$ ) then /* q is a guard node */
                Add {q} to Guard; ntry  $\leftarrow 0$ 
            Else ntry  $\leftarrow ntry + 1$ 
        End
    End

```

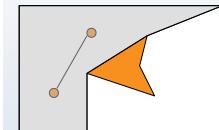


Algoritmo

```

Algorithm Visib-PRM
    Guard  $\leftarrow \emptyset$ ; Connection  $\leftarrow \emptyset$ ; ntry  $\leftarrow 0$ 
    While (ntry < M)
        Select a random free configuration q
         $g_{vis} \leftarrow \emptyset$ ;  $G_{vis} \leftarrow \emptyset$ 
        For all components  $G_i$  of Guard do
            found  $\leftarrow \text{FALSE}$ 
            For all nodes g of  $G_i$  do
                If(q belongs to  $Vis(g)$ ) then
                    found  $\leftarrow \text{TRUE}$ 
                    If ( $g_{vis} = \emptyset$ ) then  $g_{vis} \leftarrow g$ ;  $G_{vis} \leftarrow G_i$ 
                    Else /* q is a connection node */
                        Add q to Connection
                        Create edges (q, g) and (q,  $g_{vis}$ )
                        Merge components  $G_{vis}$  and  $G_i$ ;
            End
            Until found = TRUE
            If ( $g_{vis} = \emptyset$ ) then /* q is a guard node */
                Add {q} to Guard; ntry  $\leftarrow 0$ 
            Else ntry  $\leftarrow ntry + 1$ 
        End
    End

```



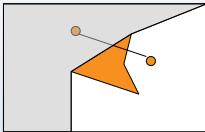
Algoritmo

Algorithm Visib-PRM

```

Guard ← ∅; Connection ← ∅; ntry ← 0
While (ntry < M)
    Select a random free configuration q
    gvis ← ∅; Gvis ← ∅
    For all components Gi of Guard do
        found ← FALSE
        For all nodes g of Gi do
            If(q belongs to Vis(g)) then
                found ← TRUE
                If (gvis = ∅) then gvis ← g; Gvis ← Gi
                Else /* q is a connection node */
                    Add q to Connection
                    Create edges (q, g) and (q, gvis)
                    Merge components Gvis and Gi;
            until found = TRUE
            If (gvis = ∅) then/* q is a guard node */
                Add {q} to Guard; ntry ← 0
            Else ntry ← ntry + 1
    End

```



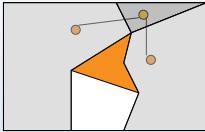
Algoritmo

Algorithm Visib-PRM

```

Guard ← ∅; Connection ← ∅; ntry ← 0
While (ntry < M)
    Select a random free configuration q
    gvis ← ∅; Gvis ← ∅
    For all components Gi of Guard do
        found ← FALSE
        For all nodes g of Gi do
            If(q belongs to Vis(g)) then
                found ← TRUE
                If (gvis = ∅) then gvis ← g; Gvis ← Gi
            Else /* q is a connection node */
                Add q to Connection
                Create edges (q, g) and (q, gvis)
                Merge components Gvis and Gi;
    until found = TRUE
    If (gvis = ∅) then/* q is a guard node */
        Add {q} to Guard; ntry ← 0
    Else ntry ← ntry + 1
End

```



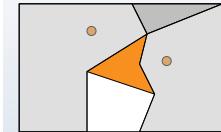
Algoritmo

Algorithm Visib-PRM

```

Guard ← ∅; Connection ← ∅; ntry ← 0
While (ntry < M)
    Select a random free configuration q
    gvis ← ∅; Gvis ← ∅
    For all components Gi of Guard do
        found ← FALSE
        For all nodes g of Gi do
            If(q belongs to Vis(g)) then
                found ← TRUE
                If (gvis = ∅) then gvis ← g; Gvis ← Gi
                Else /* q is a connection node */
                    Add q to Connection
                    Create edges (q, g) and (q, gvis)
                    Merge components Gvis and Gi;
    until found = TRUE
    If (gvis = ∅) then/* q is a guard node */
        Add {q} to Guard; ntry ← 0
    Else ntry ← ntry + 1
End

```



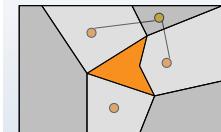
Algoritmo

Algorithm Visib-PRM

```

Guard ← ∅; Connection ← ∅; ntry ← 0
While (ntry < M)
    Select a random free configuration q
    gvis ← ∅; Gvis ← ∅
    For all components Gi of Guard do
        found ← FALSE
        For all nodes g of Gi do
            If(q belongs to Vis(g)) then
                found ← TRUE
                If (gvis = ∅) then gvis ← g; Gvis ← Gi
                Else /* q is a connection node */
                    Add q to Connection
                    Create edges (q, g) and (q, gvis)
                    Merge components Gvis and Gi;
    until found = TRUE
    If (gvis = ∅) then/* q is a guard node */
        Add {q} to Guard; ntry ← 0
    Else ntry ← ntry + 1
End

```



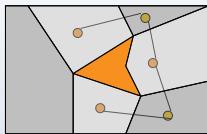
Algoritmo

Algorithm Visib-PRM

```

Guard ← ∅; Connection ← ∅; ntry ← 0
While (ntry < M)
    Select a random free configuration q
    gvis ← ∅; Gvis ← ∅
    For all components Gi of Guard do
        found ← FALSE
        For all nodes g of Gi do
            If(q belongs to Vis(g)) then
                found ← TRUE
                If (gvis = ∅) then gvis ← g; Gvis ← Gi
                Else /* q is a connection node */
                    Add q to Connection
                    Create edges (q, g) and (q, gvis)
                    Merge components Gvis and Gi;
            until found = TRUE
            If (gvis = ∅) then/* q is a guard node */
                Add {q} to Guard; ntry ← 0
            Else ntry ← ntry + 1
    End

```



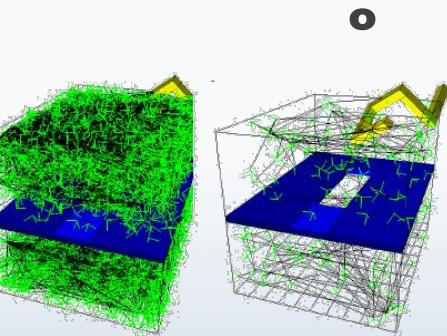
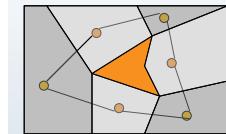
Algoritmo

Algorithm Visib-PRM

```

Guard ← ∅; Connection ← ∅; ntry ← 0
While (ntry < M)
    Select a random free configuration q
    gvis ← ∅; Gvis ← ∅
    For all components Gi of Guard do
        found ← FALSE
        For all nodes g of Gi do
            If(q belongs to Vis(g)) then
                found ← TRUE
                If (gvis = ∅) then gvis ← g; Gvis ← Gi
                Else /* q is a connection node */
                    Add q to Connection
                    Create edges (q, g) and (q, gvis)
                    Merge components Gvis and Gi;
            until found = TRUE
            If (gvis = ∅) then/* q is a guard node */
                Add {q} to Guard; ntry ← 0
            Else ntry ← ntry + 1
    End

```



PRM	Basic	Visibility
Roadmap size	4723	103
CS_{free} coverage	99.9%	99.7%
Random confs	14169	14369
Free confs	4723	4753
Local method #calls	700610	57622
Col. checker #calls	8.725985	1.121790
CPU time	3367 sec	281 sec