



Sistemas de Control para Robots

Tema 2. Planificación local y global

Manuel Ocaña Miguel
Elena López Guillén
Rafael Barea Navarro

1. Introducción a los procesos cognitivos

2. Planificación local:

- Evitación de obstáculos
- Comportamientos reactivos

3. Planificación global:

- Búsqueda en grafos
- Grafos de visibilidad
- Campos de potencial
- Teoría de decisiones : Procesos de Decisión de Markov (MDP y POMDP)

Laboratorio (Grupo reducido):

- Práctica 2. Planificación local y global



1. Introducción a los procesos cognitivos (planificación)

□ **Proceso Cognitivo / Razonamiento :**

- es la habilidad para decidir **que acciones son necesarias** realizar para conseguir un **determinado objetivo** en una **situación dada**
- decisiones varían desde **qué camino tomar** hasta **qué información del entorno hay que utilizar.**

□ Hoy en día, los **robots industriales** pueden trabajar incluso **sin ningún proceso** cognitivo ya que el entorno en el que trabajan puede ser perfectamente estructurado y estático.

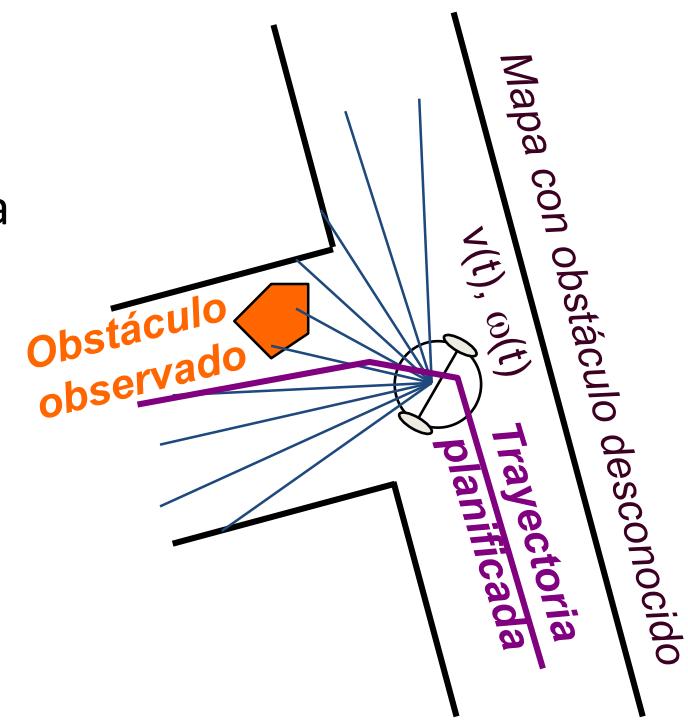
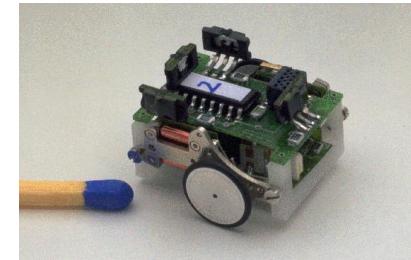
□ En la **robótica móvil**, los procesos cognitivos y el razonamiento son **procesos básicos de su naturaleza**, tales como determinar el camino más seguro o determinar donde ir después de realizar una acción.

- En robótica móvil, el conocimiento sobre el entorno y la situación es normalmente un **conocimiento parcial** y con **incertidumbre**:
 - Esto hace que las tareas sean mucho más difíciles
 - Se requieren múltiples tareas en paralelo, algunas de evitación de obstáculos para garantizar la “supervivencia del robot”.
- El control del robot normalmente se suele descomponer en varias funciones o comportamientos:
 - p.e. seguir paredes, localización, generación del camino o evitación de obstáculos.
- Entre los principales procesos cognitivos se puede distinguir la planificación (**planificación global**) y la evitación de obstáculos (**planificación local**)



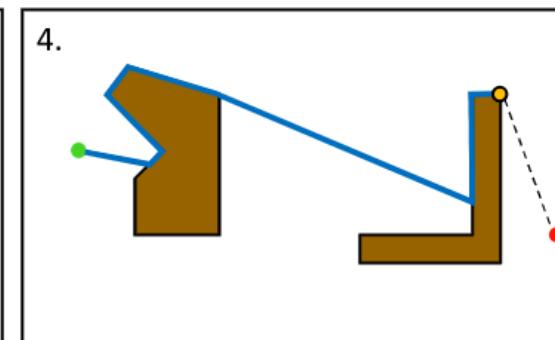
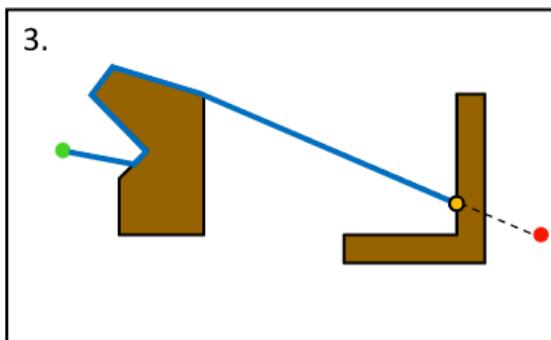
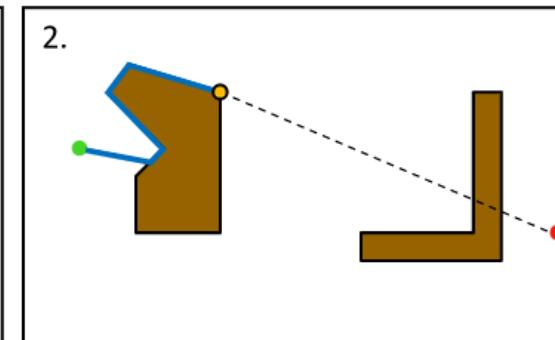
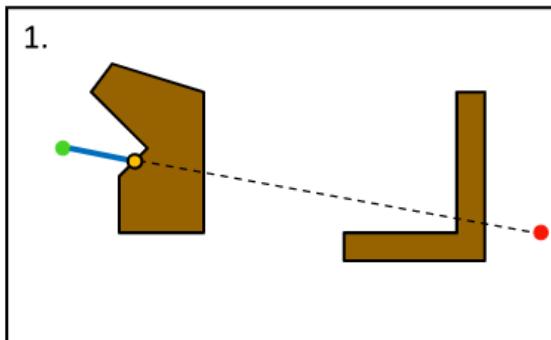
2. Planificación local

- El objetivo de los **algoritmos** de evitación de obstáculos es la de **evitar colisiones** entre el robot y los objetos del entorno
- La evitación de obstáculos se considera una **planificación local**.
- Normalmente se basan en mapas locales
- Esta tarea, **imprescindible** en la robótica móvil, a menudo se implementa como una tarea más o menos **independiente**.
- Sin embargo, la evitación de obstáculos debería ser eficiente con respecto a los siguiente criterios:
 1. El **objetivo** global
 2. La velocidad actual y la **cinemática** del robot
 3. Los **sensores** de a bordo
 4. El riesgo actual y futuro de colisión



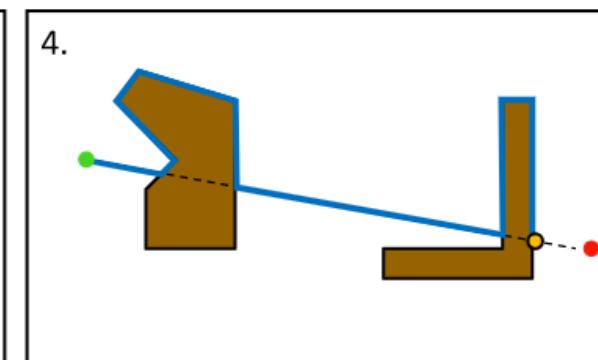
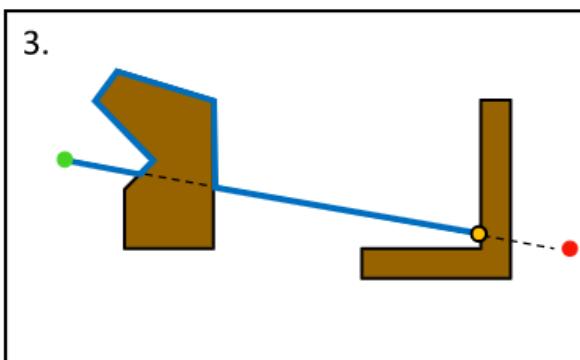
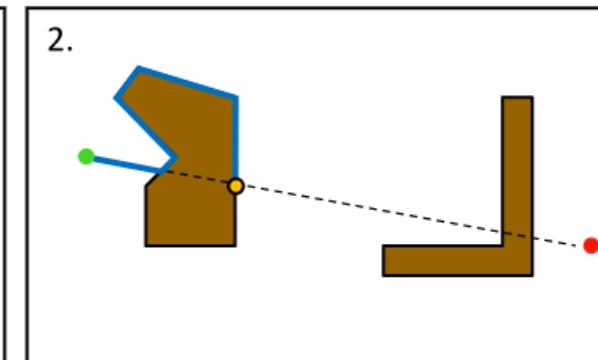
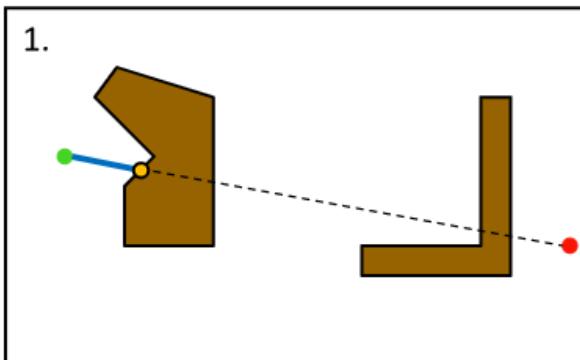
□ ALGORITMO DE EVITACIÓN BUG1

- Moverse en línea recta hacia el destino
- Si se alcanza el destino, finaliza
- Si se alcanza un obstáculo, rodearlo por la izquierda hasta que la línea que une el robot con el destino esté libre.



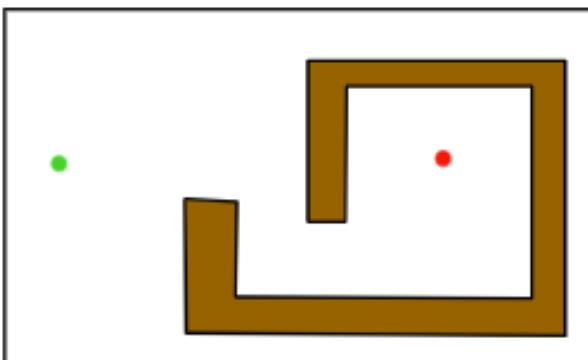
□ ALGORITMO DE EVITACIÓN BUG2

- Moverse en línea recta hacia el destino
- Si se alcanza el destino, finaliza
- Si se alcanza un obstáculo, rodearlo por la izquierda hasta alcanzar de nuevo la línea que une el origen con el destino.

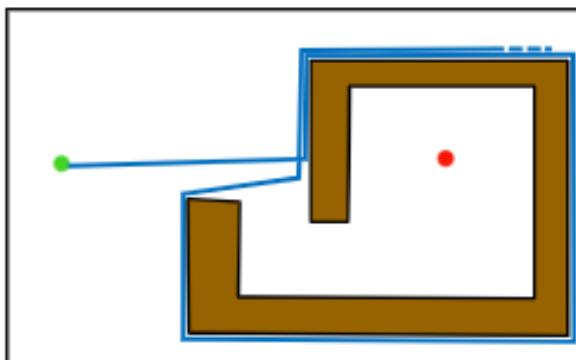


□ COMPARATIVA BUG 1 Y BUG2

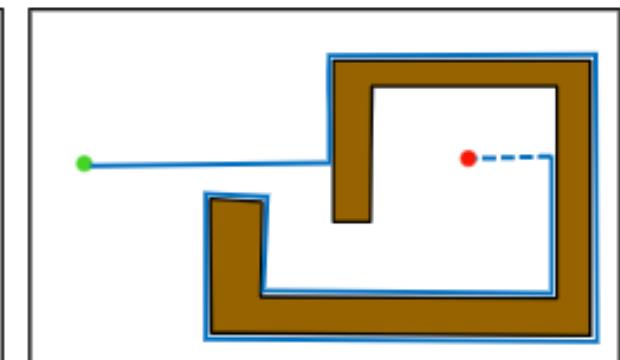
- Bug 1 es más rápido pero puede entrar en bucles y no alcanzar el destino.
- Bug 2 es más lento pero siempre llega al destino



Entorno



Bug 1

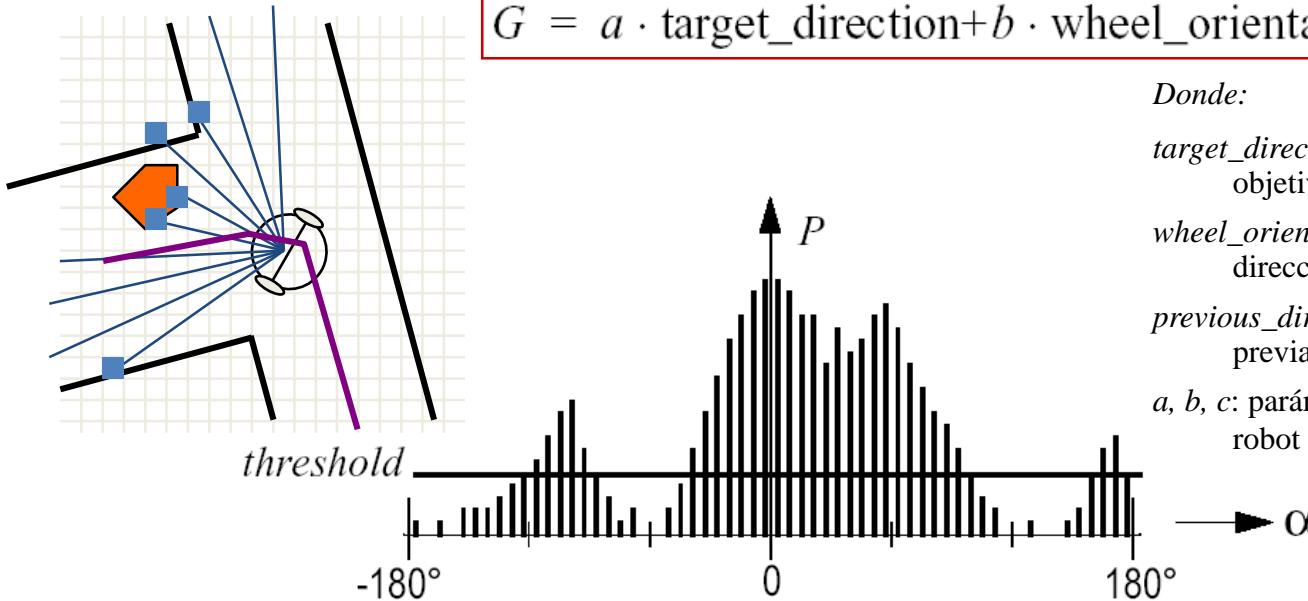


Bug 2

□ ALGORITMO DE EVITACIÓN VFH (VECTOR FIELD HISTOGRAM)

- Entorno local del robot *síncrono* se representa en una rejilla con 2DOF
 - los valores de las celdas almacenan la probabilidad de contener un obstáculo
- Se reduce a un *histograma polar* en 1DOF
 - se calcula el histograma en todas las direcciones de giro
 - se buscan todos los pasos disponibles
 - se selecciona el que menor **función de coste G** tiene

$$G = a \cdot \text{target_direction} + b \cdot \text{wheel_orientation} + c \cdot \text{previous_direction}$$



Donde:

target_direction: alineación del robot con respecto al objetivo

wheel_orientation: diferencia entre la nueva dirección y la orientación actual de las ruedas

previous_direction: diferencia entre la dirección previamente seleccionada y la nueva dirección

a, b, c : parámetros de ajuste comportamiento del robot

□ Cálculo del histograma polar:

- El valor de certidumbre de cada celda activa (c_{ij}) se trata como un vector obstáculo con una dirección (β_{ij}) y magnitud (m_{ij})

$$\beta_{ij} = \tan^{-1} \frac{y_j - y_0}{x_i - x_0} \quad m_{ij} = (c_{ij}^*)^2 (a - b \cdot d_{ij})$$

Donde:

β_{ij} = dirección de la celda activa

x_0, y_0 = coordenadas del centro del robot

x_i, y_j = coordenadas de la celda activa

m_{ij} = magnitud del vector obstáculo

c_{ij}^* = valor de certidumbre de celda activa (c_{ij})

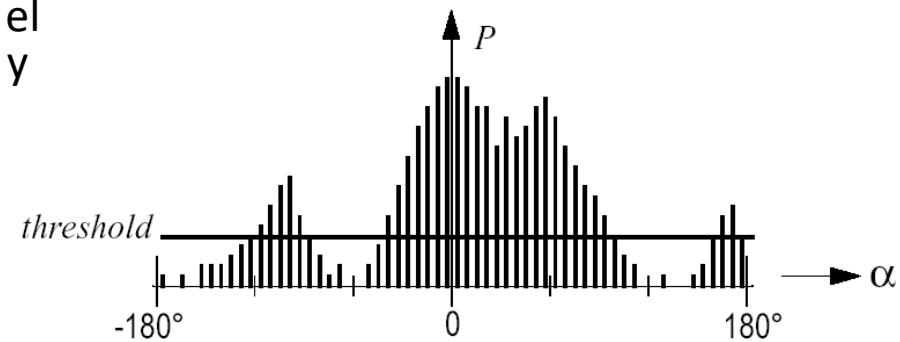
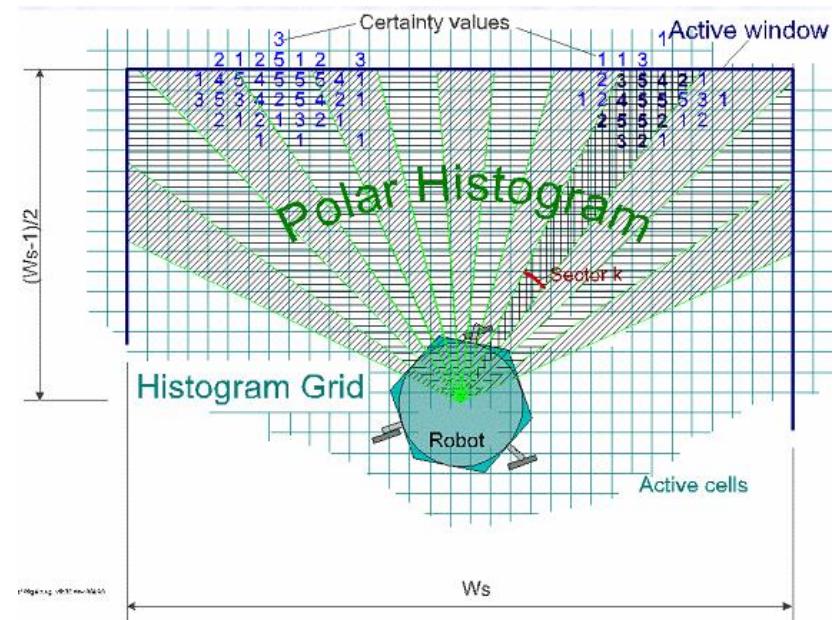
a, b = ctes. positivas

d_{ij} = distancia entre celda activa y centro robot

- Se construye el histograma obteniendo el sector k que contiene cada celda activa y el valor del histograma se obtiene de sumar las magnitudes de los vectores obstáculos en cada sector

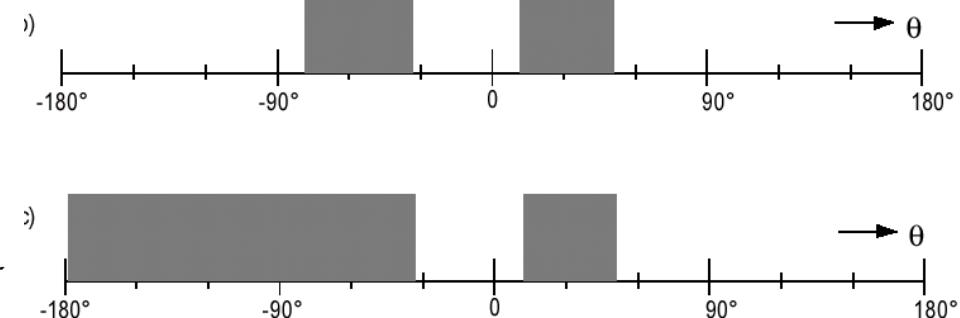
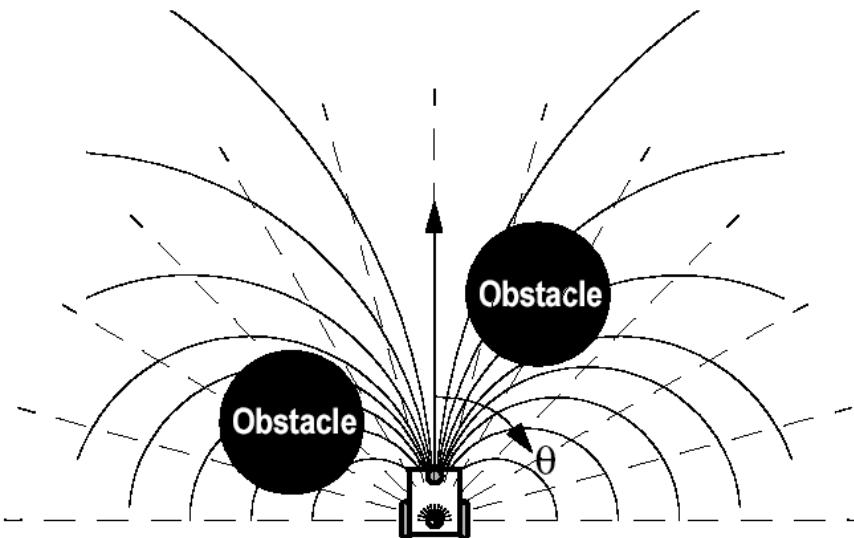
$$k = \text{INT}(\beta_{ij} / \alpha)$$

$$P = h_k = \sum_{i,j} m_{ij}$$

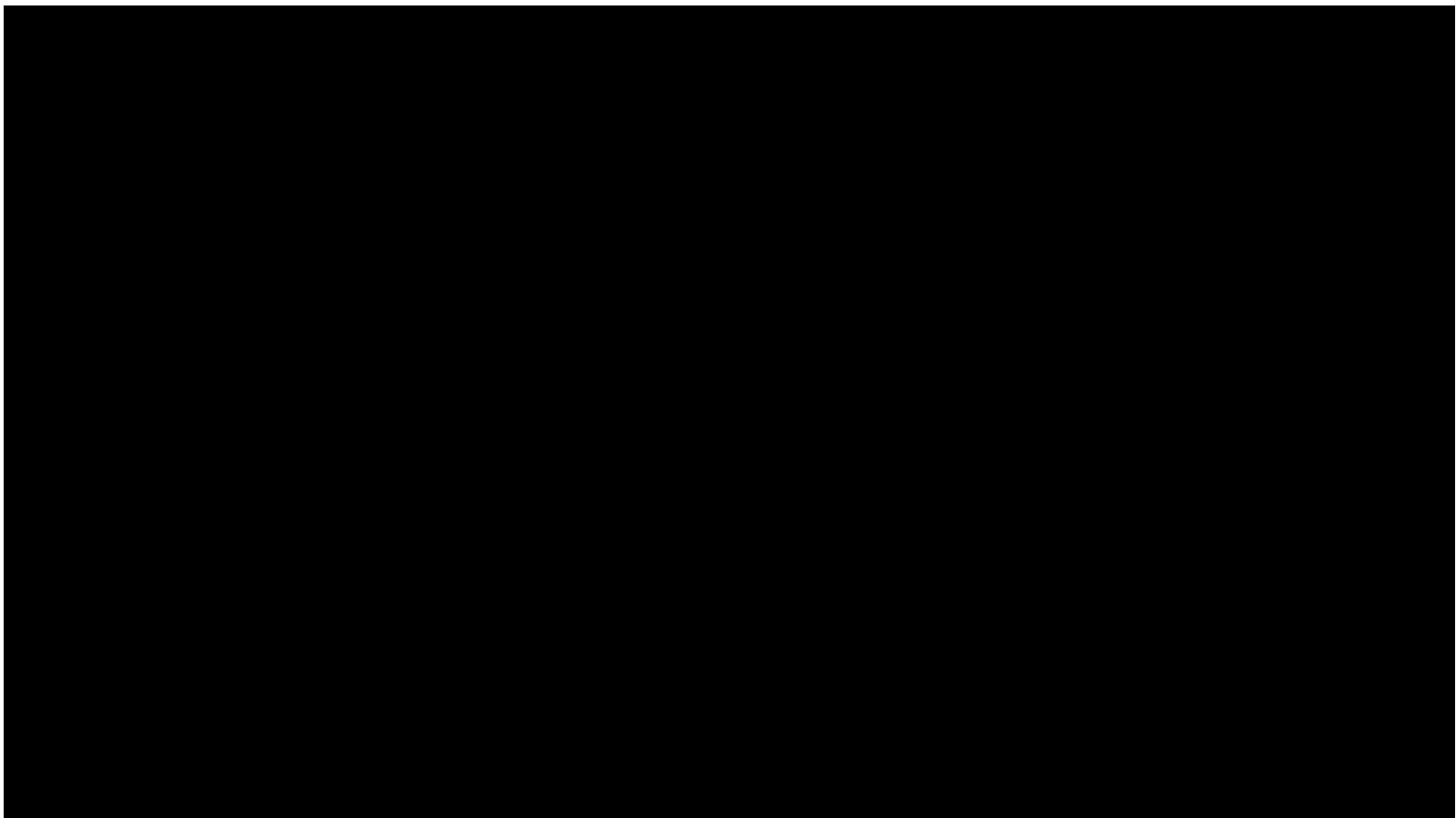


□ ALGORITMO DE EVITACIÓN VFH+

- Se realiza una **simplificación** en cuanto a la capacidad para moverse del robot (su cinemática).
 - el robot sólo se mueve en rectas o arcos
 - los obstáculos que **bloquean** una dirección dada también bloquean todas las posibles **trayectorias** a través de esa dirección



□ Ejemplo VFH:



□ ALGORITMO DE EVITACIÓN ND (NEARNESS DIAGRAM)

- Entornos densos, altamente poblados y complejos con robot *síncrono*
- Estrategia “divide y vencerás”
- Emplean paradigma “*situated-activity*”: descripción en nivel simbólico: puede ser implementado mediante diversas técnicas
- [Minguez, Montano, 2004] proponen ND como implementación basada en geometría



Paradigma “situated-activity”:

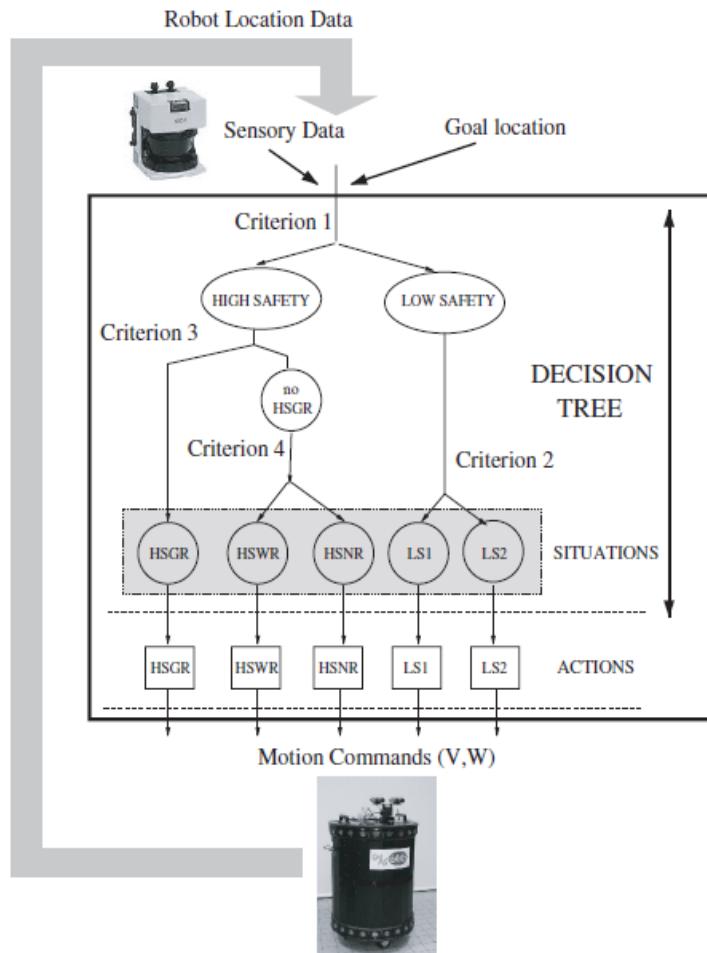
Requerimientos:

1. *Las situaciones tienen que ser identificables por la percepción, exclusivas y completas*
2. *Cada acción tiene que resolver el problema de la situación individual y completamente*

Ventajas:

1. *El paradigma resuelve el proceso percepción-acción*
2. *La estrategia “divide y vencerás” reduce la dificultad de la tarea*
3. *No tiene el problema de la coordinación acciones de tiempo real, ya que las acciones resuelven el problema de la situación y son completas*

□ ALGORITMO DE EVITACIÓN ND (NEARNESS DIAGRAM)



*Conjunto de situaciones
(HSGR, HSWR, HSNR, LS1, LS2)*

Criterio 1: seguridad (“Low Safety”, “High Safety”)

Criterio 2: distribución del obstáculo peligroso

1. *Low Safety 1 (LS1): el obstáculo en la zona de seguridad está solo en un lado del hueco del área de paso*

2. *Low Safety 2 (LS2): el obstáculo en la zona de seguridad está en ambos lados del hueco del área de paso*

Criterio 3: destino en el área de paso

3. *High Safety Goal in Region (HSGR): el destino está en el área de paso libre*

Criterio 4: ancho del área de paso

4. *High Safety Wide Region (HSWR): el destino está en un área de paso ancha*

5. *High Safety Narrow Region (HSNR): el destino está en un área de paso estrecha*

□ ALGORITMO DE EVITACIÓN ND (NEARNESS DIAGRAM)

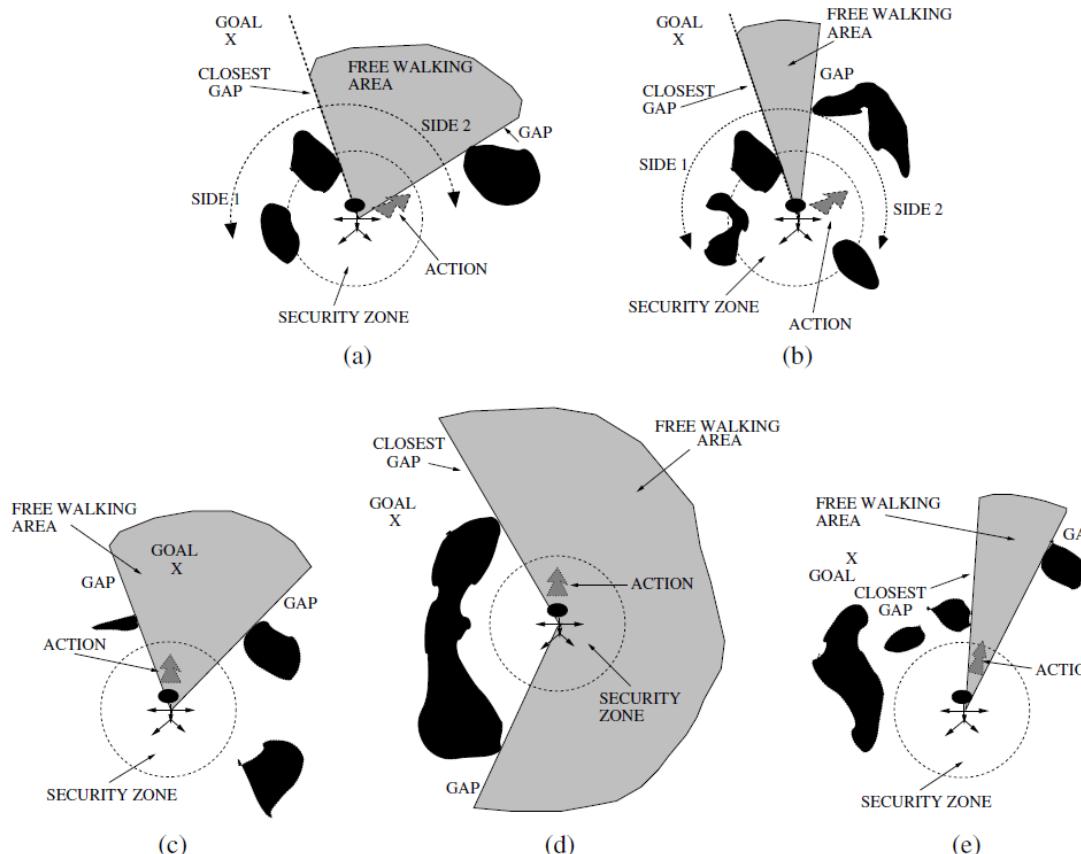


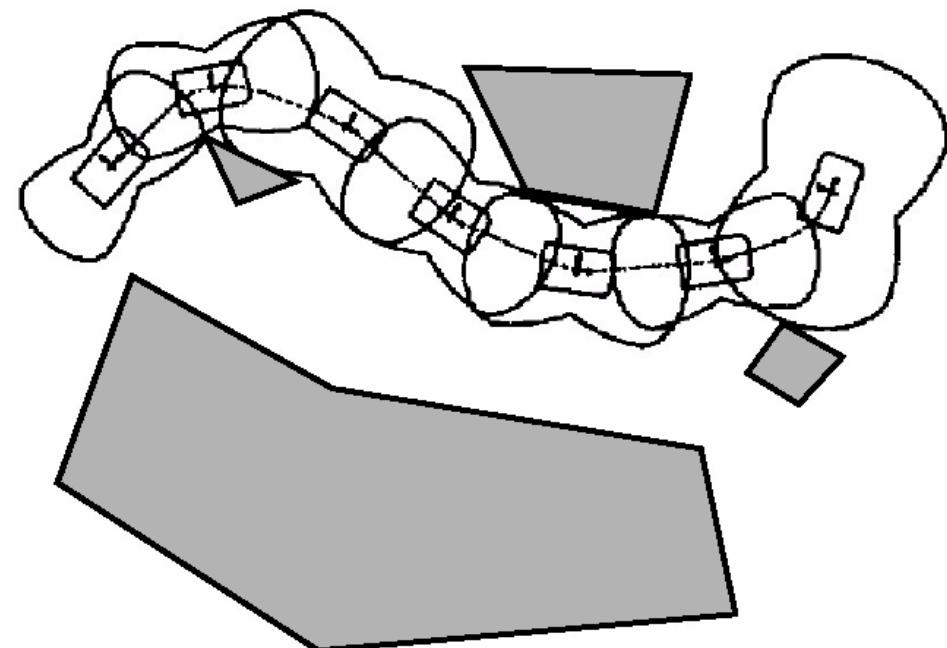
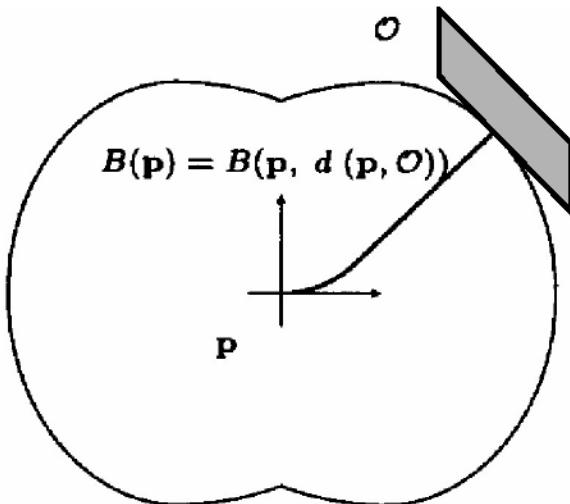
Fig. 3. (a) LS1 situación/acción ejemplo. (b) LS2 situación/acción ejemplo. (c) HSGR situación/acción ejemplo. (d) HSWR situación/acción ejemplo. (e) HSNR situación/acción ejemplo.

*Conjunto de acciones
(HSGR,HSWR,HSNR,LS1,LS2)*

1. *Low Safety 1 (LS1): alejar el robot del obstáculo más cercano pero por el área libre hacia destino*
2. *Low Safety 2 (LS2): centrar robot entre obstáculos más cercanos hacia destino*
3. *High Safety Goal in Region (HSGR): mover robot hacia destino*
4. *High Safety Wide Region (HSWR): mover robot a lo largo del obstáculo*
5. *High Safety Narrow Region (HSNR): mover robot por la parte central del área de paso*

□ ALGORITMO DE EVITACIÓN BUBBLE

- Bubble (burbuja) = máximo espacio libre alrededor del robot sin riesgo de colisión
 - se genera utilizando la distancia al objeto y un modelo simplificado del robot
 - las burbujas se utilizan para formar una banda continua de burbujas que conectan el punto inicial con el objetivo

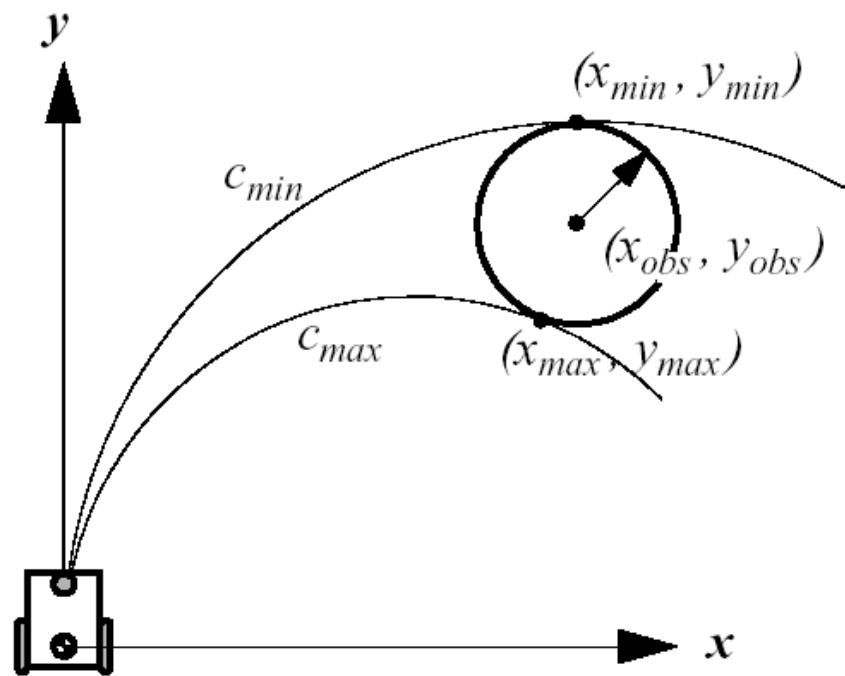


□ CURVATURE VELOCITY METHOD (CVM)

- Se imponen **restricciones** físicas al robot y el entorno en cuanto a la **cinemática** y las **velocidades** lineal y angular (v, ω) del robot
- Se supone que el robot navega describiendo arcos de curvatura

$$c = \omega / v$$

- Se imponen **restricciones** de **aceleración** máxima del robot
- Robot se toma como un punto y los **obstáculos** son **transformados** en círculos
- Se trabaja en un espacio transformado de velocidad
- Se define una **función objetivo** para seleccionar la velocidad óptima



$$f(tv, rv) = \alpha_1 \cdot dist(tv, rv) + \alpha_2 \cdot head(rv)$$

$$+ \alpha_3 \cdot speed(tv)$$

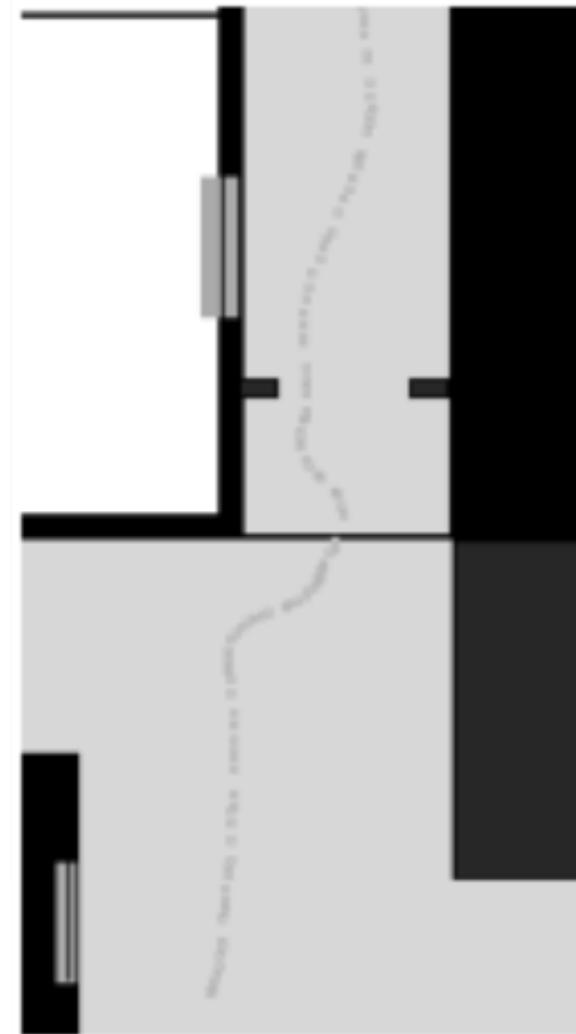
$$dist(tv, rv) = d(tv, rv, OBS)/L$$

$$head(rv) = 1 - |\theta_e - rv \cdot T_c|/\pi$$

$$speed(tv) = tv/tv_{max}$$

□ Problemas del CVM:

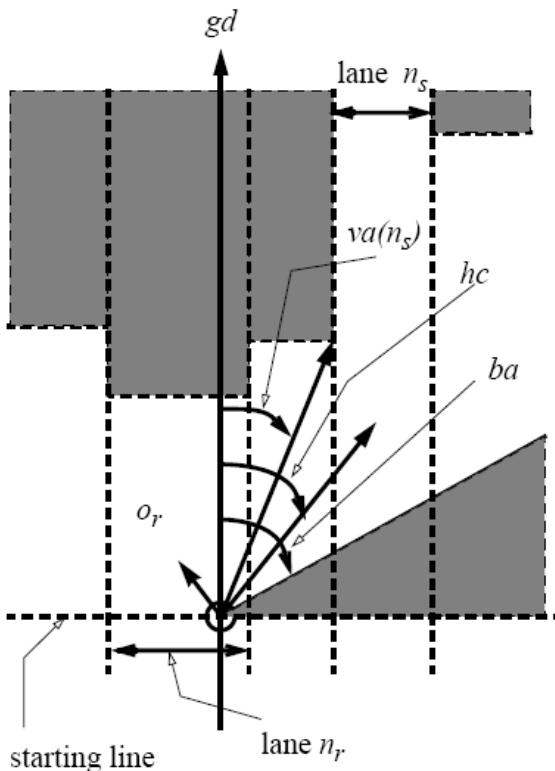
- Navegación con entornos basados en pasillos estrechos
- En algunos casos el robot se dará la vuelta y no entrará en el pasillo



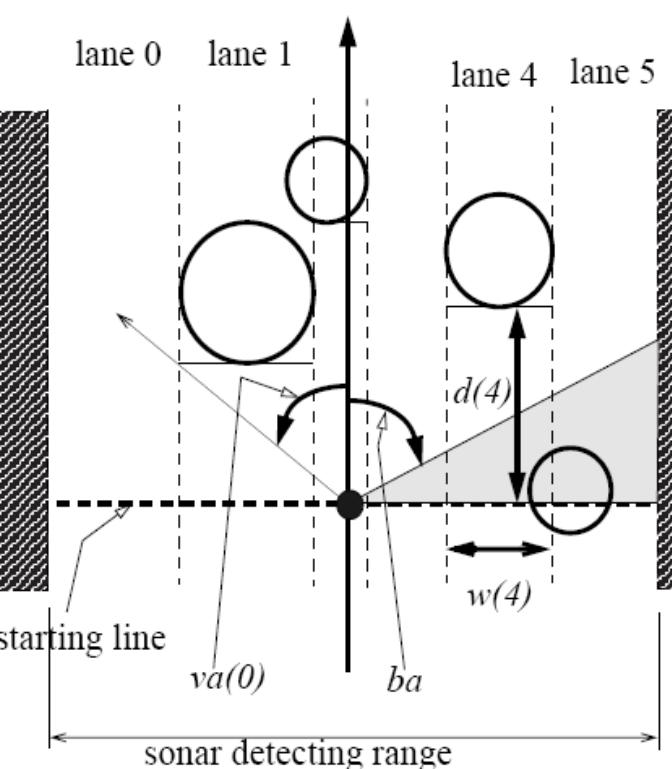
□ LANE CURVATURE METHOD (LCM)

- Mejoras al CVM:
 - Mejora el método porque considera que el robot **no navega únicamente en arcos**
 - Los pasillos/carriles se calculan evaluando la **longitud y anchura** del carril al objeto más cercano
 - El carril con mejores propiedades se elige usando una **función objetivo**
- Destacado:
 - Mejor funcionamiento en la navegación con **pasos estrechos**
 - Sigue teniendo el problema de **mínimos locales**

□ Evaluación ángulo de visión (va), bloqueo (ba) y dirección (hc)



□ Determinación de carriles



Donde:

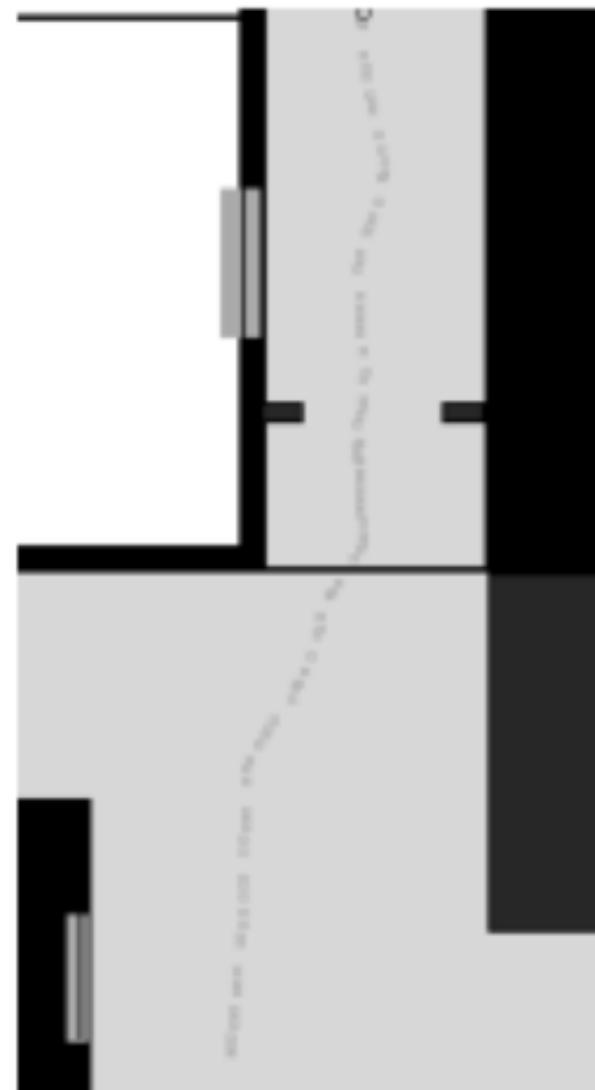
$w(i)$ = anchura del carril “i-ésimo”

$d(i)$ = distancia libre de colisión al obstáculo “i-ésimo”

$va(i)$ = dirección de paso al carril libre de colisión
 ba = ángulo de bloqueo

□ Mejora del LCM:

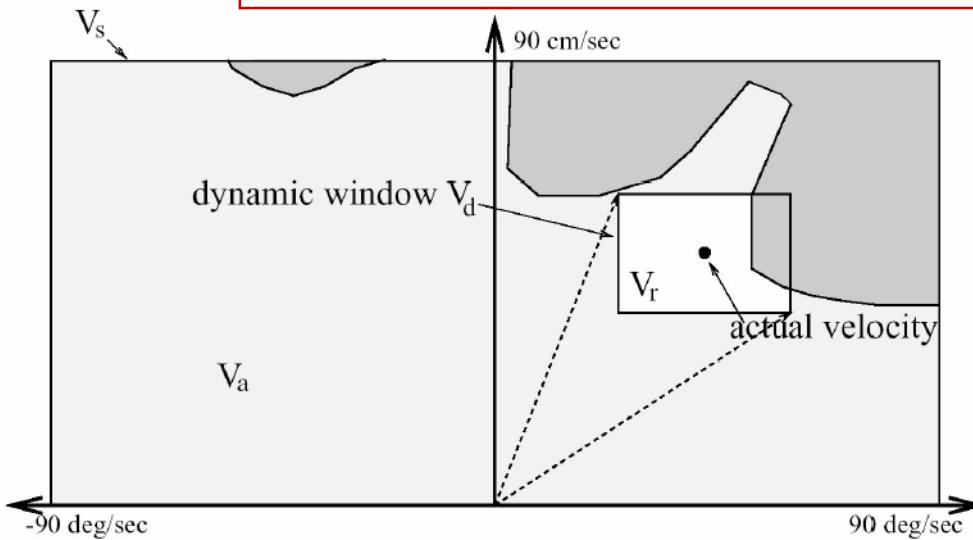
- Navegación en entornos basados en pasillos estrechos mejor que con CVM



□ ALGORITMO DE VENTANA DINÁMICA

- Se utiliza la cinemática del robot para elegir un **espacio de velocidad** adecuado
- Espacio velocidad -> un tipo de espacio de configuración
- Se asume que el robot se mueve en **arcos**
- **Asegura** que el robot **se detiene** antes de golpear ningún obstáculo
- Se elige una **función objetivo** para seleccionar la velocidad óptima

$$O = a \cdot heading(v, \omega) + b \cdot velocity(v, \omega) + c \cdot dist(v, \omega)$$



Donde:

heading: orientación del robot con respecto al objetivo

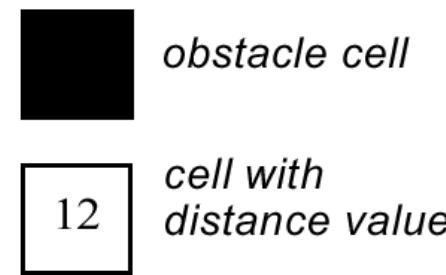
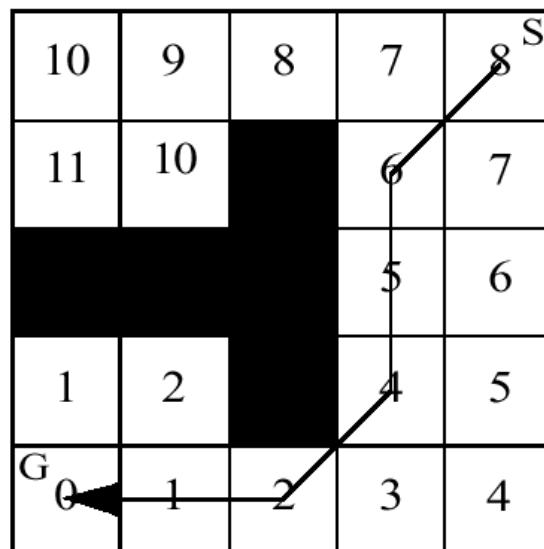
velocity: velocidad de avance del robot (preferiblemente un valor alto)

dist: distancia al obstáculo más cercano

a, b, c: parámetros de ajuste comportamiento del robot

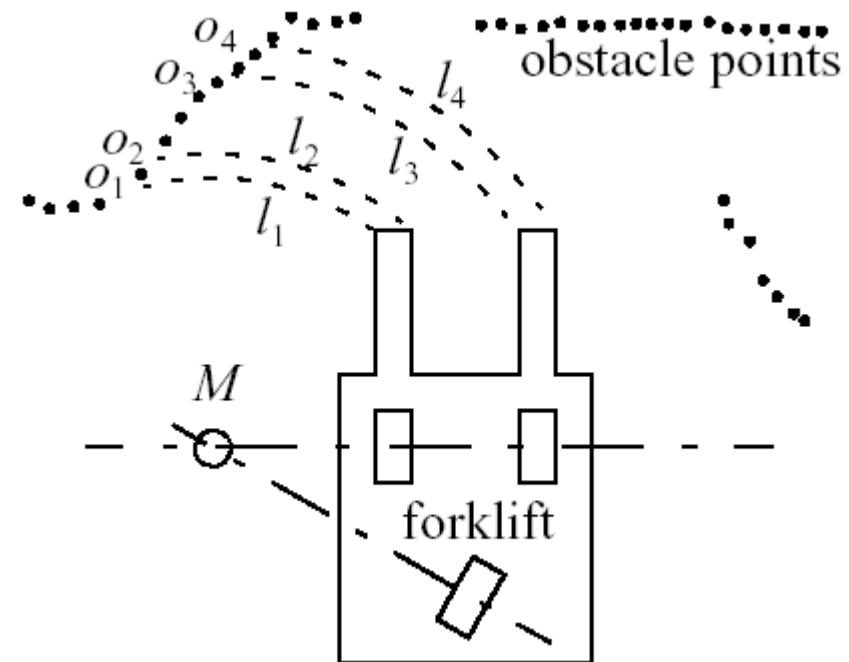
□ APROXIMACIÓN DE VENTANA DINÁMICA GLOBAL

- Se hace por medio de **añadir** una función libre de mínimos L , denominada **NF1** (Wave-Propagation), a la función objetivo O presentada en el método anterior.
- La rejilla de ocupación se **actualiza** en base a la medida de los sensores de distancia (láser, ultrasonidos)



□ APROXIMACIÓN SCHLEGEL

- Es una **variación** sobre el algoritmo de ventana dinámica
- Tiene en cuenta la **forma** del robot
- Movimiento del robot en forma de **arcos**
- Planificador global **NF1**
- Funcionamiento en **tiempo real** conseguido con **tablas precalculadas**



Comparativa de algoritmos (I)

Bug			Bubble band		Vector Field Histogram (VFH)			method
Tangent Bug [82]	Bug2 [101, 102]	Bug1 [101, 102]	Bubble band [85]	Elastic band [86]	VFH* [149]	VFH+ [92, 150]	VFH [43]	
point	point	point	C-space	C-space	circle	circle	simplistic	shape
			exact		basic	basic		kinematics
					simplistic	simplistic		dynamics
local	local	local	local	global	essentially local	local	local	view
local tangent graph					histogram grid	histogram grid	histogram grid	local map
			polygonal	polygonal				global map
			required	required				path planner
range	tactile	tactile			sonars	sonars	range	sensors
			various	various	nonholonomic (GuideCane)	nonholonomic (GuideCane)	synchro-drive (hexagonal)	tested robots
					6 ... 242 ms	6 ms	27 ms	cycle time
					66 MHz, 486 PC	66 MHz, 486 PC	20 MHz, 386 AT	architecture
efficient in many cases, robust	inefficient, robust	very inefficient, robust			fewer local minima	local minima	local minima, oscillating trajectories	remarks

Comparativa de algoritmos (II)

Dynamic window		Curvature velocity		method	model fidelity	
Global dynamic window [44]	Dynamic window approach [69]	Lane curvature method [87]	Curvature velocity method [135]			
circle	circle	circle	circle	shape		
(holonomic)	exact	exact	exact	kinematics		
basic	basic	basic	basic	dynamics		
global	local	local	local	view		
	obstacle line field	histogram grid	histogram grid	local map		
C-space grid				global map		
NF1				path planner		
180° FOV SCK laser scanner	24 sonars ring, 56 infrared ring, stereo camera	24 sonars ring, 30° FOV laser	24 sonars ring, 30° FOV laser	sensors		
holonomic (circular)	synchro-drive (circular)	synchro-drive (circular)	synchro-drive (circular)	tested robots		
6.7 ms	250 ms	125 ms	125 ms	cycle time	performance	
450 MHz, PC	486 PC	200 MHz, Pentium	66 MHz, 486 PC	architecture		
turning into corridors	local minima	local minima	local minima, turning into corridors	remarks		

Comparativa de algoritmos (III)

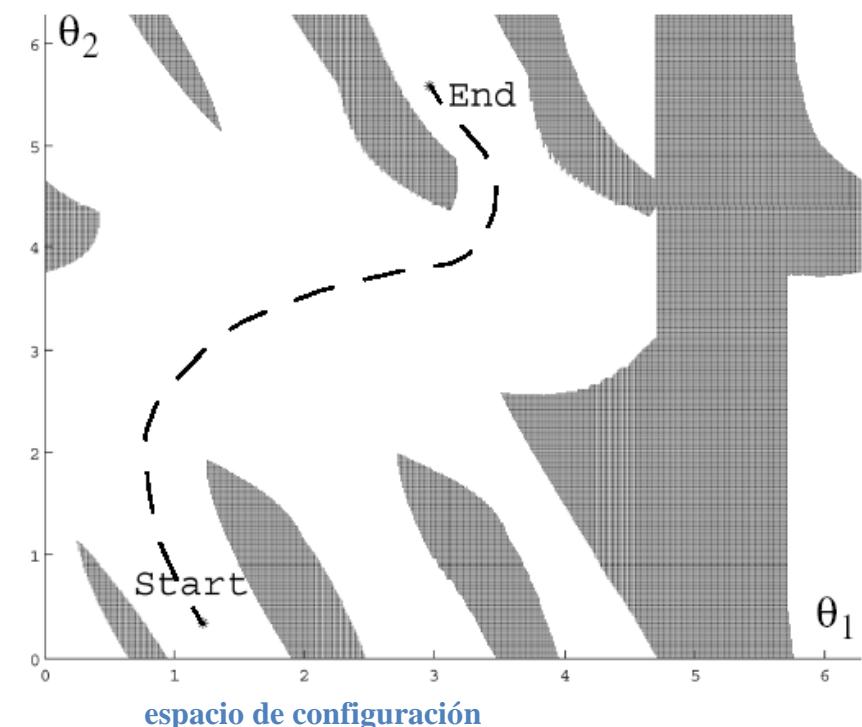
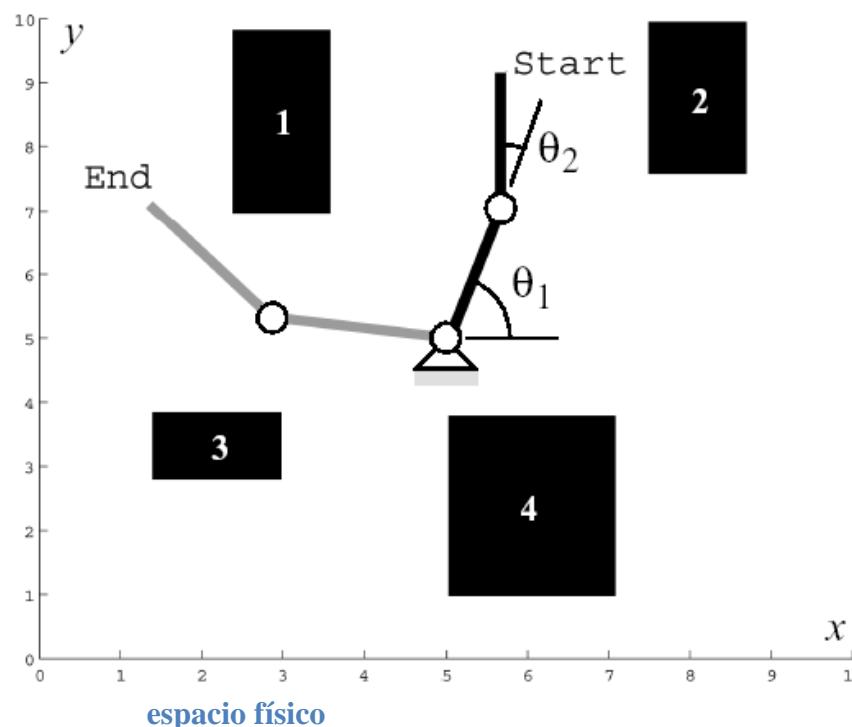
Other					method	
Gradient method [89]	Global nearness diagram [110]	Nearness diagram [107, 108]	ASL approach [122]	Schlegel [128]		
circle	circle (but general formulation)	circle (but general formulation)	polygon	polygon	shape	model fidelity
exact	(holonomic)	(holonomic)	exact	exact	kinematics	
basic			basic	basic	dynamics	
global	global	local	local	global	view	
	grid		grid		local map	other requisites
local perceptual space	NF1			grid	global map	
fused			graph (topological), NF1	wavefront	path planner	
180° FOV distance sensor	180° FOV SCK laser scanner	180° FOV SCK laser scanner	2x 180° FOV SCK laser scanner	360° FOV laser scanner	sensors	
nonholonomic (approx. circle)	holonomic (circular)	holonomic (circular)	differential drive (octagonal, rectangular)	synchrodrive (circular), tricycle (forklift)	tested robots	performance
100 ms (core algorithm: 10 ms)			100 ms (core algorithm: 22 ms)		cycle time	
266 MHz, Pentium			380 MHz, G3		architecture	
		local minima	turning into corridors	allows shape change	remarks	



3. Planificación global

- **Planificación global:** proceso de **decisión** del camino a tomar para llegar a un punto de destino (**goal**)
- Necesario: existe un mapa del entorno lo suficientemente bueno para realizar la navegación.
 - **Topológico**, **métrico** o una **mezcla** de ambos.
 - Representación del entorno **grafo**, **celdas** o un **campo de potencial**.
 - Las posiciones discretas o celdas resultantes permiten realizar algoritmos de planificación estándar.
- Ejemplos de algoritmos clásicos (determinísticos):
 - Basados en mapa:
 - Grafos de visibilidad
 - Diagrama de Voronoi
 - Descomposición en Celdas -> Grafos de conectividad
 - Campos de Potencial
- Ejemplos de algoritmos probabilísticos:
 - Procesos de Decisión de Markov (MDP)
 - Procesos de Decisión de Markov Parcialmente Observables (POMDP)

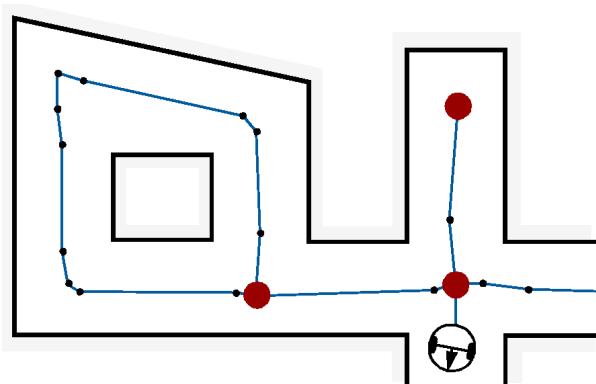
- **ESPACIO DE CONFIGURACIÓN C :** concepto heredado de la robótica industrial -> conjunto de k valores instantáneos de q_i (estados o configuraciones del robot) que puede tomar el robot



- ¿Espacio de configuración en robótica móvil? $C = C_{free} + C_{obs}$

1. Construir Grafo

- Identificar un conjunto de rutas dentro del espacio libre



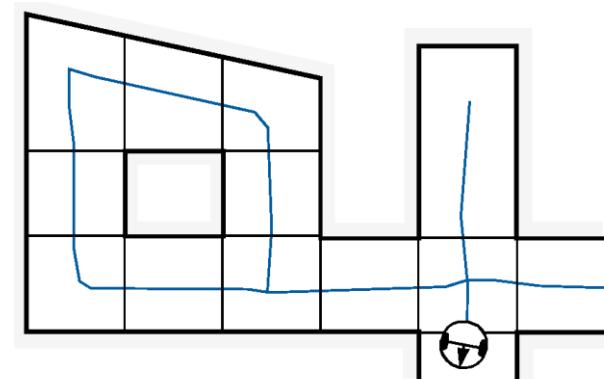
¿Dónde ponemos los nodos?

- Basado en topológico:
 - En posiciones distintivas
- Basado en métrico:
 - Donde las características se hacen visibles o son ocultas



2. Descomposición en Celdas

- Discriminar entre espacio libre y ocupado



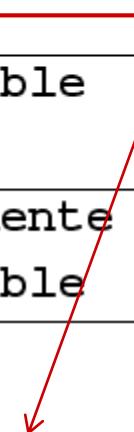
¿Dónde ponemos los límites de las celdas?

- Topológico - métrico:
 - Donde las características se hacen visibles o son ocultas

3. Campo de Potencial

- Imponer una función matemática sobre el espacio disponible

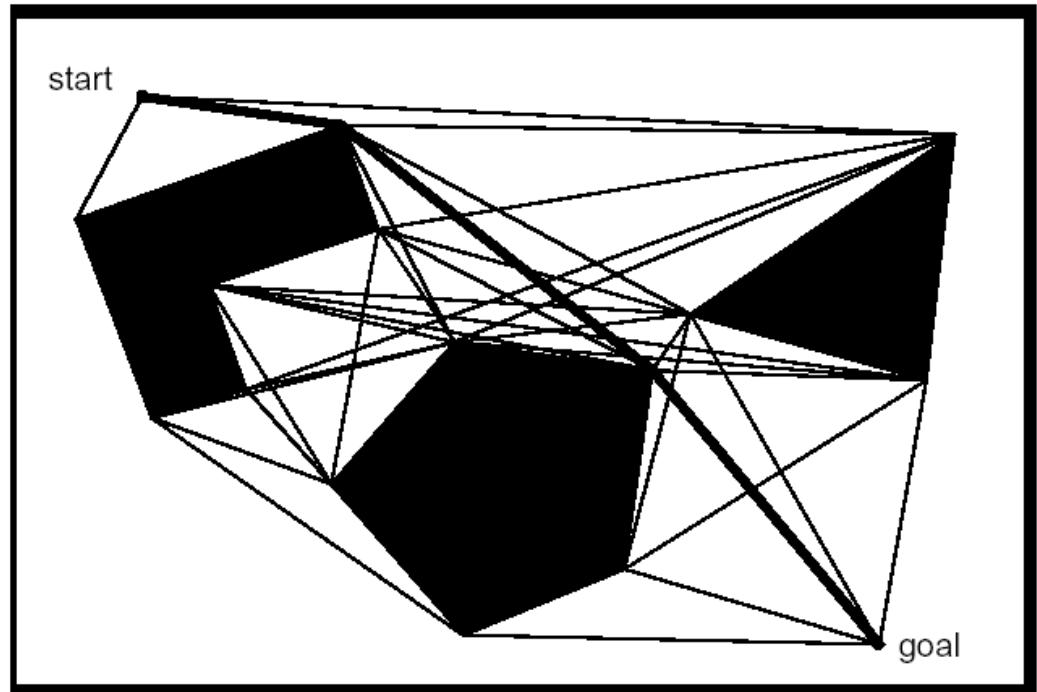
	Estado (siempre discreto)	Modelo de actuación
Planificación clásica	Observable	Determinístico
MDP	Observable	Estocástico
POMDP	Parcialmente Observable	Estocástico



Grafos de visibilidad
 Diagrama de Voronoi
 Descomposición en Celdas -> Grafos de conectividad
 Campos de Potencial

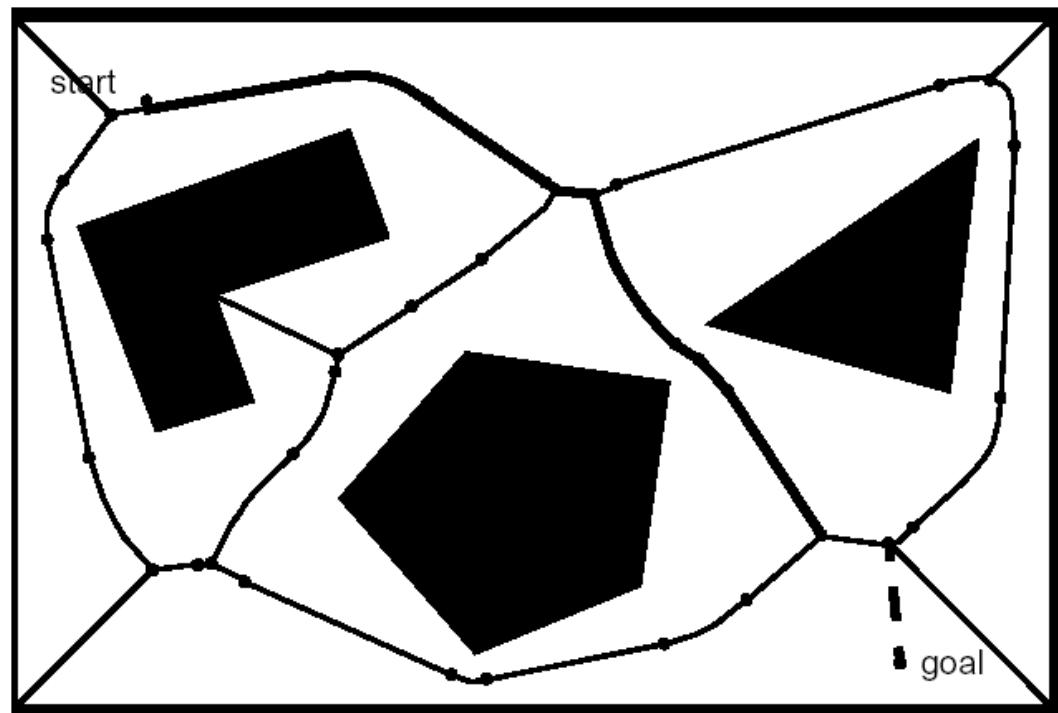
□ GRAFO DE VISIBILIDAD

- Se trazan segmentos desde todos los vértices de los obstáculos hasta los vértices que son visibles, incluyendo los puntos inicial y final.
- Como ruta óptima se selecciona el camino de **longitud menor**
- **Problema:** colisiones con los obstáculos -> se **aumenta** el tamaño de los obstáculos de forma virtual



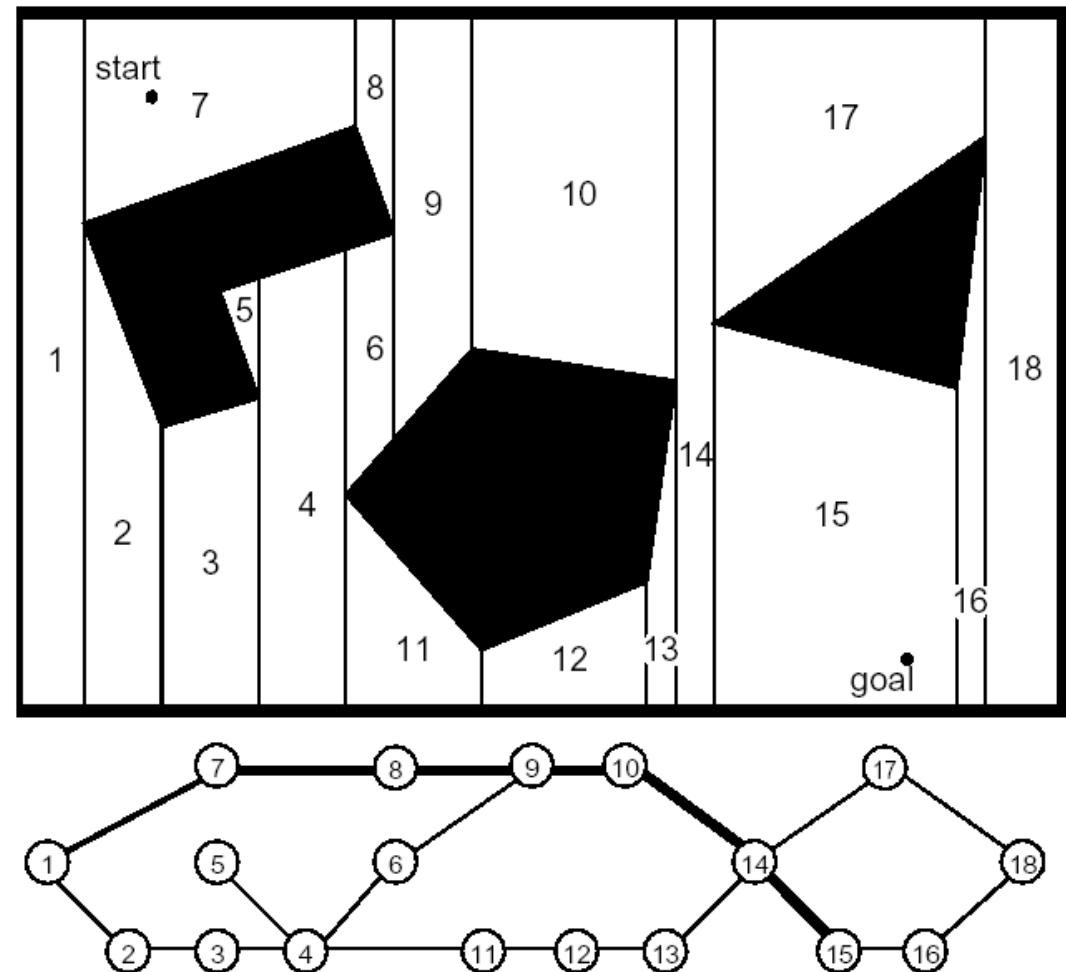
□ DIAGRAMA DE VORONOI

- Se tiende a **maximizar** la distancia entre el robot y los obstáculos del espacio de configuración
- Se calcula la distancia desde todos los puntos en el espacio libre hasta los obstáculos y se seleccionan los de mayor distancia.
- En un espacio con obstáculos poligonales se obtienen segmentos de tipo **recta** y **parábola**. Los puntos indican su unión.
- Como ruta óptima se selecciona el camino de **longitud menor**



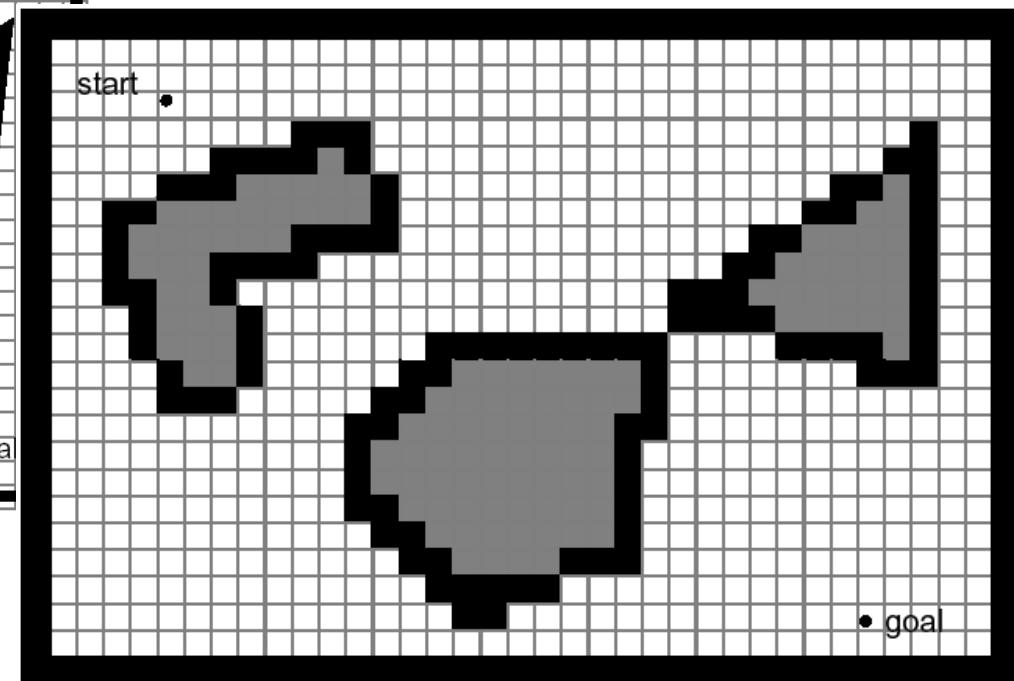
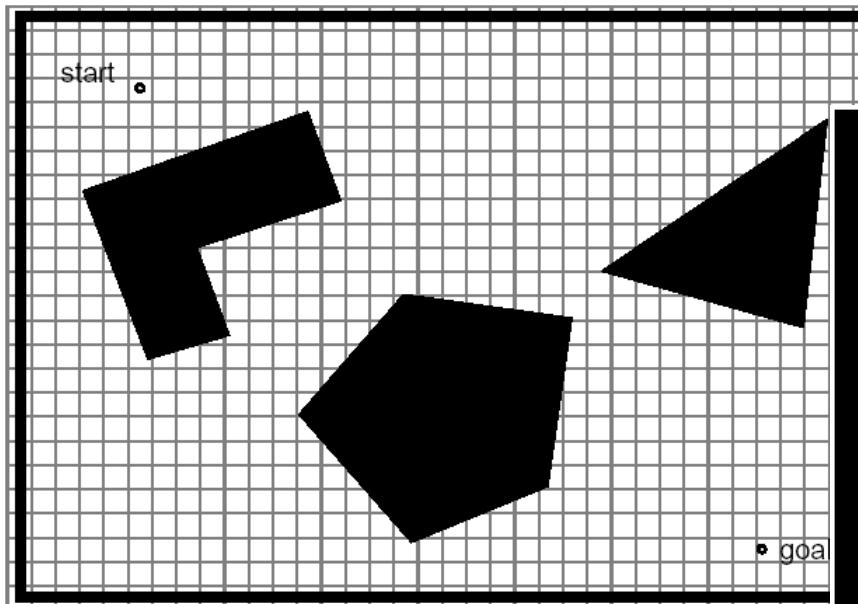
□ DESCOMPOSICIÓN EN CELDAS EXACTAS

- Se divide el espacio en regiones conectadas que se llaman **teselas**
- En la planificación hay que determinar que teselas son adyacentes y se construye un **grafo de conectividad**
- Se busca en el grafo de conectividad el camino que une las teselas que contienen los puntos de **partida** y **final**.
- Se utiliza un algoritmo de búsqueda entre todas las teselas para obtener el **camino óptimo**.
- Para pasar de una tesela a otra se pueden emplear diferentes **estrategias** (seguir una pared, navegar entre las posiciones centrales de cada celda, etc).
- Se complica la navegación local



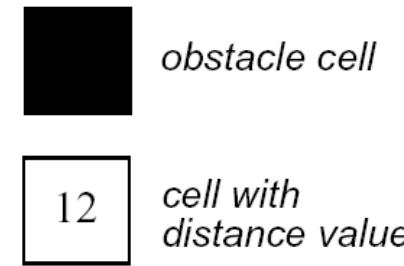
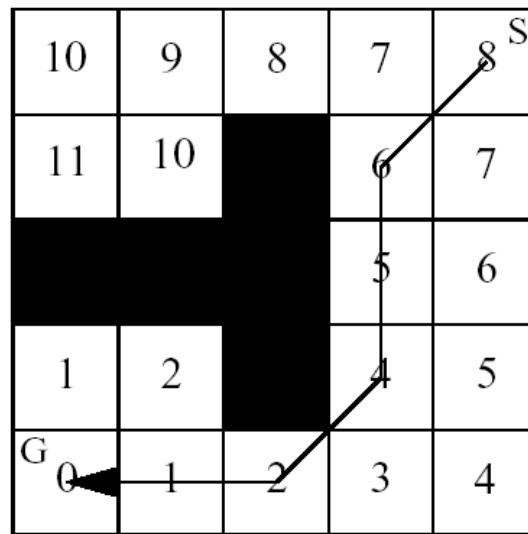
□ DESCOMPOSICIÓN EN CELDAS FIJAS

- Existen diferentes estrategias (Wavefront, Breadth-First, Depth-First,...)
- Computacionalmente tiene un coste muy bajo
- Problemas: pasos estrechos, memoria depende del tamaño del mapa

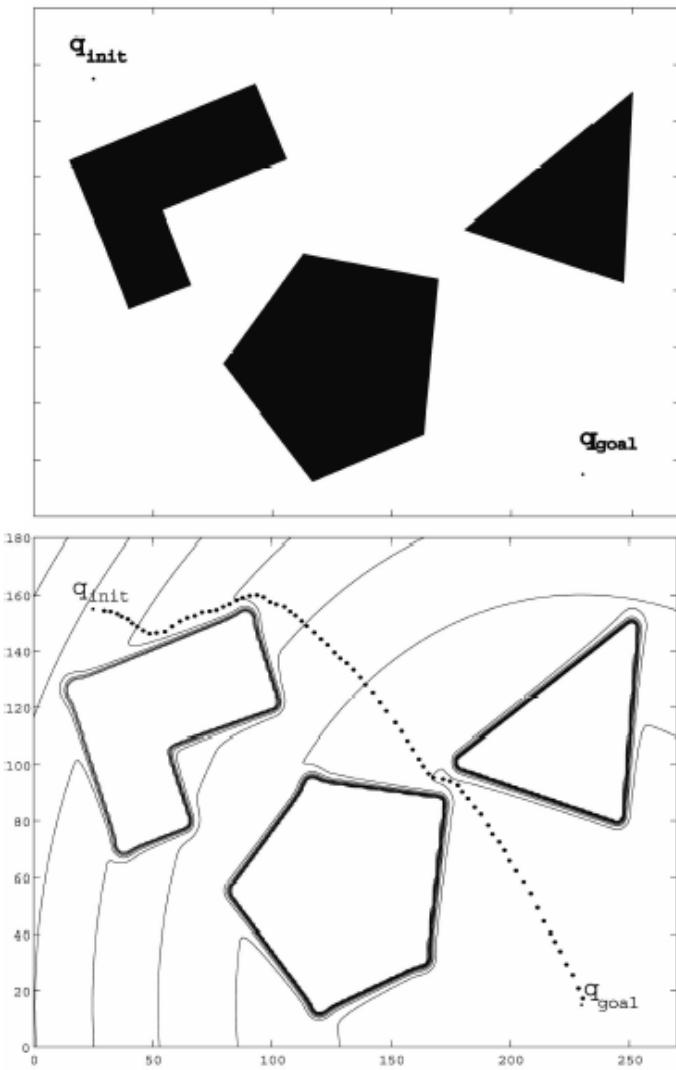


□ ESTRATEGIAS DE BÚSQUEDA EN GRAFOS/PLANOS:

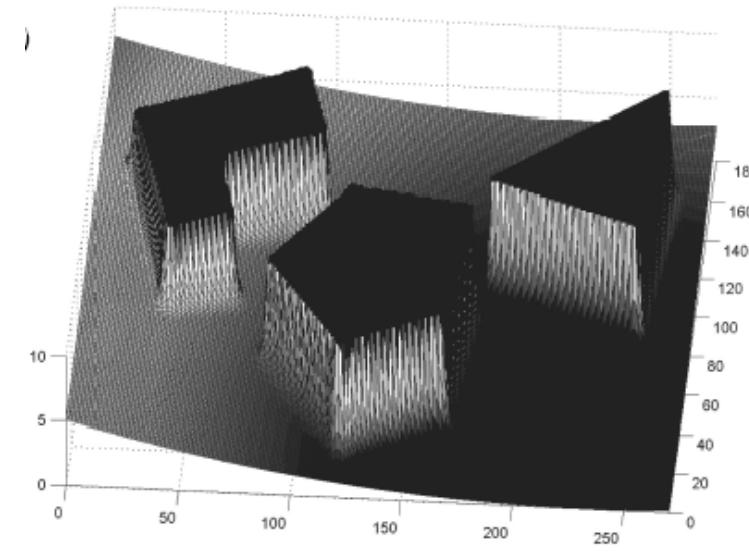
- NF1 o expansión de frente de ondas (WaveFront Expansion)



- Otras estrategias de búsqueda
 - Breadth-First
 - Depth-First
 - Greedy y A*



- El robot se trata como un punto sometido a la influencia de un **CAMPO DE POTENCIAL**.
 - El **objetivo** genera una fuerza de atracción
 - Los **obstáculos** generan fuerzas de repulsión
 - Los movimientos del robot son similares a una bola rodando cuesta abajo hacia el objetivo



- Generación de la función del campo de potencial $U(q)$:
 - Campos de: atracción (objetivo) y repulsión (obstáculos)
 - Se suman los campos
 - Las funciones deben ser derivables
- Se genera una Fuerza de campo artificial $F(q)$

$$F(q) = -\nabla U(q) = -\nabla U_{att}(q) - \nabla U_{rep}(q) = \begin{bmatrix} \frac{\partial U}{\partial x} \\ \frac{\partial U}{\partial y} \end{bmatrix}$$

- Se aplica un velocidad al robot (v_x, v_y) proporcional a la Fuerza $F(q)$ generada por el campo
 - La fuerza del campo controla al robot hasta llegar al objetivo
 - El robot se asume como un punto de masa (sin orientación)

□ Campo de potencial de atracción:

- Es una función parabólica que depende de la distancia Euclídea $\| q - q_{goal} \|$ hasta el objetivo

$$U_{att}(q) = \frac{1}{2} k_{att} \cdot \rho_{goal}^2(q)$$

k_{att} : factor de escala positivo

ρ : distancia euclídea $\| q - q_{goal} \|$

- La fuerza de atracción converge linealmente a cero a medida que se acerca al objetivo

$$\begin{aligned} F_{att}(q) &= -\nabla U_{att}(q) \\ &= -k_{att} \cdot \rho_{goal}(q) \nabla \rho_{goal}(q) \\ &= -k_{att} \cdot (q - q_{goal}) \end{aligned}$$

□ Campo de potencial de repulsión:

- Debería generar una barrera alrededor del obstáculo:
 - Más fuerte cuanto más cerca se encuentre del obstáculo
 - No debe influir si está muy lejano

$$U_{rep}(q) = \begin{cases} \frac{1}{2}k_{rep}\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)^2 & \text{if } \rho(q) \leq \rho_0 \\ 0 & \text{if } \rho(q) \geq \rho_0 \end{cases}$$

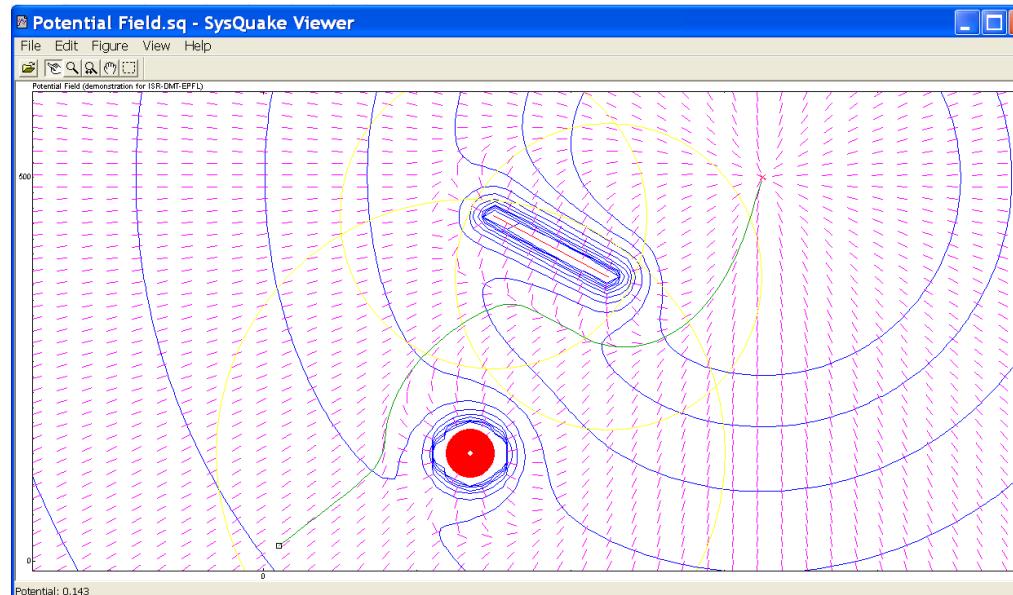
k_{rep} : factor de escala positivo
 $\rho(q)$: distancia mínima al objeto
 ρ_0 : distancia de influencia del objeto

- El campo es positivo o cero, y tiende a infinito a medida que el robot está más cerca del objeto

$$F_{rep}(q) = -\nabla U_{rep}(q) = \begin{cases} k_{rep}\left(\frac{1}{\rho(q)} - \frac{1}{\rho_0}\right)\frac{1}{\rho^2(q)}\frac{q - q_{goal}}{\rho(q)} & \text{if } \rho(q) \leq \rho_0 \\ 0 & \text{if } \rho(q) \geq \rho_0 \end{cases}$$

□ Problemas de la planificación basada en campos de potencial:

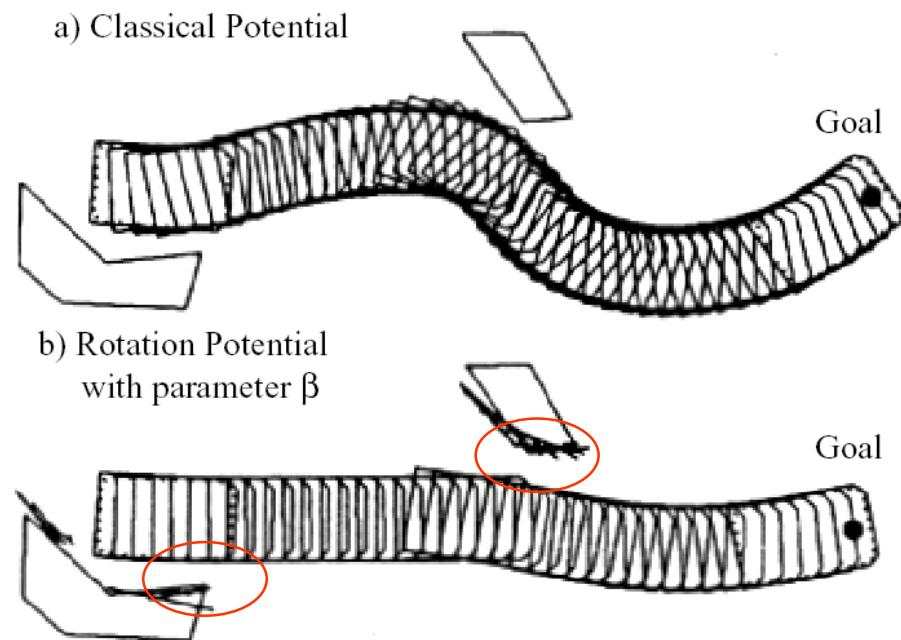
- Existen problemas de **mínimos locales**
- El problema es mucho más complejo si el robot se modela como si **no fuera un único punto de masa**
- Si el objeto es convexo se producen situaciones donde existen varias distancias mínimas → pueden existir **oscilaciones**



Sysquake Viewer

□ MÉTODO EXTENDIDO DE CAMPO DE POTENCIAL

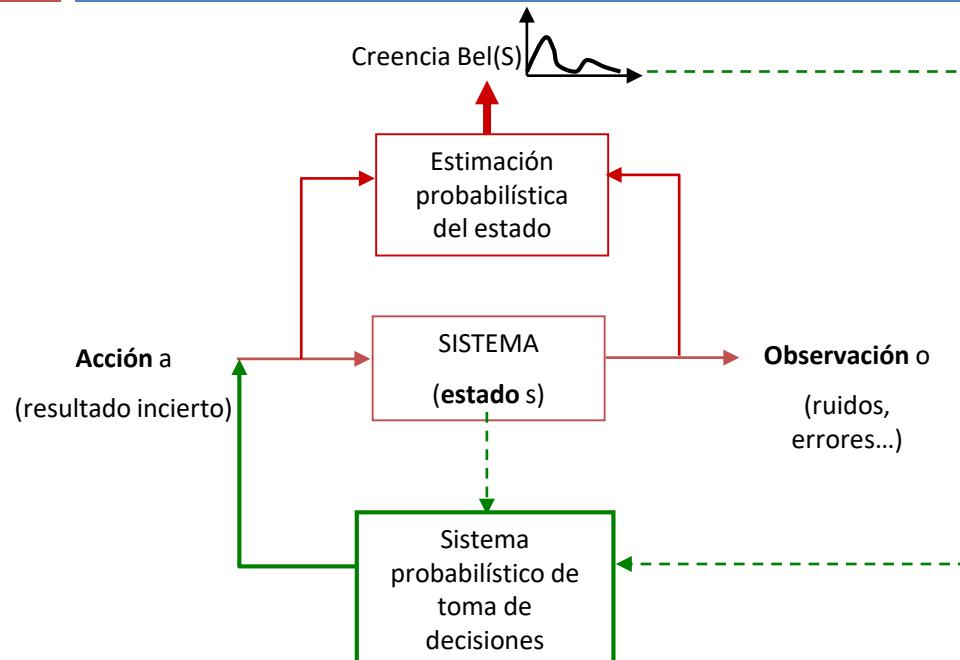
- Se introduce un campo de potencial de rotación y un campo de potencial de tarea
- Campo de potencial de rotación:
 - La fuerza es función de la orientación del robot al obstáculo
- Campo de potencial de tarea:
 - Se filtran los obstáculos que no influyen en el movimiento del robot, sólo se incluyen los que están dentro de un sector Z delante del robot



	Estado (siempre discreto)	Modelo de actuación
Planificación clásica	Observable	Determinístico
MDP	Observable	Estocástico
POMDP	Parcialmente Observable	Estocástico

TEORÍA DE
DECISIONES

El problema de la toma de decisiones



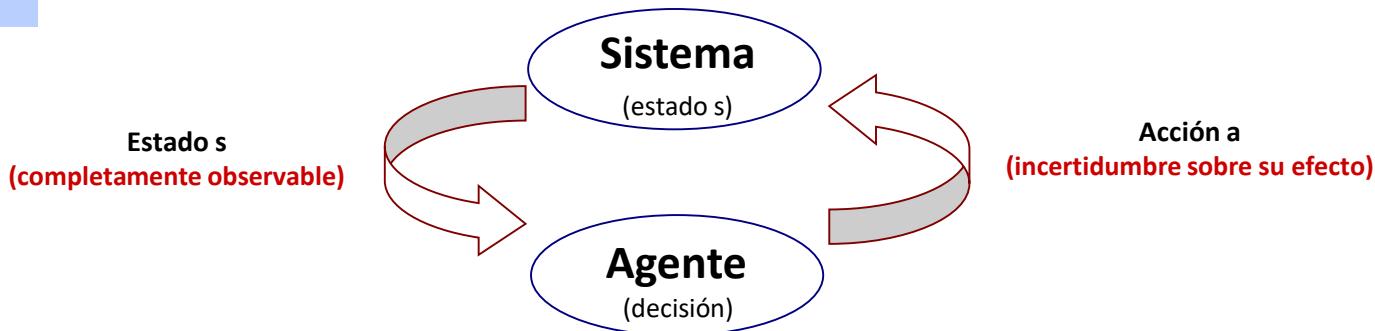
¿Qué acción realizar a continuación para que el sistema llegue a un estado deseado S_d ?

PROBLEMAS:

1. Incluso conociendo el estado actual, hay **incertidumbre sobre el resultado de las acciones**
2. Normalmente no se conoce el estado actual, sino una estimación del mismo

Se plantea un **problema de planificación bajo condiciones de incertidumbre** que afectan tanto al resultado de las acciones como a la percepción del estado

Definición



ELEMENTOS:

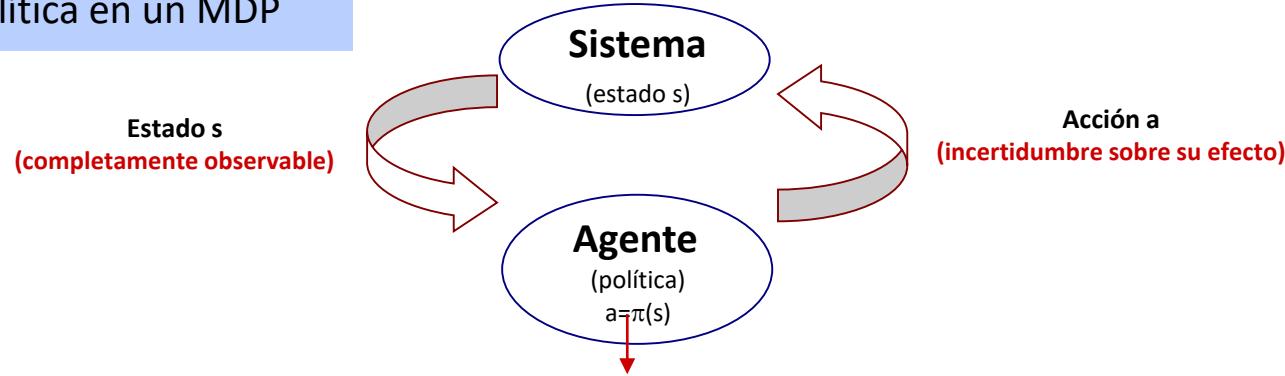
- Estados $s \in S$
- Acciones $a \in A$

Debe cumplir la propiedad de Markov:
 $p(s_{t+1} | s_0, a_0, s_1, a_1, \dots, s_t, a_t) = p(s_{t+1} | s_t, a_t)$

MODELOS:

- Modelo de transición T ($p(s'|s,a)$) \Rightarrow *incertidumbre de las acciones*
- Modelo de recompensa R ($r(s,a)$) \Rightarrow *utilidad de las acciones para conseguir el objetivo*

Definición de Política en un MDP

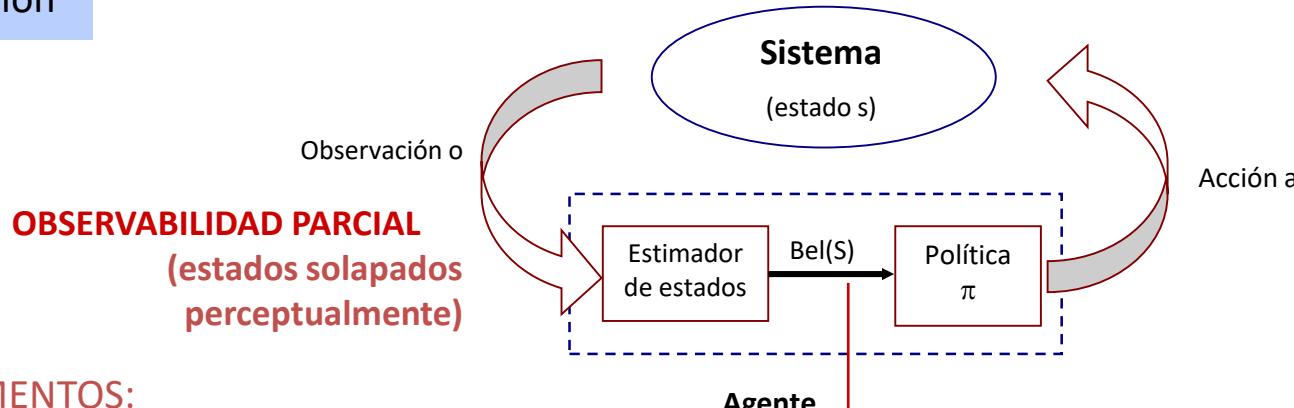


Una política es una función que asigna una acción a cada estado

Cálculo de Políticas en MDPs

- Se trata de encontrar la política óptima $a=\pi^*(s)$ que maximiza la recompensa futura
- Existen varios algoritmos, entre los que cabe destacar el **Algoritmo de Iteración de Valores**

Definición



ELEMENTOS:

- Estados $s \in S$
- Acciones $a \in A$
- **Observaciones** $o \in O$

MODELOS:

- Modelo de transición T ($p(s'|s,a)$) \Rightarrow incertidumbre de las acciones
- Modelo de recompensa R ($r(s,a)$) \Rightarrow utilidad de las acciones para conseguir el objetivo
- **Modelo de observación** ϑ ($p(o|s)$) \Rightarrow incert. en las observaciones y SOLAPE PERCEPTUAL

Cálculo de políticas en POMDPs

Problema **mucho más complejo que en un MDP.** $a=\pi(Bel)$

$a=\pi^*(Bel)$ es un problema computacionalmente intratable (infinitas posibles Bel)

Soluciones adoptadas en robótica
(métodos heurísticos aproximados)

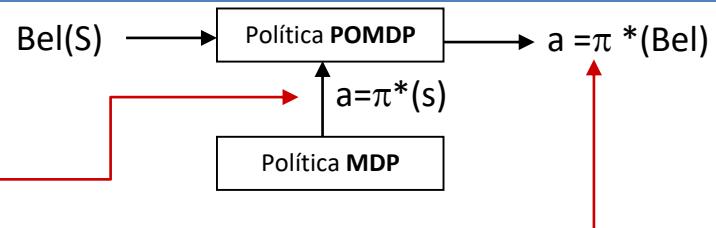
- Planificación por suposición
- **Métodos basados en MDPs**
- Aprendizaje por refuerzo

Métodos basados en MDPs

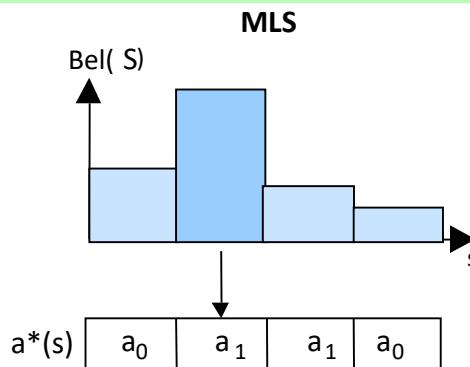
Consiste en dividir el problema en dos partes:

1. Resolver el MDP subyacente $a = \pi^*(s)$

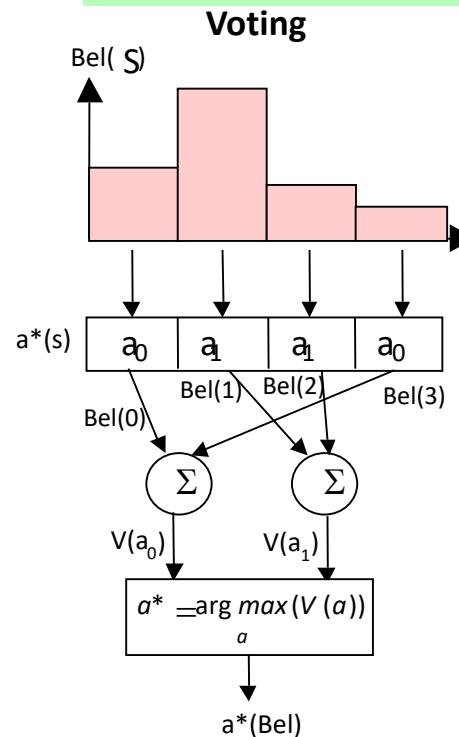
2. Seleccionar la acción final a partir de $\text{Bel}(S)$ y $\pi^*(s) \Rightarrow$ **MÉTODOS HEURÍSTICOS:**



1. Método del Estado Más Probable



2. Método de Votaciones



Aprendizaje en POMDPs

¿Qué se puede aprender de un POMDP?

1. **Su estructura** (Número de estados, “topología” o “reglas lógicas” para deducir las matrices de transición y observación, etc.)

No hay muchos trabajos en esta línea.

2. **Sus parámetros** (entradas de las matrices de transición T y de observación Φ)

Algoritmo más utilizado: **Algoritmo Baum-Welch o algoritmo EM**
 (“Expectation-Maximization”)

Aplicación de los POMDPs a la robótica móvil

