Técnicas de los Sistemas Inteligentes

Práctica2: Planificación de Caminos con A* en Robótica.
Sesion6. Planificación **de caminos y Navegación Global**

Agenda

- Crear mapas y mundos con Stage.
- Costmaps
- Discretización para mapas
- Práctica2: Cómo construir nuestro planificador global y ejecutar planes gobales de navegación.

Map_server

- La funcionalidad principal de map_server es la de publicar un mapa como una matriz de ocupación, a partir de un fichero con una imagen.
- Descargar y descomprimir mistage1516
- Abrimos terminal en mistage1516/launch
- Comprobar contenido launch_mapserver.launch
  - Se lanza el servidor de mapas con un nuevo mapa.
  - Se lanza la herramienta rviz, para visualización de entornos con robot.
- Lanzamos launch_mapserver.launch
- Observar que se muestra el nuevo mapa en rviz
  - Es así porque rviz se subscribe al topic /map, que es el topic donde map_server publica el mapa
- Pero no hay robot …

Map_server y Stage

- Ahora vamos a lanzar stage de manera que en el simulador:
  - Su mundo simulado se haga a partir de la imagen actualmente tratada por map_server y mostrada por rviz.
- Lanzar el fichero launch_mapserver_stage.launch
  - Ahora sí hay robot …. Pero no se ve aun en rviz.
- Observar el fichero launch_mapserver_stage.launch
- Contiene como argumento el fichero mi-simplerooms.world que está en /mistage156/configuracion/mundos
- Es el fichero de configuración del mundo de Stage.
- Ver el fichero .world. En la documentación hay un tutorial sobre cómo hacer ficheros world en Stage.

Map_server y Stage

- Basta con configurar el objeto “floorplan”
  - Name nombre del mundo
  - Bitmap: localización del fichero
  - Size: tamaño [x y z] en metros
  - Pose del mapa respecto a la ventana (objeto “window”)
- El objeto robot pr2(pose [x y z theta] name “string” color “blue”)
- El objeto window
  - Size: tamaño de la ventana de Stage
  - scale: ¿cuántos metros corresponden a cada pixel?
    - Valor óptimo=round(W/M)
    - W:tamaño ventana
    - M:tamaño mapa
  - Descripción detallada en:
- http://playerstage.sourceforge.net/doc/Stage-3.2.1/group__worldgui.html

Map_server y Stage

```
define block model
(
  #definimos un block como un cubo de 50x50x75 cm cubicos.
  size [0.500 0.500 0.750]
  gui_nose 0
)

define topurg ranger
(
  #caracteristicas              del              modelo              sensor
      http://rtv.github.io/Stage/group__model__ranger.html
  sensor(
    range_max 30.0
    fov 270.25
    samples 1081
  )
  # generic model properties
  color "black"
  size [ 0.050 0.050 0.100 ]
)

define pr2 position
(
  size [0.650 0.650 0.250]
  origin [-0.050 0.000 0.000 0.000]
  gui_nose 1
  drive "omni"
  topurg(pose [ 0.275 0.000 0.000 0.000 ])
)

define floorplan model
(
  # sombre, sensible, artistic
  color "gray30"

  # most maps will need a bounding box
  boundary 1

  gui_nose 0
  gui_grid 0

  gui_outline 0
  gripper_return 0
  fiducial_return 0
  ranger_return 1.000
)
```

```
# set the resolution of the underlying raytrace model
in meters
resolution 0.02

interval_sim 100  # simulation timestep in milliseconds


window
(
  size [ 608 300 ]

  rotate [ 0.000 0.000 ]
  scale 12.100
)

# load an environment bitmap
floorplan
(
  name "simple_rooms"
  bitmap "../maps/simple_rooms.png"
  size [40.000 30.000 1.000]
  pose [ 0.000 0.000 0.000 0.000 ]
)

# throw in a robot
pr2( pose [ 8.557 6.313 0.000 -127.719 ] name "pr2"
color "blue")
block( pose [ 16.649 5.740 0.000 0.000 ] color "red")
block( pose [ -25.062 12.909 0.000 180.000 ] color
"red")
block( pose [ -25.062 12.909 0.000 180.000 ] color
"red")
```

Experimentación con mapas

- Tener en cuenta esta información para generar distintos mapas con los que experimentar.

costmaps

- Vamos a conocer y visualizar un nuevo tipo de objeto fundamental para tareas de la entrega 2 y entrega 3.
- En PRADO tenéis una guía, GuiaCostmaps.pdf, para entender el uso costmaps.

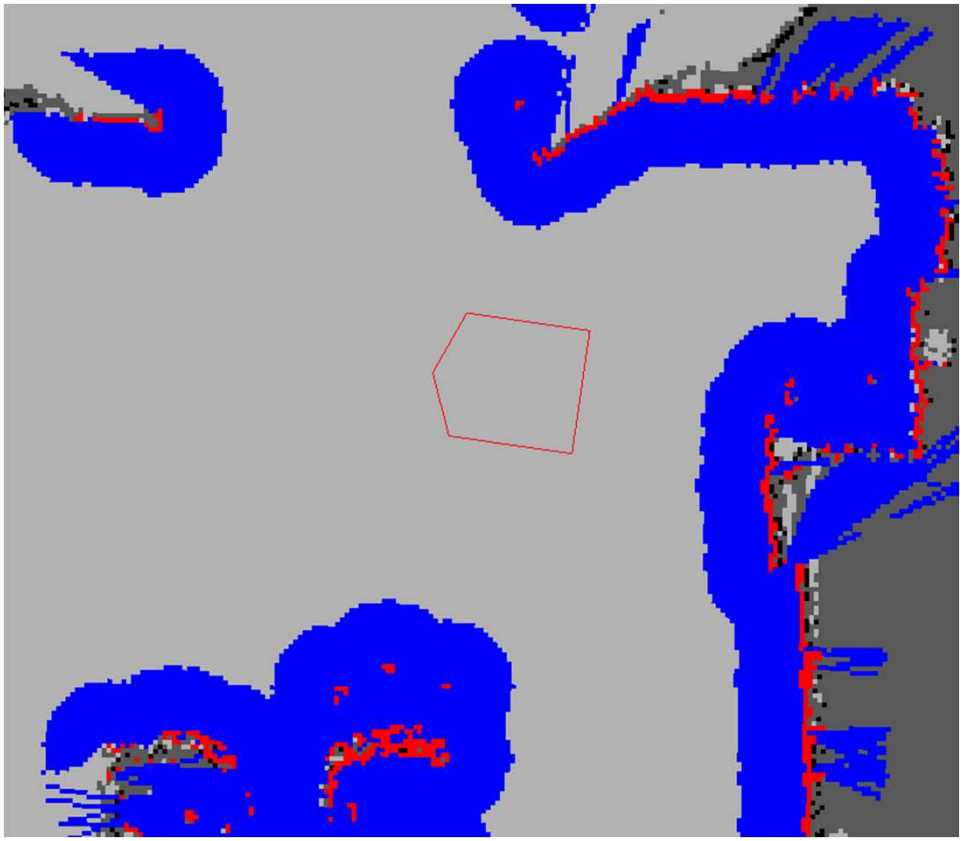
Costmap

- Un ***costmap*** es una estructura de datos que representa lugares seguros en los que el robot puede estar, en una cuadrícula de celdas.
- Los valores en el ***costmap*** representan o espacio libre o lugares donde puede colisionar, dentro de un rango amplio de valores [0,255].
- No confundir con el mapa creado por ***gmapping:*** *en este sólo se representa ocupado/libre/desconocido.*
- Usados en **técnicas de navegación con mapa.**
- En ROS la gestión de cotsmaps se hace con **nodos de tipo costmap,** se encargan de actualizar automáticamente la información del costmap conforme el robot se mueve.

- El robot se mueve usando navegación global o local.
- **Navegación global**: crear rutas para un *goal* en el mapa o una distancia más lejana:
  - Uso de ***global_costmap***
- **Navegación local:** crear rutas en distancias cercanas y *evitar obstáculos*
  - Uso de ***local_costmap.***

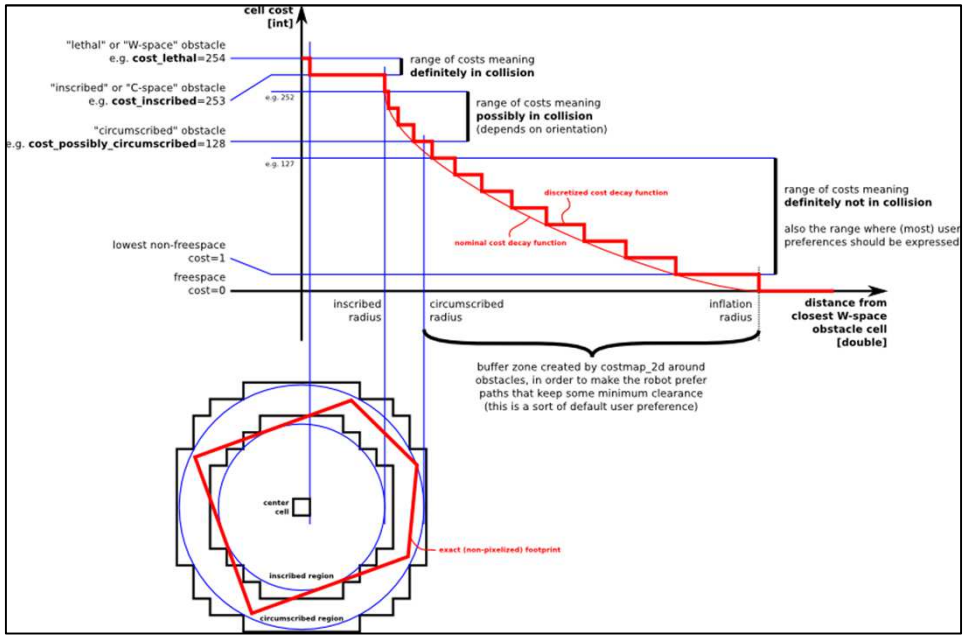
costmap_2d Package

- http://wiki.ros.org/costmap_2d
- **costmap_2d** package:
  - usa datos de sensores e información del mapa estático (construido con gmapping, p.ej.)
  - construye una malla de ocupación en 2D con costos basados en la información de ocupación e información del usuario.



Costmap Values

- Each cell in the costmap has a value in the range [0, 255] (integers).
- There are some special values frequently used in this range. (defined in include/costmap_2d/cost_values.h)
  - costmap_2d::**NO_INFORMATION** (255) - Reserved for cells where not enough information is sufficiently known.
  - costmap_2d::**LETHAL_OBSTACLE** (254) - Indicates a collision causing obstacle was sensed in this cell.
  - costmap_2d::**INSCRIBED_INFLATED_OBSTACLE** (253) - Indicates no obstacle, but moving the center of the robot to this location will result in a collision.
  - costmap_2d::**FREE_SPACE** (0) - Cells where there are no obstacles and the moving the center of the robot to this position will not result in a collision.



- API de costmap: http://docs.ros.org/indigo/api/costmap_2d/html/

Visualización de costmaps

- Descargar y descomprimir el paquete my_astar_planner de Prado en <workspace>/src/
- Compilar con catkin_make (en <workspace>).
- Hacer source devel/setup.sh
- Comprobar con rospack find my_astar_planner
- Ejecutar
  - roslaunch my_astar_planner move_base_amcl_10cm.launch

move_base_amcl_10cm.launch

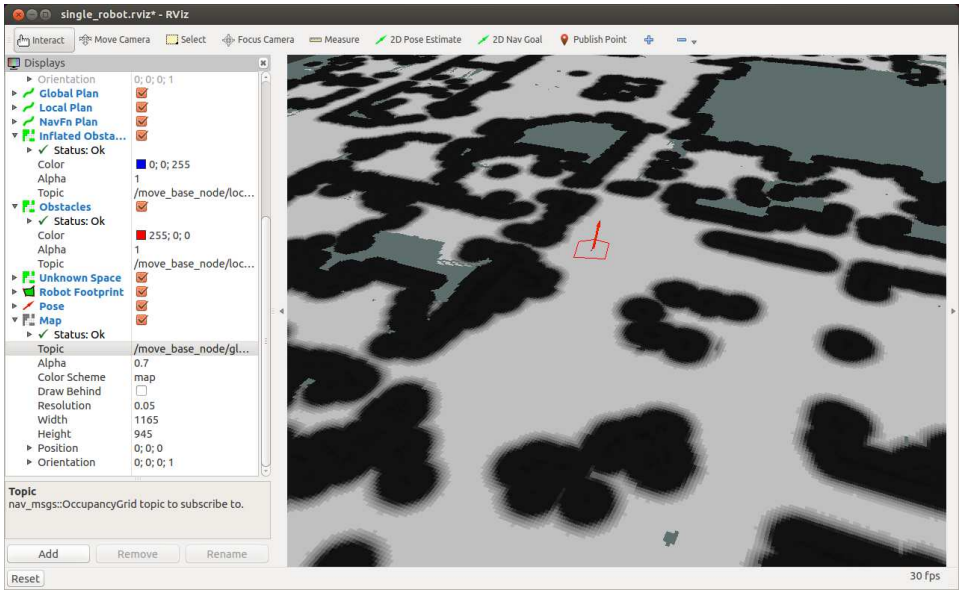
```
<launch>
  <master auto="start"/>
  <param name="/use_sim_time" value="true"/>
<!-- Lanzamos move_base para navegacion, con el planificador global por defecto -->

  <include file="$(find my_astar_planner)/move_base_config/move_base.xml"/>

<!-- Lanzamos map_server con un mapa, a una resolución de 10cm/celda -->

  <node name="map_server" pkg="map_server" type="map_server" args="$(find my_astar_planner)/stage_config/maps/willow-full.pgm 0.1"
respawn="false" />

<!-- Lanzamos stage con el mundo correspondiente al mapa -->

  <node pkg="stage_ros" type="stageros" name="stageros" args="$(find my_astar_planner)/stage_config/worlds/willow-pr2.world"
respawn="false">
    <param name="base_watchdog_timeout" value="0.2"/>
  </node>

<!-- Lanzamos el nodo amcl -->

  <include file="$(find my_astar_planner)/move_base_config/amcl_node.xml">
     <arg name="initial_pose_x"  value="47.120"/>
     <arg name="initial_pose_y"  value="21.670"/>
     <arg name="initial_pose_a"  value="0.0"/>

  </include>

<!-- Lanzamos rviz -->

 <node name="rviz" pkg="rviz" type="rviz" args="-d $(find my_astar_planner)/rviz_config/single_robot_OpenClosed.rviz" />
 </launch>
```

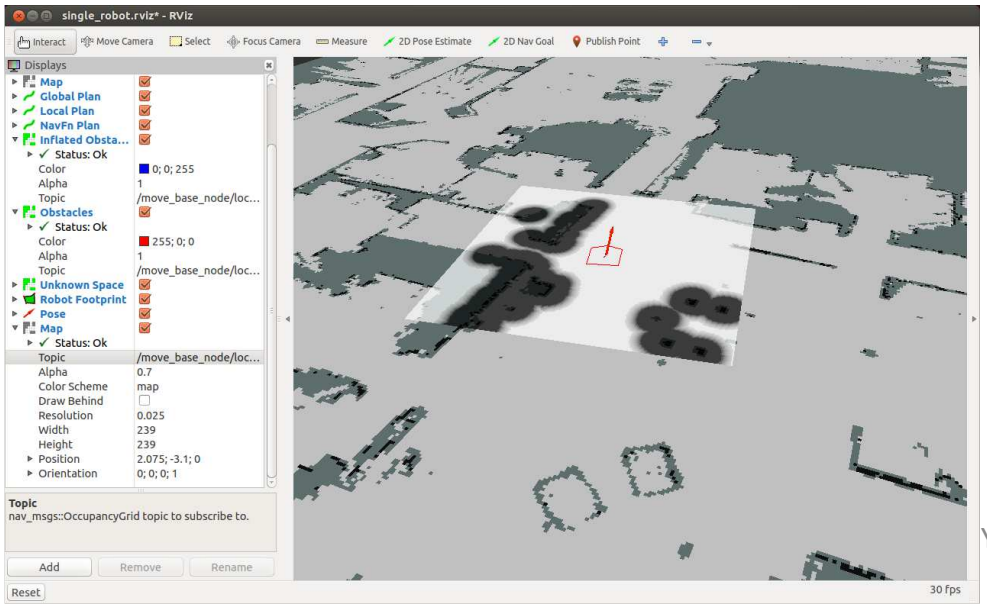
Visualización de costmaps

- Añadir un ***Map Display (si no está ya definido).***
- Local costmap topic:

  /miscostmaps_node/local_costmap/costmap
- Global costmap topic:

  /miscostmaps_node/global_costmap/costmap





- Configurar el footprint del robot (si no está configurado).
- Poner como topic:
- …/local_costmap/footprint
- Ejecutar

  rosrun rqt_reconfigure rqt_reconfigure

  para reconfigurar costmaps y planificador local si se desea

Configuración de costmaps

- Observar el fichero move_base_config/move_base.xml que está incluido en el launch anterior.
- Contiene la configuración del costmap global y local.

```
<launch>
<!--
  Example move_base configuration. Descriptions of parameters, as well as a full list of all amcl parameters, can be found
at http://www.ros.org/wiki/move_base.
-->
  <node pkg="move_base" type="move_base" respawn="false" name="move_base_node" output="screen">
    <param name="footprint_padding" value="0.01" />
    <param name="controller_frequency" value="10.0" />
    <param name="controller_patience" value="3.0" />

    <param name="oscillation_timeout" value="30.0" />
    <param name="oscillation_distance" value="0.5" />

    <param name ="GlobalPlanner/visualize_potential" value = "true" />
    <param name ="GlobalPlanner/use_dijkstra" value = "false" />

    <!--
    <param name="base_local_planner" value="dwa_local_planner/DWAPlannerROS" />
    -->

    <rosparam file="$(find my_astar_planner)/move_base_config/costmap_common_params.yaml" command="load"
ns="global_costmap" />
    <rosparam file="$(find my_astar_planner)/move_base_config/costmap_common_params.yaml" command="load" ns="local_costmap"
/>
    <rosparam file="$(find my_astar_planner)/move_base_config/local_costmap_params.yaml" command="load" />
    <rosparam file="$(find my_astar_planner)/move_base_config/global_costmap_params.yaml" command="load" />
    <rosparam file="$(find my_astar_planner)/move_base_config/base_local_planner_params.yaml" command="load" />
    <!--
    <rosparam file="$(find my_astar_planner)/move_base_config/dwa_local_planner_params.yaml" command="load" />
    -->
  </node>
```

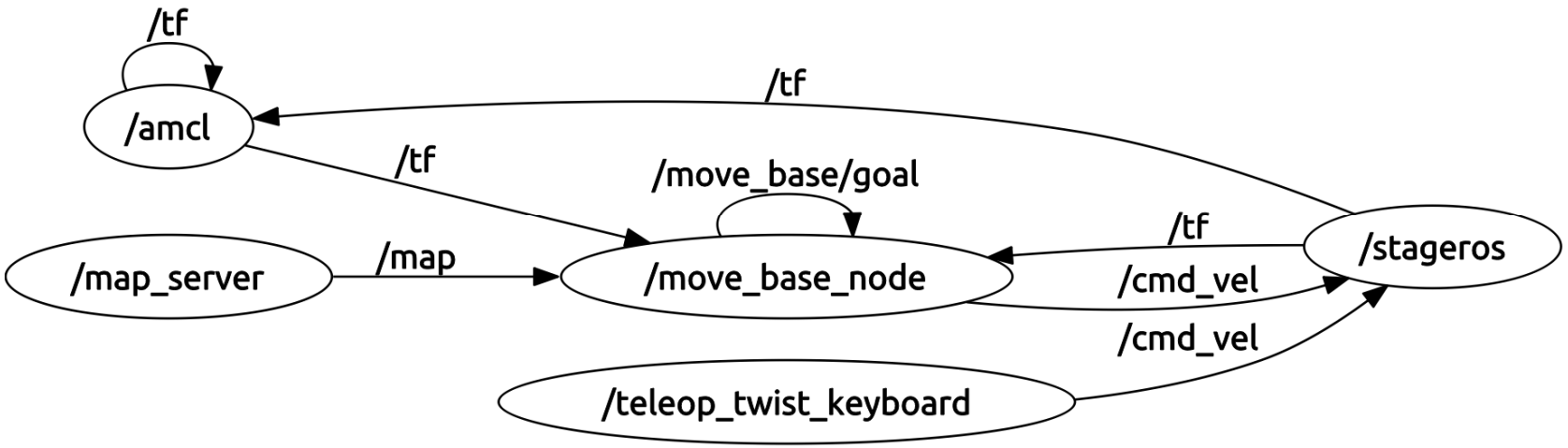
Configuración de costmaps

- **Ver documento GuiaCostmaps.pdf en PRADO para más información.**
- **Costmap_common_params.yaml**: se configuran parámetros comunes a ambos costmaps como, por ejemplo,
  - qué **topics** relativos a fuentes sensoriales (observation_sources) se van a usar (especificando el topic y tipo de mensaje entre otras cosas),
  - qué **valor de coste del costmap usamos como umbral** para, a partir de él, considerar que la celda que contiene ese valor es un obstáculo,
  - qué **radio consideramos para hacer la inflación** (en este ejemplo 55 cm, es decir, los valores de las casillas a menos de 55 cm de un obstáculo tendrán un valor distinto al de FREE_SPACE),
  - qué **dimensiones tiene el footprint del robot** (especificado en este caso como los puntos de un pentágono "con pico"), etc.

Configuración de costmaps

- **Local_costmap_params.yaml**: parámetros exclusivos del costmap local:
  - a qué frecuencia se modifica el costmap (**update frequency**). Este valor es **importante** porque hay que asignarlo considerando la frecuencia a la que se actualiza el escaneo láser y la frecuencia a la que se envían órdenes al robot.
  - **Frecuencia de publicación**: este valor es **fundamental** porque por defecto está a 0 (significa no publicar el costmap) puede dar dolores de cabeza si no se pone a un valor adecuado, por ejemplo 2 o 5 Hz.
  - Configuración del **rolling window** (ventana que avanza junto al robot):
    - rolling_window:**tiene que estar a true**,
    - las **dimensiones de la ventana** (width y height, en metros),
    - el **origen de coordenadas** (dejarlo a origin_x = origin_y = 0)
    - **la resolución** (resolution en metros/celda) indica "**cómo de grandes son las celdas del costmap"**, a mayor resolución, las celdas serán más pequeñas, pero esto afectará al tamaño del costmap y por tanto a la eficiencia de cualquier algoritmo que lo recorra.
- **Global_costmap_params.yaml**: son los mismos parámetros que los del local costmap pero con valores distintos, especialmente
  - **rolling_window tiene que ser false** (el global costmap no se desplza con el robot), por tanto es estático (**static_map = true**) y
  - es importante especificar en **qué topic se publica el mapa** del que se nutre el costmap local que en nuestro caso es "/map".

Nodes Graph



Las 4 cuestiones de la navegación

- ¿A dónde voy?
  - Objetivo: determinado por un humano o un planificador de misiones
- ¿Cuál es el mejor camino?
  - Problema de planificación de caminos (path planning), el que recibe más atención
    - Métodos cualitativos
    - Métodos cuantitativos
- ¿Por dónde he pasado?
  - Creación de mapas.
- ¿Dónde estoy?
  - Localización y creación de mapas.

Técnicas de Navegación

- En particular un robot necesita planificar un camino que le especifique cómo alcanzar una localización particular, mediante la consecución de varios hitos intermedios.
- Hay distintas técnicas para Navegación Global:
  - Navegación topológica o cualitativa
  - Navegación métrica o cuantitativa
  - En cualquier caso, para la navegación global es necesario tener un mapa del entorno
    - Siempre aparece la misma pregunta: ¿Cómo construyo mi mapa, o de dónde puedo obtenerlo?

Navegación topológica o cualitativa

- Basada en encontrar relaciones de conectividad entre objetos del mundo
- Representación del conocimiento
  - (habitación-tiene habitacion1 puerta1) (conecta puerta1 pasillo1) ….
- Paso previo a la planificación automática de tareas.
  - **Práctica 3**
- Distinguir:
  - Planificación de tareas (STRIPS)
    - Muchos tipos de acciones: mover, coger, soltar, apilar,….
    - Representación sofisticada de distintos efectos y precondiciones
  - Planificación de caminos
    - Las acciones planificadas son sólo movimientos del tipo:
      - (move-to x, y, theta)
    - Efectos de acciones: el robot cambia de posición

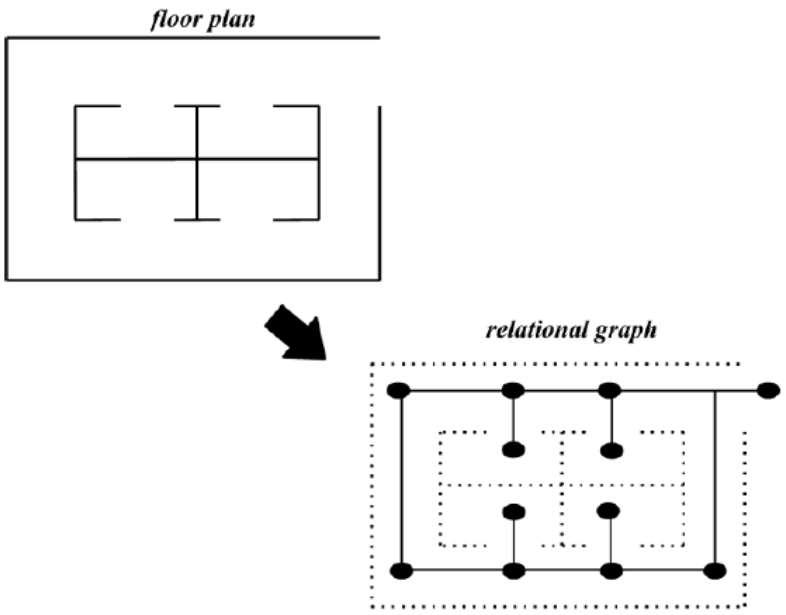
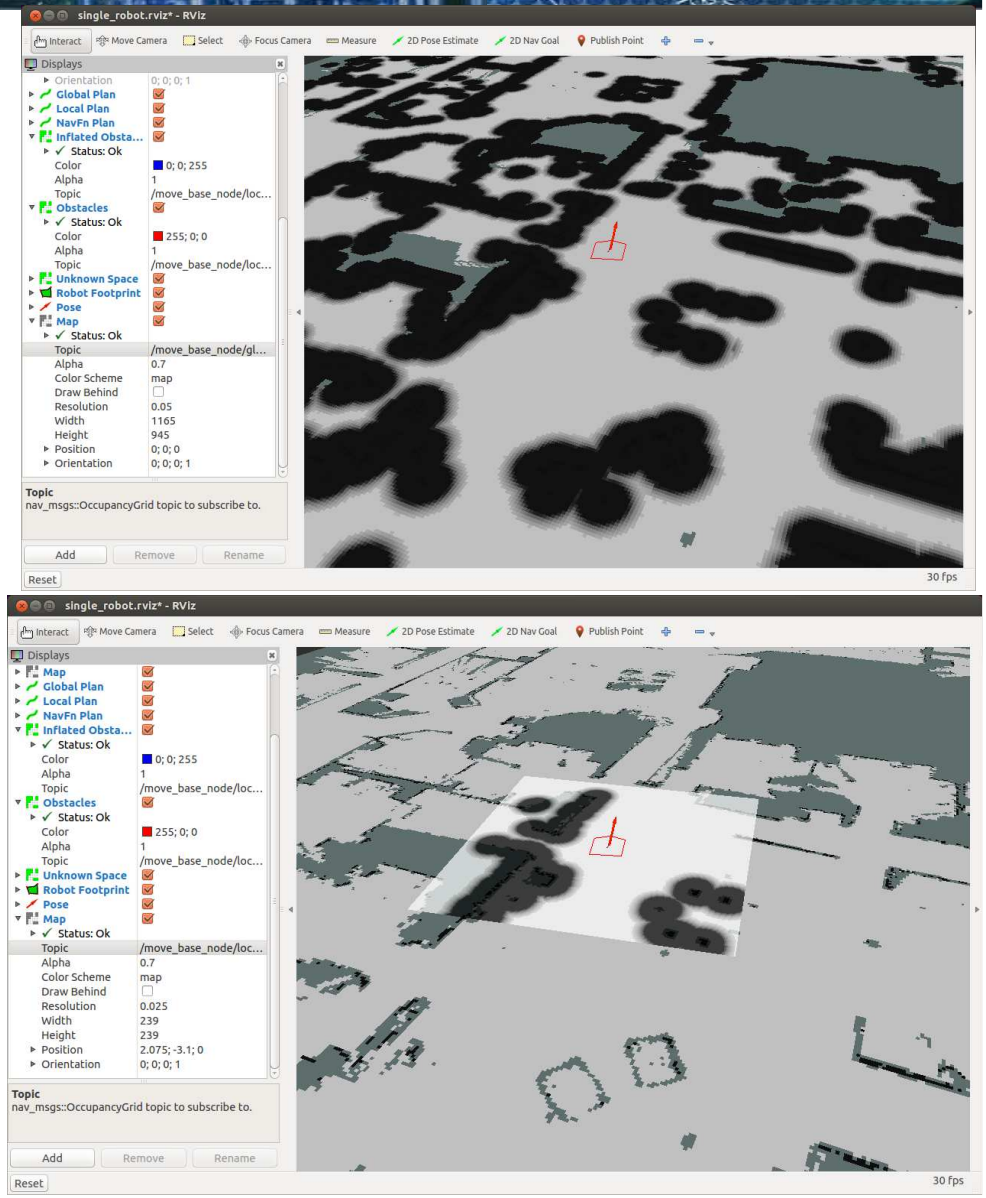


Figure 9.2 Representation of a floor plan as a relational graph.

Navegación métrica o cuantitativa

- A partir de una representación del mundo como un "mapa“ discretizado de posibles posiciones (espacio de configuraciones) del robot
- Tratar de encontrar un camino, si es posible óptimo.
- Ejecutar el plan con el navegador local.



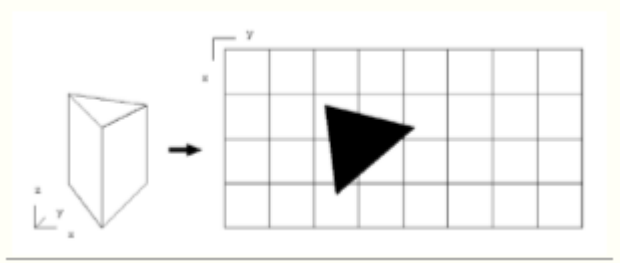
¿Cómo construyo mi mapa, o de dónde puedo obtenerlo?

- No conozco el mapa: necesito técnicas de **localización**
- Sí lo conozco: necesito técnicas de "discretización", porque trabajamos con dominios contínuos.

DISCRETIZACIÓN

Representación: Espacio de configuraciones (CSPace)

- Espacio del mundo
  - robots, objetos, obstáculos...
- Cspace (configuration space)
  - Una estructura de datos que permite especificar la posición (pose) de cualquier objeto y el robot (por ejemplo: array bidimensional).
  - Una buena representación del Cspace reduce el número de grados de libertad con los que se tiene que enfrentar un planificador
    - Por ejemplo, aunque el robot pueda trabajar con 6 grados de libertad, un robot móvil terrestre sólo necesita 3 (x,y,yaw)



Representación: Espacio de configuraciones (CSPace) (2)

- Representaciones de un Cspace
  - Varios tipos de representaciones, todas orientadas a crear particionamiento (discretización) del espacio
    - Espacio libre (configuraciones en las que puede estar el robot)
    - Espacio ocupado (configuraciones inalcanzables - obstáculos, muros)
  - Métodos de descomposición en celdas
  - Métodos de esqueletización

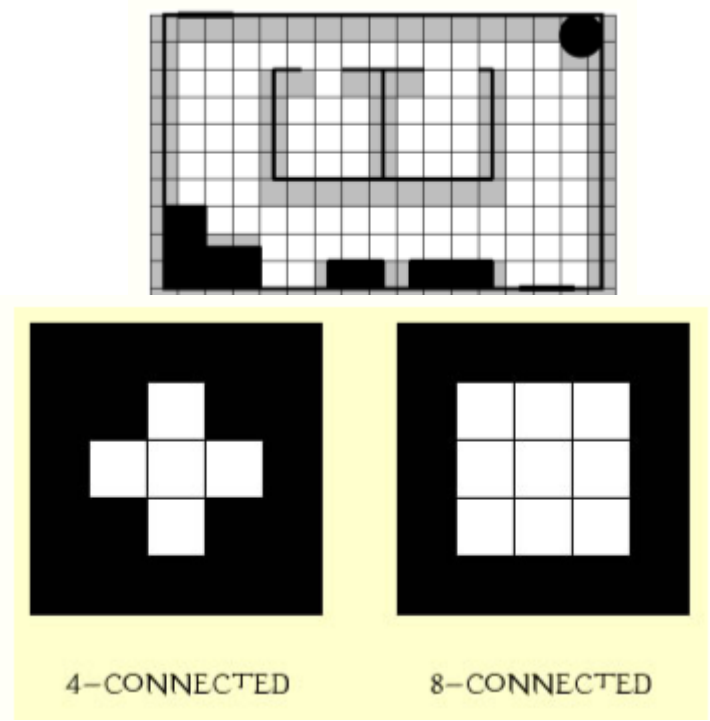
Descomposición en celdas

- Discretizar el espacio libre en un número finito de regiones contiguas, llamadas celdas.
- El problema de búsqueda de un camino se convierte fácilmente en un problema de búsqueda en grafos (de los que ya conocemos)
- En general recibe el nombre de Matriz de Ocupación (Occupancy Grid)

Cuadrículas regulares

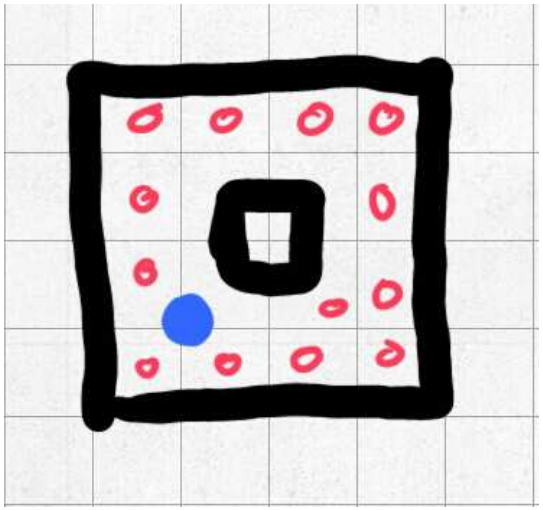
- Superponer una cuadrícula cartesiana sobre el espacio del mundo (lo que hace un costmap)

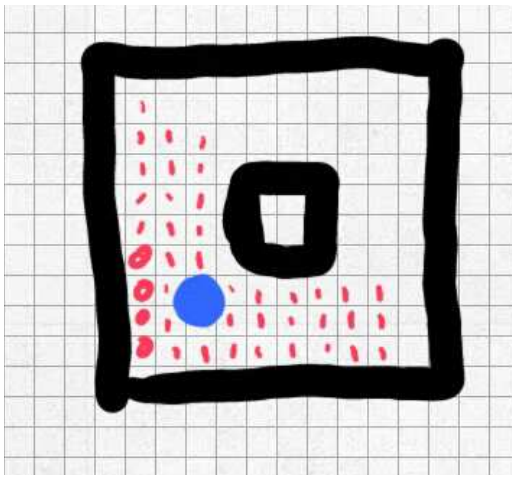
- Referidas como "matrices o mallas de ocupación" (occupancy grid)
  - Casilla = ocupada (0), si hay objeto en el área contenida por la casilla
  - Casilla = libre (1), si no hay objeto.
  - Casilla = desconocida (-1), si se desconoce
    - En problemas de localización es útil.
- Su aplicación en búsqueda de grafos es inmediata.
  - El centro de cada celda es un nodo en el grafo.
  - 4-conexas u 8-conexas (se permite diagonal)



Cuadrículas regulares (2)

- Inconvenientes
  - Sesgo en la digitalización
    - Si un objeto cae en un área muy pequeña de la celda, se marca como ocupada
    - Una solución:
      - cuadrículas muy finas (6 a 10 cm/cuadrado)
    - Costo muy alto de almacenamiento y un número muy alto de nodos en procesos de búsqueda.

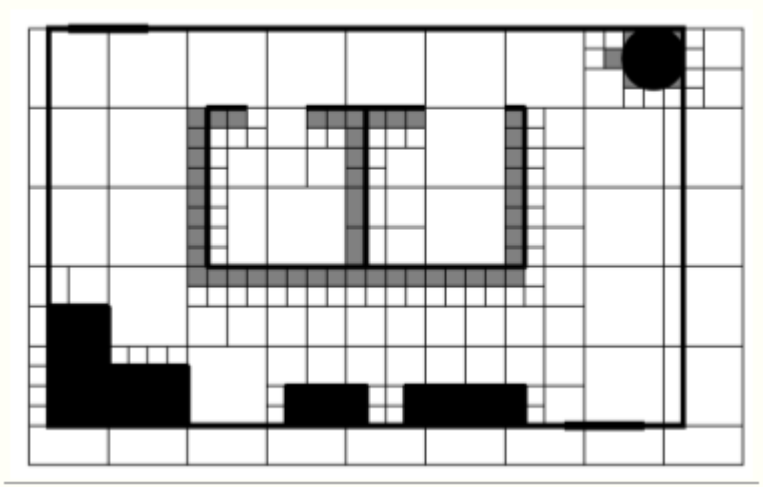




Cuadrículas regulares recursivas (Quadtrees)

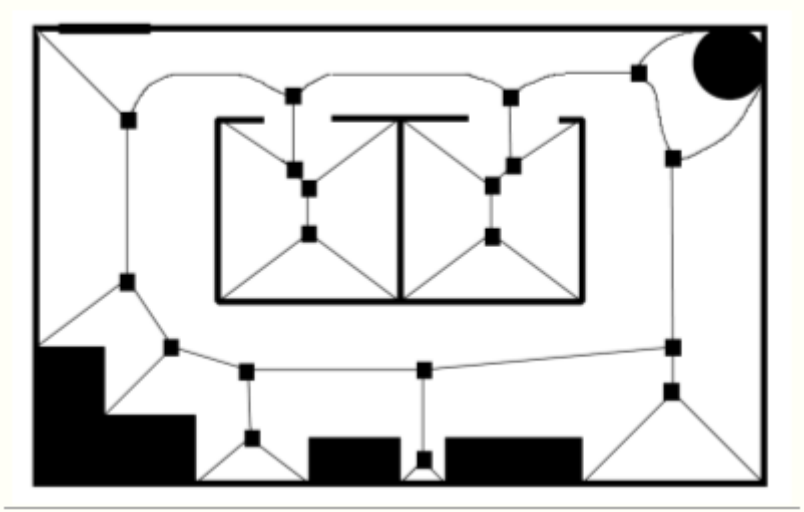
- Variante de cuadrícula regular

- Evitar espacio desperdiciado
  - Tamaño celda inicial (p.ej: 16x16cm)
  - Si un objeto ocupa parte, pero no toda la celda
  - Entonces el algoritmo de Cspace divide esa celda en 4 (quad) celdas.
  - Hacerlo recursivamente.
- Ver la opción "View> Data> Occupancy grid" de Stage



Esqueletización

- Grafos de Voronoi

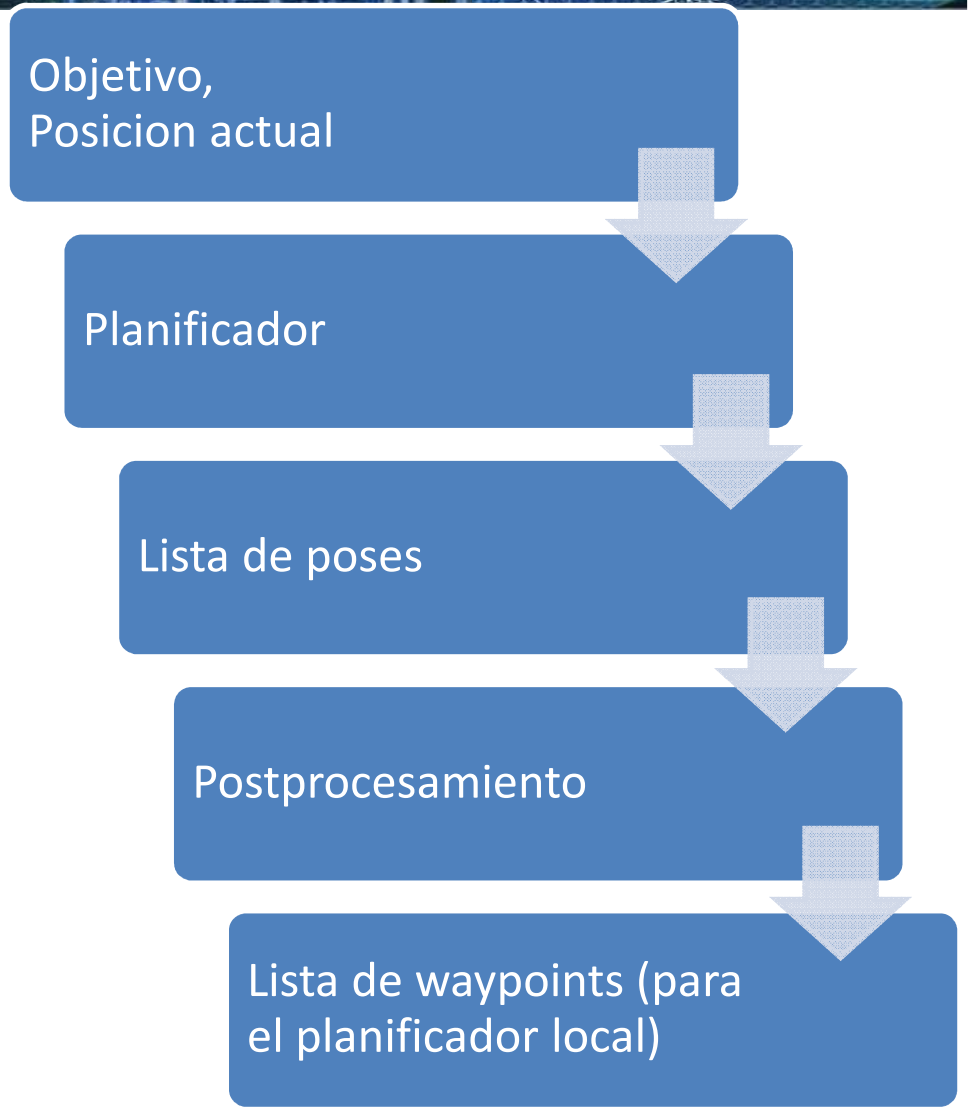


  - Arco de Voronoi: línea equidistante a todos los puntos de obstáculos.
    - Esta idea genera líneas que van por el centro del los pasillos
  - Vértice (nodo) de Voronoi: punto donde se encuentran varios arcos de Voronoi
  - Un grafo de Voronoi, construido a partir de un mapa, involucra una estrategia de control implícita
    - Permite a un robot navegar de un vértice a otro permaneciendo equidistante a todos los obstáculos.
    - Si sigue un arco de Voronoi, no colisionará.
  - Requieren más tiempo para construirse, pero la planificación es más eficiente que en descomposición de celdas.

PLANIFICACIÓN GLOBAL (PLANIFICACIÓN DE CAMINOS MÉTRICA)

Planificación de caminos

- Objetivo de la planificación de caminos:
  - Determinar un camino hacia un objetivo especificado.
  - Camino: secuencia de "poses"
  - Objetivo: una pose.
- Principales características
  - Tratan de encontrar un camino óptimo.
  - Waypoint (hito)
    - La secuencia de configuraciones se "postprocesa" y se obtiene un conjunto de subobjetivos
    - Cada punto guía (subobjetivo) es una pose (x,y, theta).
    - Ubicaciones donde el robot podría cambiar su orientación



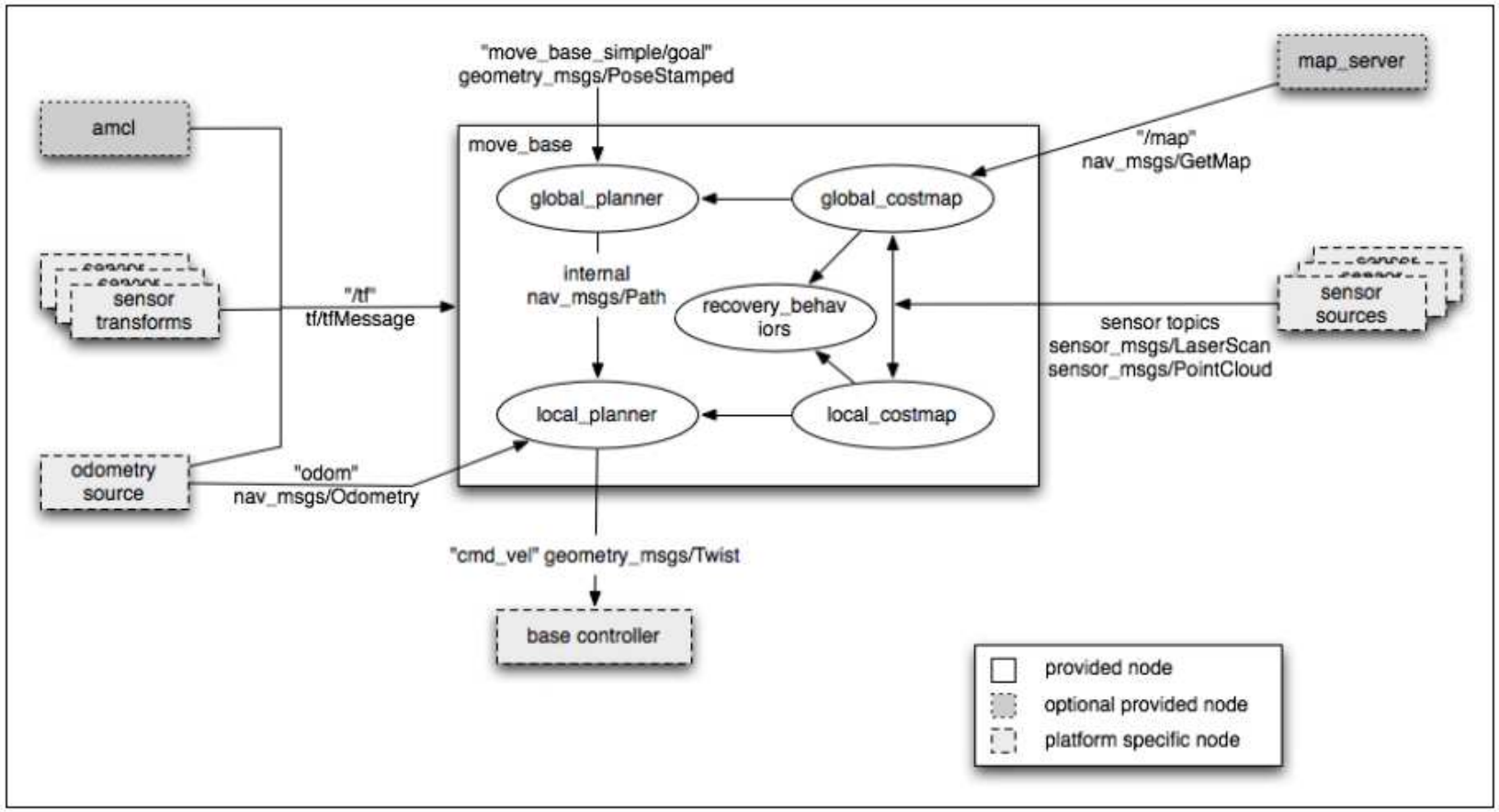
Planificación de caminos (2)

- Componentes de un planificador de caminos global
  - La representación
    - Normalmente cuadrículas regulares, en ROS se utiliza el concepto de **costmap** para representar el mundo discretizado.
    - http://wiki.ros.org/costmap_2d
  - El algoritmo
    - Problema de búsqueda en grafos (por ejemplo A*)
    - Tratamiento directo de la cuadrícula, similar a un problema de "coloreado de grafos".
  - Un problema que siempre aparece
    - ¿Cuándo uso el planificador y cuando me muevo?
    - Entrelazado de planificación y ejecución, este problema está resuelto en ROS con el paquete move_base http://wiki.ros.org/move_base dentro de navigation stack.

ROS Navigation Stack

- http://wiki.ros.org/navigation
  - Un stack de paquetes ROS que
    - a partir de información sobre odometría, sensores y una pose objetivo,
    - devuelve comandos de velocidad enviados a una base móvil de robot
  - Diseñada para mover cualquier robot móvil sin que se quede perdido ni choque.
  - Incorpora soluciones para la Navegación Global y Navegación Local con mapa.
  - **Instalar Navigation Stack:**
    - **sudo apt-get install ros-indigo-navigation**
- ROS Navigation Introductory Video

Navigation Stack

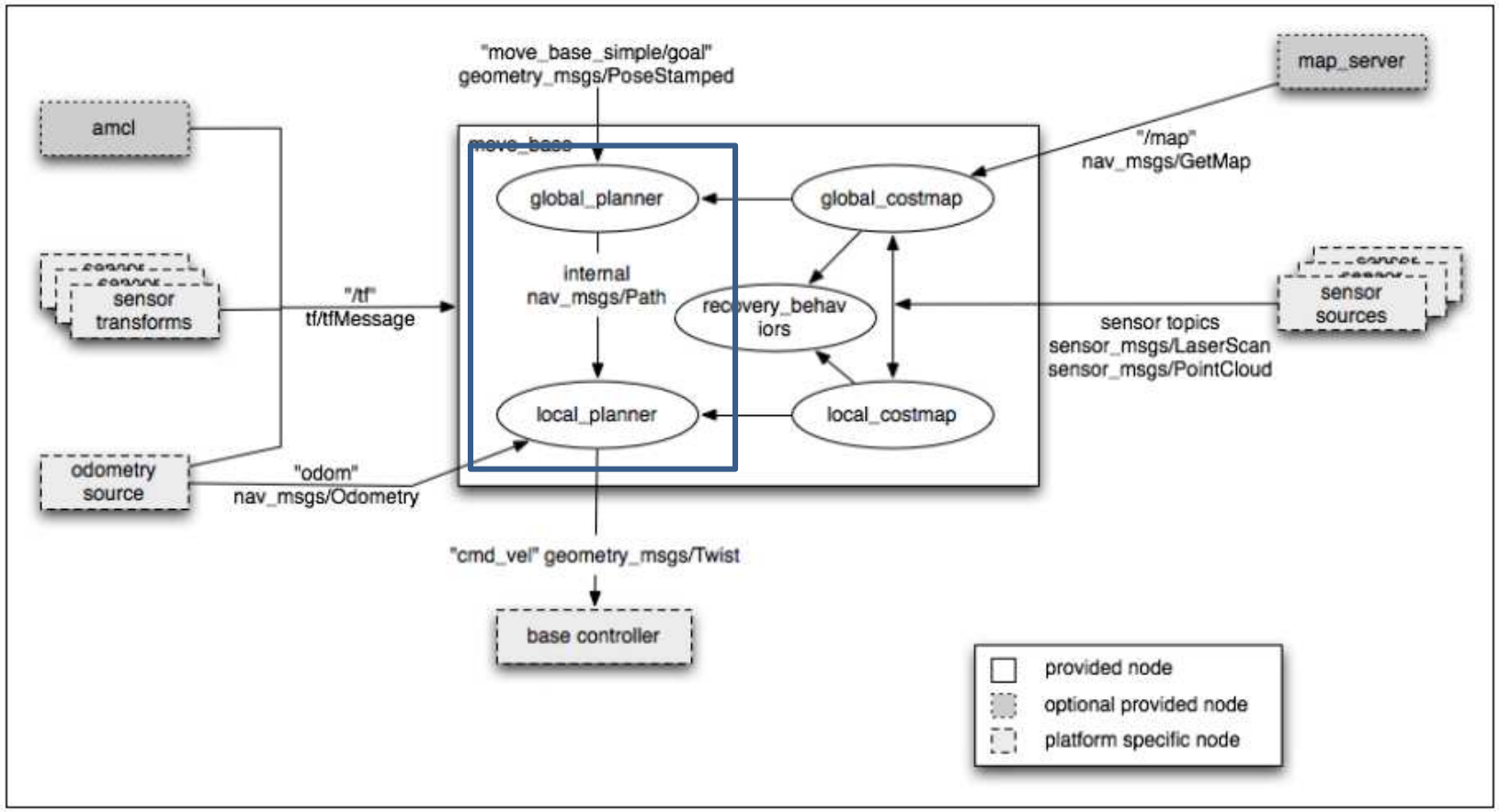


PAQUETE MOVE_BASE

move_base

- Este paquete permite mover el robot a una posición deseada usando ***navigation_stack***
- El *nodo move_base* aúna un *global_planner* y un *local_planner* para desempeñar la tarea de ***navegación global.***
- El *nodo move_base* puede realizar opcionalmente *recovery behaviours* cuando el robot percibe que está atascado.
- Basta con ejecutar un fichero *launch* porque el paquete ***move_base*** viene con la instalación de ROS.

move_base package



CONFIGURAR MOVE_BASE CON UN PLANIFICADOR GLOBAL DE COSECHA PROPIA

Idea de la Práctica 2

- Demostración de una implementación de A* propia.
  - Visualización de Abiertos y Cerrados en rviz.
  - En my_astar_planner de PRADO hay implementada una búsqueda en anchura.
  - Ejecutar
  - roslaunch my_astar_planner move_base_amcl_10cm+myAstar.launch
- Se pide implementar un A* mejorando la implementación de anchura.
  - Se pide también mejorar el algoritmo A* para tratar de reducir el tiempo en que tarda en encontrar una solución.
  - Este es un requisito importante porque la implementación está integrada en una arquitectura para la navegación de un robot en tiempo real.
  - Para intentar reducir el tiempo de búsqueda puede usarse cualquier técnica explicada en teoría o alguna de las técnicas descritas en un blog muy referenciado sobre A* para juegos (que apunta técnicas totalmente aplicables a nuestro problema en robótica) .
    http://theory.stanford.edu/~amitp/GameProgramming/

move_base_amcl_10cm+myAstar.launch

```
<launch>
  <master auto="start"/>
  <param name="/use_sim_time" value="true"/>


  <!-- Lanzamos move_base para navegacion, con la configuración de un planificador global específico -->

  <include file="$(find my_astar_planner)/move_base_config/move_base+global_planner.xml"/>

 <!-- Lanzamos map_server con un mapa, a una resolución de 10cm/celda -->

  <node  name="map_server"  pkg="map_server"  type="map_server"  args="$(find  my_astar_planner)/stage_config/maps/willow-
full.pgm 0.1" respawn="false" />

<!-- Lanzamos stage con el mundo correspondiente al mapa -->

  <node  pkg="stage_ros"  type="stageros"  name="stageros"  args="$(find  my_astar_planner)/stage_config/worlds/willow-
pr2.world" respawn="false">
    <param name="base_watchdog_timeout" value="0.2"/>
  </node>

<!-- Lanzamos el nodo amcl -->

  <include file="$(find my_astar_planner)/move_base_config/amcl_node.xml">
     <arg name="initial_pose_x"  value="47.120"/>
     <arg name="initial_pose_y"  value="21.670"/>
     <arg name="initial_pose_a"  value="0.0"/>

  </include>

<!-- Lanzamos rviz -->

 <node name="rviz" pkg="rviz" type="rviz" args="-d $(find my_astar_planner)/rviz_config/single_robot_OpenClosed.rviz" />
</launch>
<launch>
```

¿Qué tengo que saber?

- Move_Base funciona con un planificador global por defecto pero puede cambiarse por otro planificador global implementado, por ejemplo A*.
- Para desarrollar un planificador global hay que saber que en ROS se pueden desarrollar plugins para adaptar funcionalidades a nuestras necesidades.
- ¿Cómo desarrollo un planificador global como plugin?
- ¿Cómo configurar move_base para que ejecute mi plugin?

- Un plugin se desarrolla como un paquete normal, pero en el directorio raíz del paquete tiene que haber un fichero xml que declare que lo que se está desarrollando es un plugin.
- El resultado de la compilación no es un nodo ejecutable, sino una librería, y se almacena en el directorio lib del workspace.
- Vamos a ver primero
  - como se gestiona la ejecución de plugins en move_base,
  - luego vemos como se implementa.

move_base+global_planner.xml

```
<launch>
<!--
  Example move_base configuration. Descriptions of parameters, as well as a full list of all amcl parameters, can be found
at http://www.ros.org/wiki/move_base.
  Ejemplo de configuración de move_base para un planificador global específico.
  Los parámetros que se han añadido al move_base.xml estándar están marcados
-->
  <node pkg="move_base" type="move_base" respawn="false" name="move_base_node" o
    <param name="footprint_padding" value="0.01" />
    <param name="controller_frequency" value="10.0" />
    <param name="controller_patience" value="10.0" />
    <param name="oscillation_timeout" value="30.0" />
    <param name="oscillation_distance" value="0.5" />
```

Configuración planificador local
http://wiki.ros.org/base_local_planner

```
   <param name="planner_patience" value="120.0" /> <!-- Define el timeout para en
    <param name="base_global_planner" value="my_astar_planner/MyAstarPlanner"/>
se va a usar -->
```

Configuración planificador global
http://wiki.ros.org/move_base

```
    <!--
    <param name="base_local_planner" value="dwa_local_planner/DWAPlannerROS" />
    -->

    <rosparam file="$(find my_astar_planner)/move_base_config/costmap_common_params.yaml" command="load"
ns="global_costmap" />
    <rosparam file="$(find my_astar_planner)/move_base_config/costmap_common_params.yaml" command="load" ns="local_costmap"
/>
    <rosparam file="$(find my_astar_planner)/move_base_config/local_costmap_pa
    <rosparam file="$(find my_astar_planner)/move_base_config/global_costmap_p
    <rosparam file="$(find my_astar_planner)/move_base_config/base_local_plann
    <!--
    <rosparam file="$(find my_astar_planner)/move_base_config/dwa_local_planne
    -->
  </node>
</launch>
```

Configuración costmaps

http://wiki.ros.org/costmap_2d

Declarar un plugin

- ¿Dónde declaro que **myastar_planner/MyastarPlanner** es un plugin para move_base?
  - En el fichero xml del plugin que he desarrollado previamente.
- Descargar la carpeta my_astar_planner.zip y descomprimirla en
  - <workspace>/src/
  - comprobar que hay un directorio my_astar_planner en src.
- Observar que es un directorio de paquete, pero que contiene un xml adicional
  - astar_planner_plugin.xml (el nombre da igual, lo importante es el contenido)
  - Observar que el valor del parámetro de move_base.xml que llama al plugin es el argumento “name” del tag “class”.
  - Más información en
  - http://wiki.ros.org/navigation/Tutorials/Writing%20A%20Global%20Path%20Planner%20As%20Plugin%20in%20ROS

```
<library path="lib/libmy_astar_planner">
  <class name="my_astar_planner/MyAstarPlanner" type="myastar_planner::MyastarPlanner"
         base_class_type="nav_core::BaseGlobalPlanner">
    <description>
      El planner que me invento yo.
    </description>
  </class>
</library>
```

Código fuente A*

- El directorio include/my_astar_planner contiene el fichero donde se define la clase del planificador global.
  - Es una clase que implementa el A*.
- El directorio src/ contiene el fichero .cpp donde se implementa el A*.
- Para compilar:

$cd <workspace>
$catkin_make (debería compilar sin error)
para comprobar que el plugin está exportado con éxito hacer
$source devel/setup.sh (comprobar después con rospack find my_astar_planner)
$**rospack plugins --attrib=plugin nav_core**
(debería salir una lista de ficheros xml y entre ellos)
/home/<tuhome>/<workspace>/src/my_astar_planner/astar_planner_plugin.xml

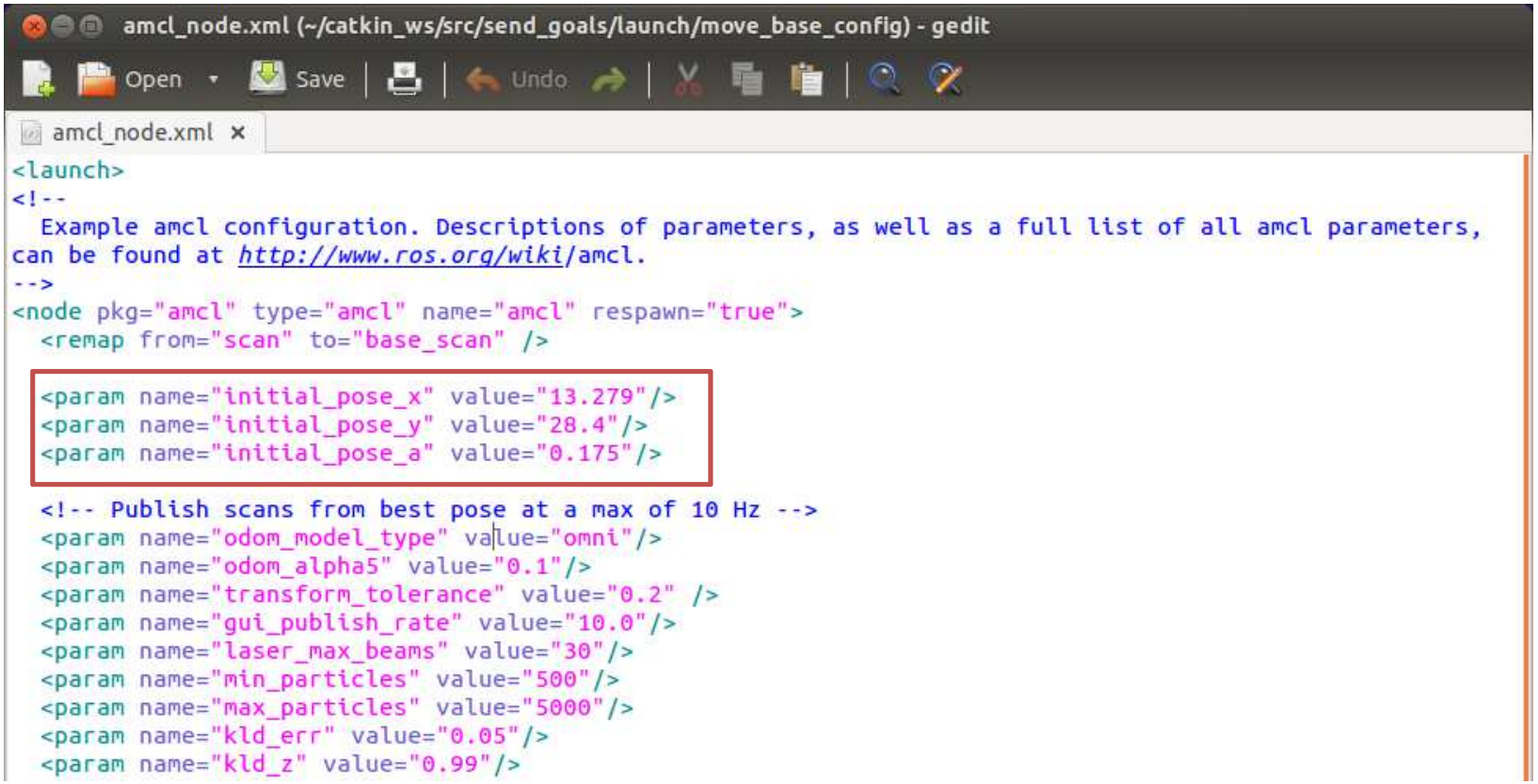
launchs

- En el directorio launch del paquete my_astar_planner hay varios ficheros para lanzar distintas configuraciones:
- move_base_**cm.launch
  - lanza el planificador global por defecto de ROS con mapas configurados a una resolución de 5 y 10 cm.
- move_base_**cm+myAstar.launch
  - lanza el planificador global construido por nosotros.


DECSAI
ejecutar el launch

Configurar amcl.xml para recibir una pose inicial por defecto

- Para evitar estar todo el rato indicándole la pose inicial, podemos reconfigurar amcl para que reciba una pose inicial por defecto.



move_base_amcl_10cm+myAstar.launch

- Para evitar estar todo el rato indicándole la pose inicial, podemos reconfigurar amcl para que reciba una pose inicial por defecto.

```
 <!-- Lanzamos map_server con un mapa, a una resolución de 10cm/celda -->

  <node  name="map_server"  pkg="map_server"  type="map_server"  args="$(find  my_astar_planner)/stage_config/maps/willow-
full.pgm 0.1" respawn="false" />

<!-- Lanzamos stage con el mundo correspondiente al mapa -->

  <node   pkg="stage_ros"   type="stageros"   name="stageros"   args="$(find   my_astar_planner)/stage_config/worlds/willow-
pr2.world" respawn="false">
     <param name="base_watchdog_timeout" value="0.2"/>
  </node>

<!--  Lanzamos el nodo amcl  -->

  <include file="$(find my_astar_planner)/move_base_config/amcl_node.xml">
     <arg name="initial_pose_x"  value="47.120"/>
     <arg name="initial_pose_y"  value="21.670"/>
     <arg name="initial_pose_a"  value="0.0"/>

  </include>

<!-- Lanzamos rviz -->

 <node name="rviz" pkg="rviz" type="rviz" args="-d $(find my_astar_planner)/rviz_config/single_robot_OpenClosed.rviz" />
</launch>
<launch>
```

Entrega 3

- Implementar A* a partir del código fuente entregado para ejecutarlo como plugin de *global_planner* en *move_base*.
  - El objetivo es obtener el camino **más corto y más seguro** desde la posición actual del robot hasta la posición dada como objetivo.
    - Camino más seguro: el que garantiza en todo paso que la distancia del robot a los obstáculos es la más amplia.
  - En el código fuente se proveen funciones que calculan heurística en función de la distancia pero no de la seguridad del camino.
- El resultado final tiene que ser no solo que se obtenga el mejor camino posible, además se tiene que conseguir que en global el robot alcance el punto lo más rápido posible.
- Entregar:
  - Código fuente del A*, implementado modifcando el fichero my_astarplanner.cpp.
  - Memoria como se especifica en el documento Práctica2.pdf de PRADO
- Fecha de entrega para todos los grupos: 28 de Abril a las 23:59.

MATERIAL ADICIONAL (VARIANTES DE ALGORITMOS PARA BÚSQUEDA EN COSTMAPS)

Planificación de caminos basada búsqueda en grafos

- Un CSpace puede convertirse en un grafo
  - Nodo: una configuración (una celda)
  - Arco: celdas adyacentes
  - Nodo inicial: configuración actual
  - Nodo objetivo: configuración objetivo
- Cualquier técnica de búsqueda que ya conocéis puede aplicarse

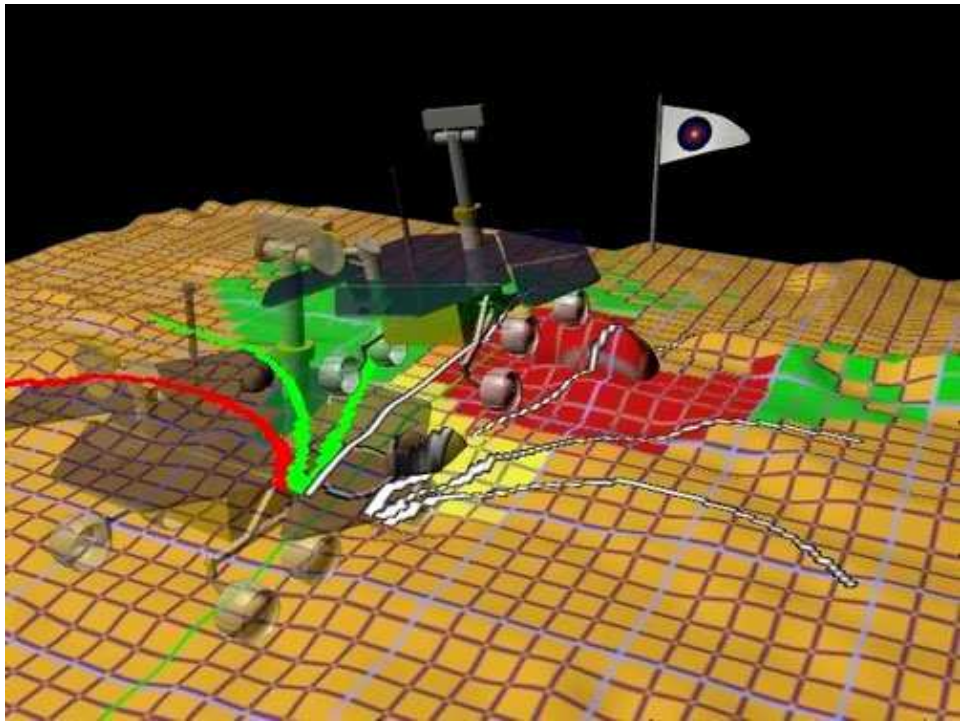
A* es muy usado

- Produce un camino óptimo:
  - Comenzando por el nodo inicial n_0
  - Explorando sus descendientes e incrementa el camino seleccionando siempre el mejor nodo
  - Selección basada en la función f(n) = g(n) + h(n)
    - g(n) = d(n_0,n-1) + d(n-1,n)
    - h(n) =d_linearecta(n, n_f)
  - Si h(n) es admisible y monótona entonces en una búsqueda de grafos A* encuentra el camino óptimo
  - La distancia en línea recta es una heurística admisible porque nunca sobreestima el coste real hasta el objetivo

A* para planificación de caminos

- Ventaja:
  - Puede usarse en cualquier CSpace que pueda convertirse a un grafo (cuadriculas regulares, Voronoi, conectividad,....)
- Inconveniente
  - En entornos donde hay más factores o información a tener en cuenta (p.ej: gasto de energía) hay que afinar mucho en la heurística para que siga siendo admisible.

D*



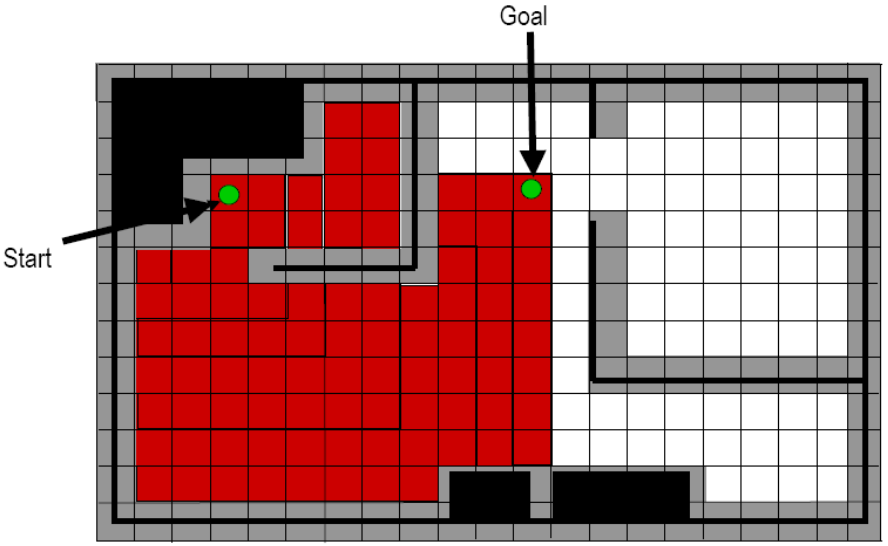
- Variante de A*
  - Planifica previamente un camino (A*) desde cualquier posición hasta el objetivo
  - A* : el camino más corto desde una sola fuente (estilo Dijkstra)
  - D*: el caminio más corto para todos los pares (situación, objetivo) (estilo Floyd-Warshall)
  - h* se basa en la idea de traversabilidad
    - la traversabilidad se actualiza dinámicamente con nueva información detectada
    - D* replanifica contínuamente, modificando su mapa con información en tiempo real

Basados en "frente de onda" (wavefront)

- A* requiere construir un grafo (estructuras de datos adicionales)
- La técnica Wavefront es muy adecuada para trabajar directamente en la cuadrícula (en la matriz)

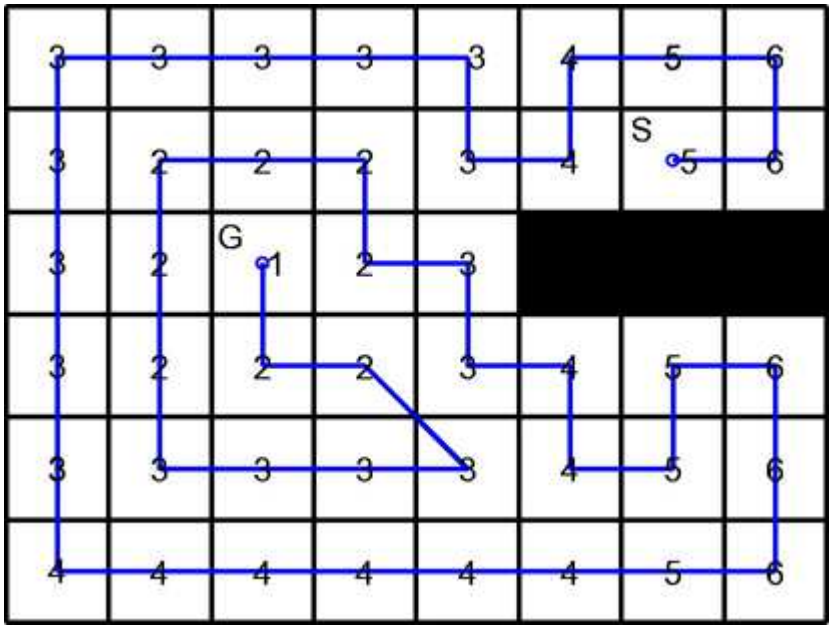
Wavefront

- Idea
  - considerar CSpace como un material conductivo con el calor radiando desde la posición actual hasta el objetivo.
  - Si hay camino, se supone que el calor llega al objetivo.
  - Se puede obtener el camino óptimo desde todas las celdas hasta el objetivo.
  - Al final tenemos un mapa que parece un campo de potencial



Wavefront (2)

- Algoritmo wavefront
  - 1. Propagar la onda desde el objetivo hasta el estado inicial
    - Cuadrícula binaria: 1 ocupado, 0 libre
    - Celda objetivo = 2
    - Celdas adyacentes a 2, que tengan 0, poner 3
    - Celdas adyacentes a 3, que tengan 0, poner 4
    - ....
    - Hasta llegar a la celda objetivo.
  - 2. Extraer el camino usando gradiente descendiente
    - Si la celda inicial tiene x, buscar una adyacente que tenga x-1 (grad. descendiente). Marcarla como nodo del camino (waypoint).
    - Después buscar las vecinas de esta, con x-2
    - ...
    - Hasta que llegue al nodo objetivo.
  - 3. Suavizar el camino (extraer waypoints)
    - Iterativamente, eliminar el waypoint i, si de i-1 a i+1 (en línea recta) no se atraviesa ningún obstáculo.
    - repetir hasta que no se puedan eliminar más waypoints
    - Devolver los waypoints.



Wavefront (3)

- Puede usarse en diferentes tipos de terrenos
  - Obstáculo: conductividad 0
  - Espacio libre: conductividad infinita
  - Terreno no deseable (areas rocosas): baja conductividad -> camino más costoso.
  - Se puede usar bidireccional (para ahorra coste computacional)

ENTRELAZANDO PLANIFICACIÓN (DELIBERATIVA) Y EJECUCIÓN (REACTIVA)

Aproximación inicial:

- Primero planifico, luego ejecuto.
  - Para cada waypoint (subobjetivo)
    - Lo tomo como vector direccional de entrada
    - Aplico un comportamiento reactivo "move-to-goal", basado en campos de potencial, por ejemplo
  - Hasta que llego al objetivo global.

Problemas:

- Obsesión por el subobjetivo
- No hay replanificación oportunística

- Gastar tiempo y energía intentando alcanzar la posición exacta del subobjetivo.
- Porque la condición de terminación se ha puesto con una tolerancia no realista
  - Por ejemplo: ¿ya estoy en la (6,3, 90º)?
  - Estando en la (6,2.7, 90º) sigue moviéndose y llega a la (6,3.2,90º)....
  - Solución1: tolerancia
    - (abs(y_actual - y_goal) < epsilon)
  - Solución2: timeout
    - si no alcanzo el objetivo en un tiempo dado, o devuelvo error, o me quedo en un nuevo estado ("estoy perdido")

Replanificación contínua

- La obsesión puede ser porque ha cambiado el entorno y el mapa "a priori" ya no es correcto
  - P.ej: el robot tiene una cámara que detecta terreno fangoso
  - Solución3:
    - Actualizar el mapa
    - Replanificar (con el D* por ejemplo)
    - En cada posición, conoce el camino óptimo
- Esto nos lleva a una replanificación contínua
  - Muy costosa (p.ej: en un rover planetario)
  - Sujeta a errores por sensores con ruido.

Replanificación basada en eventos

- En lugar de replanificar en cuanto se detecta un pequeño desvío del plan original
- Tomar una medida de "cómo de importante" es el desvío
- Solo replanificar en este caso.