



**DEPARTAMENTO  
DE COMPUTACION**

Facultad de Ciencias Exactas y Naturales - UBA

## TP4 - Planificación de caminos utilizando RRT

28 de octubre de 2016

Introducción a la Robótica Móvil

Grupo (número de grupo)

Integrante	LU	Correo electrónico
Schmit, Matias	714/11	matias.schmit@gmail.com
Negri, Franco	893/13	franconegri2004@hotmail.com



**Facultad de Ciencias Exactas y Naturales**  
Universidad de Buenos Aires

Ciudad Universitaria - (Pabellón I/Planta Baja)

Intendente Güiraldes 2610 - C1428EGA

Ciudad Autónoma de Buenos Aires - Rep. Argentina

Tel/Fax: (++54 +11) 4576-3300

<http://www.exactas.uba.ar>

# 1. Introduccion

Para este trabajo intentaremos resolver la planificacion de caminos de un robot movil utilizando la tecnica de Rapidly-exploring random tree. Para ello implementaremos el algoritmo y luego realizaremos diferentes experimentaciones sobre el mismo con la intención de ver como se comporta.

A continuación respondemos las preguntas requeridas en el informe.

*Explicar como definieron el "area cercana al goal". ¿Tomaron en cuenta el angulo  $\theta$ ? ¿Como?*

Para definir el area mas cercana al goal definimos un rectangulo con constantes `GOAL_AREA_X`, `GOAL_AREA_Y` con centro en el goal. Para considerar el angulo de las poses aleatorias que generamos, en caso de caer en el area definida anteriormente cercana al goal dicho angulo es aleatorio en un rango  $(\pi/2\theta_{goal}, \pi/2\theta_{goal})$ .

Las demas configuraciones, fuera del area definida, son completamente aleatorias.

*Explicar que definicion de distancia utilizaron. ¿Como integran el angulo  $\theta$ ?*

Para definir la distancia utilizamos una formula que pondera la distancia euclidea y la diferencia de orientaciones entre las dos poses.

Para definirlo mas precisamente, dadas dos configuraciones  $c_1$  y  $c_2$  la función de distancia estará dada por:

$$K_{dist}.distancia\_euclidea(C_1, C_2) + K_{Ori}.dist_{Ori}(C_1, C_2)$$

Donde  $K_{dist}$ ,  $K_{Ori}$  son los pesos de importancia que le daremos a cada una de estas metricas.  $distancia\_euclidea(C_1, C_2)$  es la norma 2 entre el x,y de cada configuración y  $dist_{Ori}(C_1, C_2)$  es el modulo de la distancia angular entre el  $\theta$  de  $c_1$  y  $c_2$

*Explicar como establecieron "discretizaron" el espacio de posibilidades a partir de la "configuracion mas cercana".*

Se cuenta con dos variables que nos permiten ajustar el stepping en velocidad lineal (`wx_step`) y angular (`wz_step`).

A partir de la configuracion mas cercana tomaremos un numero discreto de acciones posibles, dejando constante `wx_step` y variando la direccion en cero menos uno o uno `wz_step`.

De todas las discretizaciones posibles nos quedamos con aquella mas cercana al  $q_{new}$

*Explicar como resolvieron esta comprobacion.*

Para resolver la comprobacion de colisión, definimos un area rectangular alrededor de la nueva configuracion verificando si una cantidad de puntos de ese rectangulo estan o no libres. En caso de encontrarse todos libres decimos que no se producen colisiones en caso contrario diremos que hay colisiones y ese punto no será valido.

# 2. Experimentación

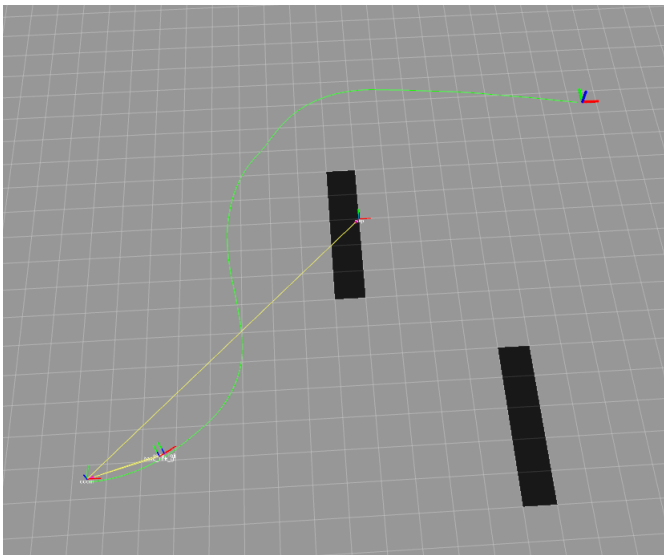
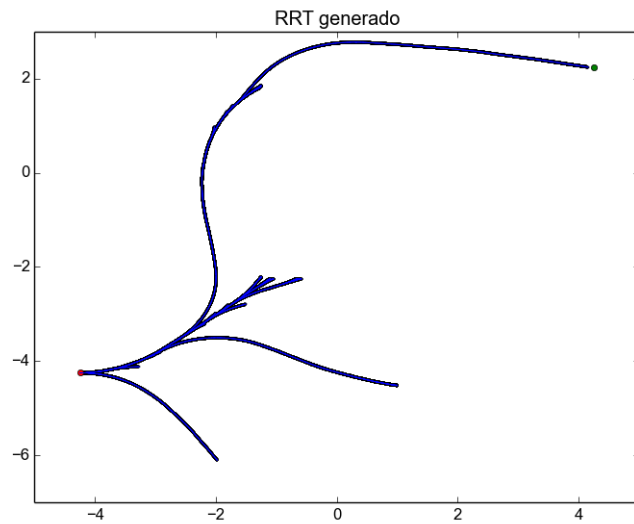
En esta sección veremos como afectan las variaciones en el stepping de velocidad y el goal bias al algoritmo y sacaremos las conclusiones convenientes.

Para toda esta sección se utilizaran los valores

- `GOAL_AREA_X` 1
- `GOAL_AREA_Y` 1
- `K_DIST` 0.8
- `K_ORI` 0.2

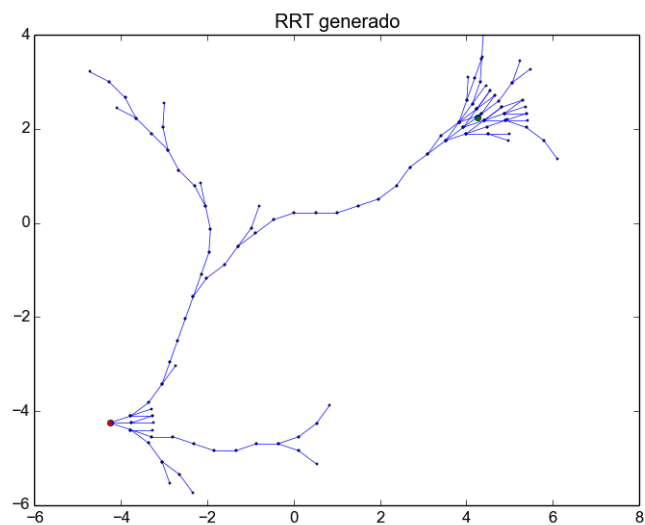
Como primera instancia variaremos el stepping de velocidad dejando constante el goal bias en 0,6.

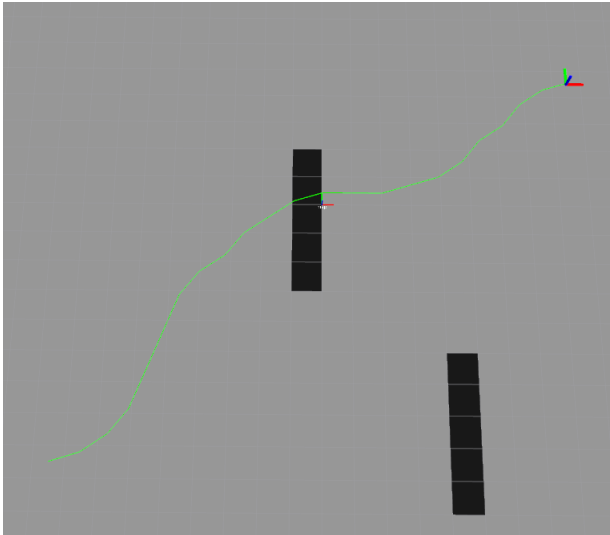
Variando el stepping en la velocidad lineal a 0,005 y variando el de la velocidad angular a 0,002 obtenemos el siguiente grafico:



Si bien llega al objetivo fueron necesarias muchas mas iteraciones de las que venian por default. Con 20000 iteraciones el arbol no lograba llegar a destino.

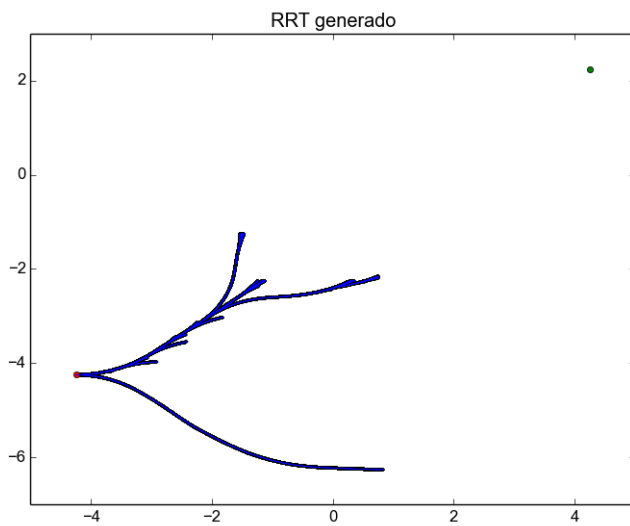
Como segunda prueba aumentamos el stepping a 0.5 y 0.3 respectivamente. El grafico resultante es el siguiente:





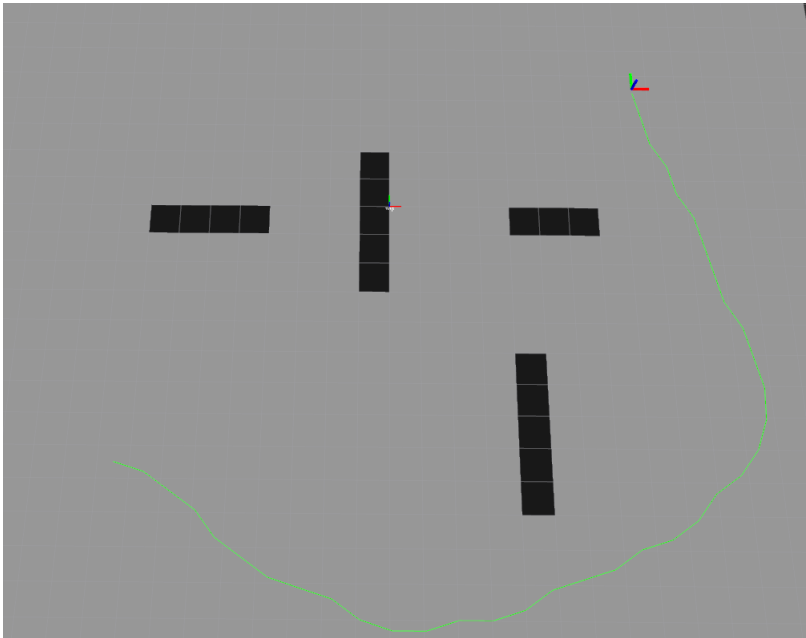
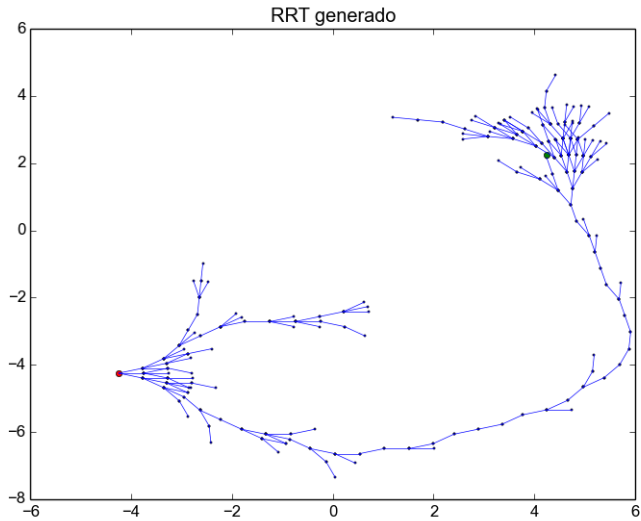
Si bien el algoritmo llego a destino, este camino no es valido, ya que como podemos ver en rviz pasa por entre medio de un obstaculo. Creemos que esto se debio a que la baja granularidad empeoró el calculo de las posiciones intermedias libres.

Realizamos la misma experimentación para la segunda escena provista por la catedra. Para el stepping bajo observamos el siguiente grafico:



Esto se debio a que posiblemente el pequeño stepping tanto en velocidad angular como en velocidad lineal no le permitio llegar al goal en las iteraciones provistas.

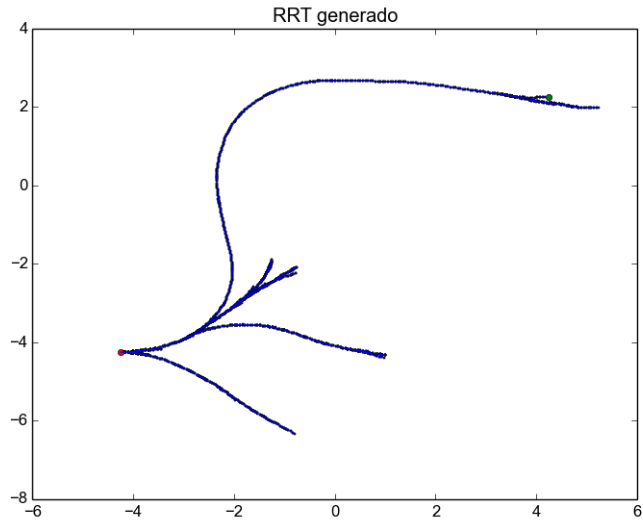
Para el stepping alto:



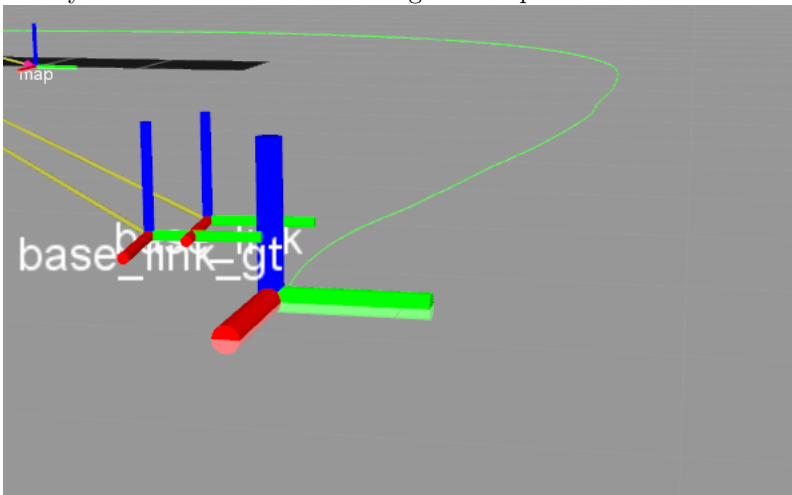
Es posible llegar a la solución sin embargo es posible observar al acercarse a la posición del goal no le dio en ángulo para correcto y tuvo que continuar con una exploración poco acertada. En cuanto a la experimentación sobre el goal bias llegamos a la conclusión de que valores muy altos y muy bajos hacen que no siempre se logre conseguir una trayectoria válida hacia destino.

Comenzamos mostrando un escenario con goal bias *alto* de 0,8, recordemos que esto nos dice que el 80% de las configuraciones aleatorias van a caer en el área *cercana* al goal que definimos.

En la escena: rrt\_pioneer\_planning el árbol resultante fue el siguiente:

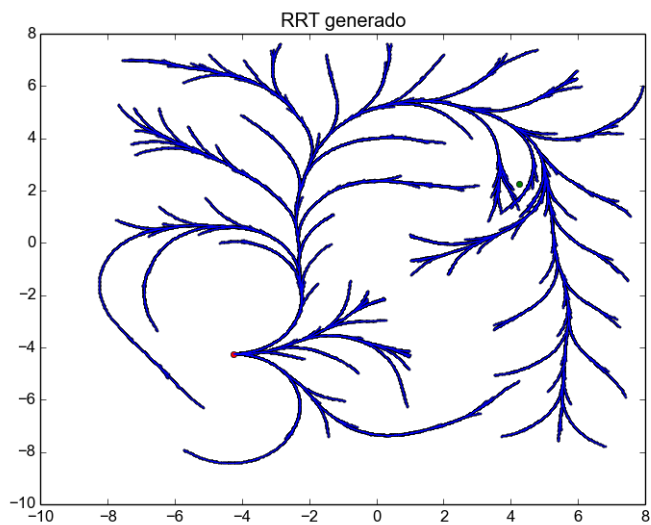


Si bien consideró un par de caminos que no lo llevaba al goal, la trayectoria es bastante directa y no se obtuvo un árbol muy ramificado. La elección del goal bias puede considerarse acertada para este escenario.



Podemos ver en rviz que las poses finales son bastante cercanas a la del goal

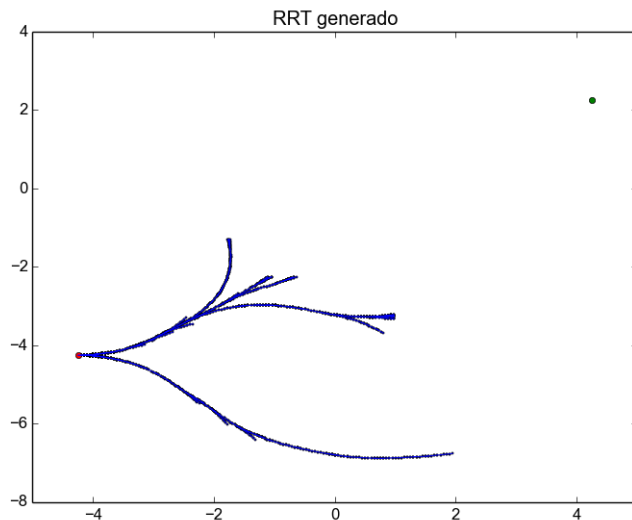
A medida que disminuimos este valor el árbol comienza a randomizarse, como caso extremo mostramos lo que sucede con *goal\_bias* 0,1, el árbol que se obtiene es este:



La gran diferencia de este caso con el anterior es el nivel de exploración sobre el mapa. Al tener casi todos los

puntos aleatorizados en cualquier posición (cercana o lejana al goal), los posibles caminos no están lo suficientemente sesgados para que se dirija al goal. Si bien algunos de los subcaminos se acercan nunca es lo suficiente a una pose (con énfasis en la orientación) parecida a la del destino.

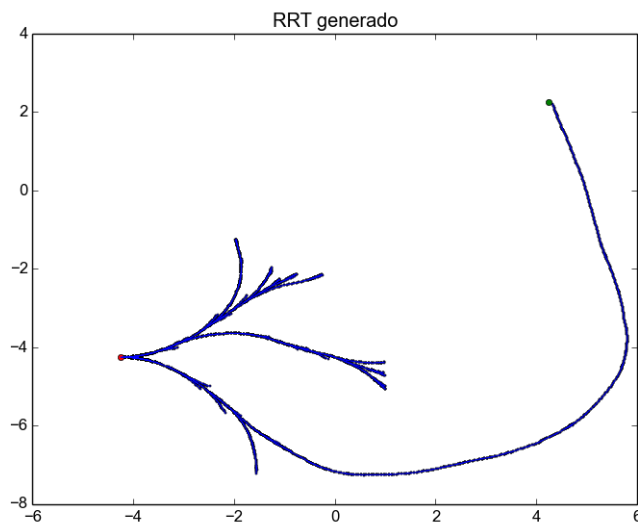
Para otros escenarios el comportamiento no es necesariamente similar, ya vimos que un goal\_bias bajo no nos daba buenos resultados, pero si mantenemos el 0,8 anterior ya no encuentra el trayecto



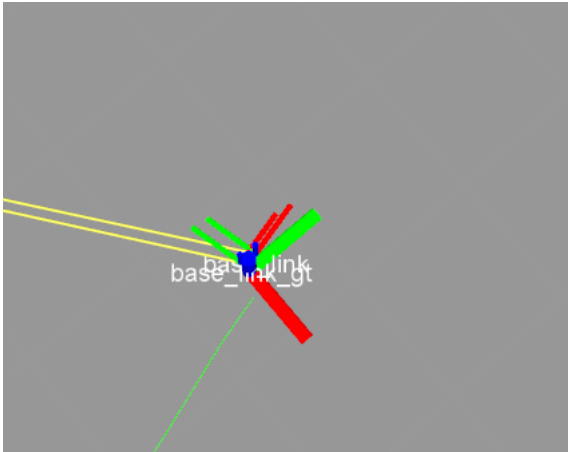
Lo que está pasando en este caso es que al tener muchos más obstáculos y tan pocos puntos aleatorios fuera del área cercana al goal es mucho mas costoso formar caminos que se alejen de los objetos que causan las colisiones.

Vale la pena destacar que agregando más iteraciones al algoritmo esto puede solucionarse (de todas formas no es la solución óptima al problema).

Para mejorar esto es conveniente no usar un goal tan cercano a 1, bajandolo a 0.6 logra mitigarse el problema anterior y se llega a destino.

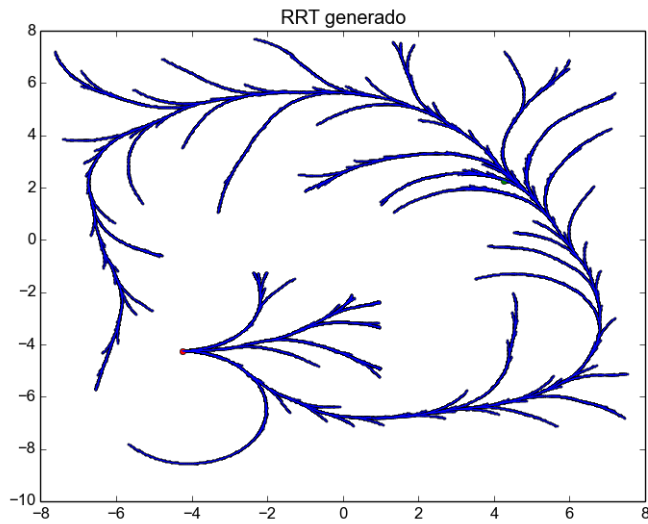


Los resultados son claros ya que hay poca diferencia entre los caminos que considera respecto al caso anterior, pero al elegir más configuraciones mejor distribuidas en el mapa sale de las zonas de colisiones y finalmente logra dirigirse al goal.



La pose final que encuentra no fue tan buena como en el escenario anterior pero sigue siendo una buena aproximación al goal.

Si bien no aporta mucho más a lo discutido anteriormente vale la pena mencionar que el caso con bias muy bajo tampoco es bueno en este escenario



En esta ocasión se acercó al goal pero ninguno de los caminos le permitió obtener una orientación cercana.