Introducción a la Robótica Móvil

Primer cuatrimestre de 2018

Departamento de Computación - FCEyN - UBA

Planificación de caminos - clase 16

Algoritmo RRT (*Rapidly Exploring Random Tree*)

Repaso: ¿qué estamos buscando?

- Podemos definir como camino a un mapeo continuo en el espacio de configuración tal que:

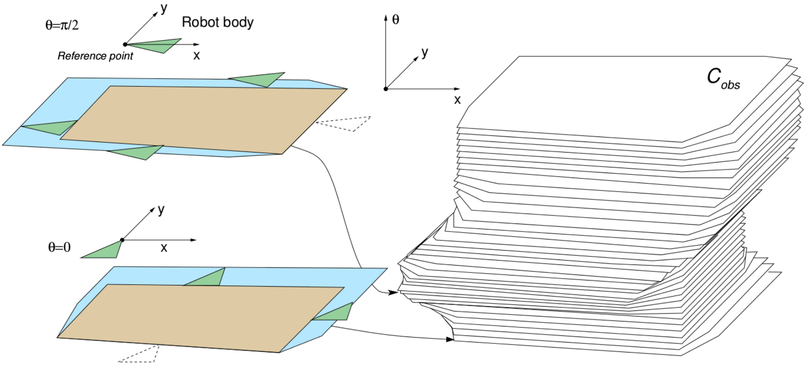
$$\pi : [0,1] \rightarrow \mathcal{C}_{free},\ \text{con } \pi(0) = q_0 \text{ y } \pi(1) = q_f,$$

  donde q_0 es la configuración inicial y q_f es la final.

- El problema de la planificación de caminos entonces consiste en encontrar la función $\pi(\cdot)$.

Ejemplo de $\mathcal{C}_{obs}$

Consideremos el espacio de los obstáculos de un robot con forma de triángulo donde consideramos la orientación θ:



- Un simple obstáculo en 2D deviene en un complicado $\mathcal{C}_{obs}$.
- Existen algoritmos determinísticos para computar $\mathcal{C}_{obs}$ pero requieren tiempo exponencial respecto de la dimensión de $\mathcal{C}$
- Una representación explícita de $\mathcal{C}_{free}$ es impráctica de computar.

Representación del espacio de configuración $\mathcal{C}-$space

¿Cómo lidiar con la representación del espacio de configuración?

Representación continua de $\mathcal{C}-$space, resulta intratable.

$\downarrow$

Discretización
Procesando geométricamente el espacio, haciendo muestreo aleatorio del espacio, con una descomposición en celdas, con campos potenciales, etc.

$\downarrow$

Técnicas de búsqueda en grafos
BFS, Dijkstra, A*

Planificación de caminos basada en muestro

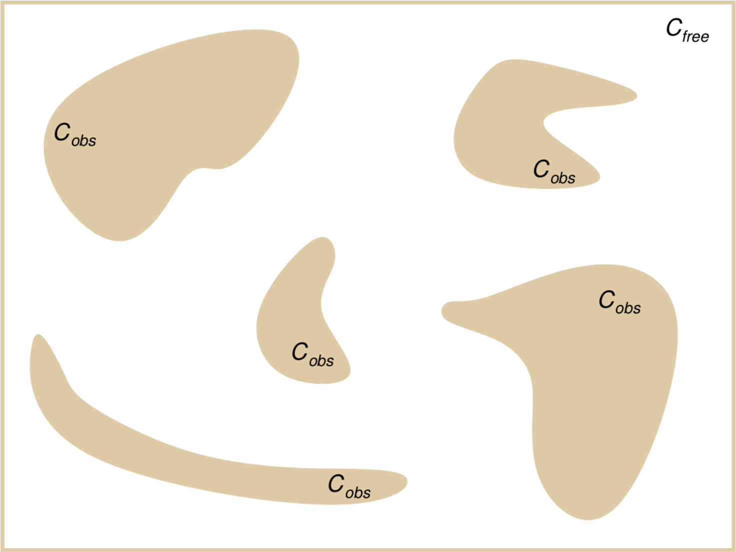
- Queremos evitar el cómputo explícito de la representación de $\mathcal{C}_{free}$ porque es muy costoso.
- En lugar de eso utilizamos una función que nos va a decir si una configuración $q()$ del robot colisiona con algún obstáculo. (por ejemplo basada en modelos greométricos y que testee la colisión entre los modelos)
- Discretizamos la representación de $\mathcal{C}$ y la muestreamos de manera aleatoria para construir posibles caminos u hojas de ruta (*roadmap*)
- En lugar de buscar completitud buscamos completitud probabilística, i.e, a medida que crece el número de muestreo mejora la solución que encontramos (si existe).

Roadmap probabilístico (PRM)

- Consiste en una representación discreta del espacio continuo $\mathcal{C}$ creada a partir de un muestreo aleatorio de configuraciones del robot en $\mathcal{C}_{free}$ que se conectan en un **grafo**.
- Los **nodos** del grafo representan una configuración admisible del robot.
- Los **ejes** del grafo representan una camino factible entre dos configuraciones.
- Conectamos la **posición inicial** y la **posición final** al grafo
- Encontramos un **camino** en este grafo (*roadmap*) entre la posición inicial y final.

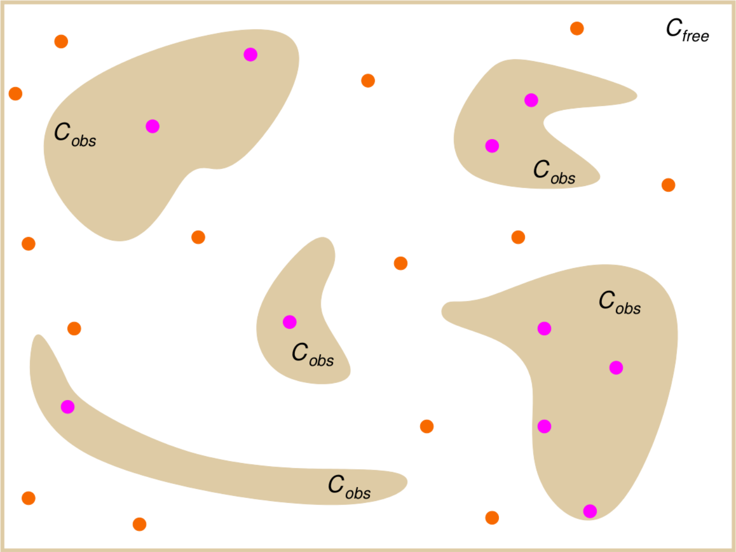
Roadmap probabilístico (PRM)

Sea un espacio de configuración $\mathcal{C}$



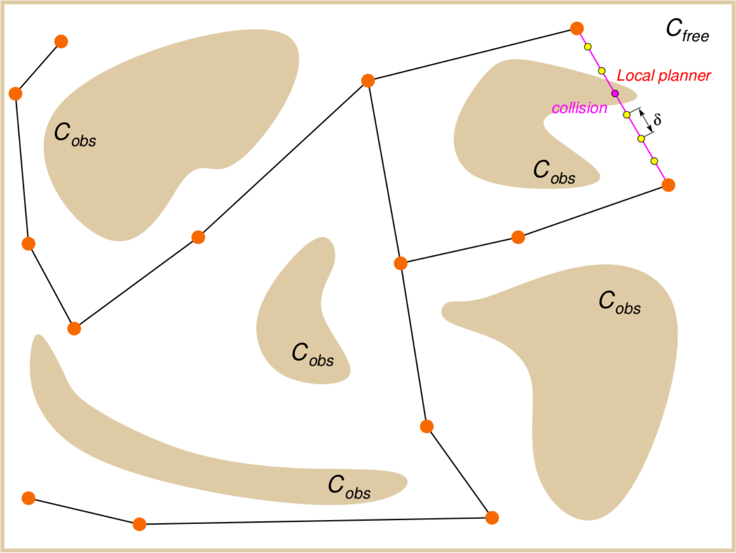
Roadmap probabilístico (PRM)

1) Muestreamos de manera aleatoria configuraciones admisibles del robot.



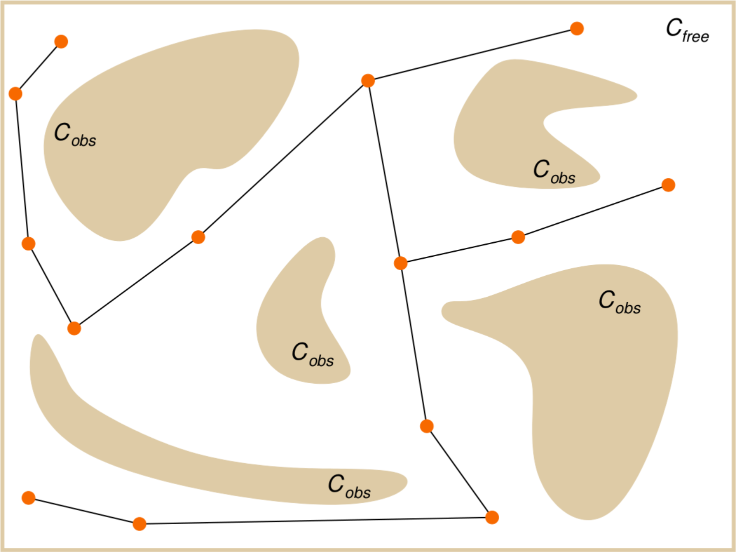
Roadmap probabilístico (PRM)

2) Conecto las muestras aleatorias armando un grafo libre de colisiones.



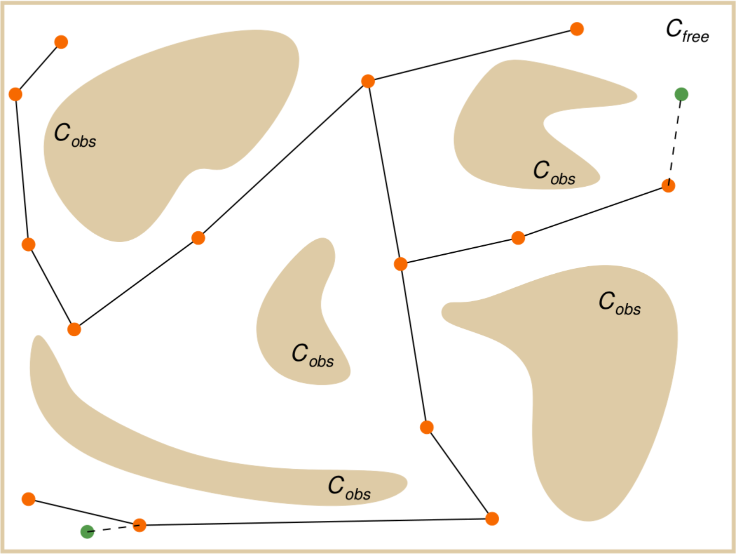
Roadmap probabilístico (PRM)

3) El grafo que obtenemos es el roadmap



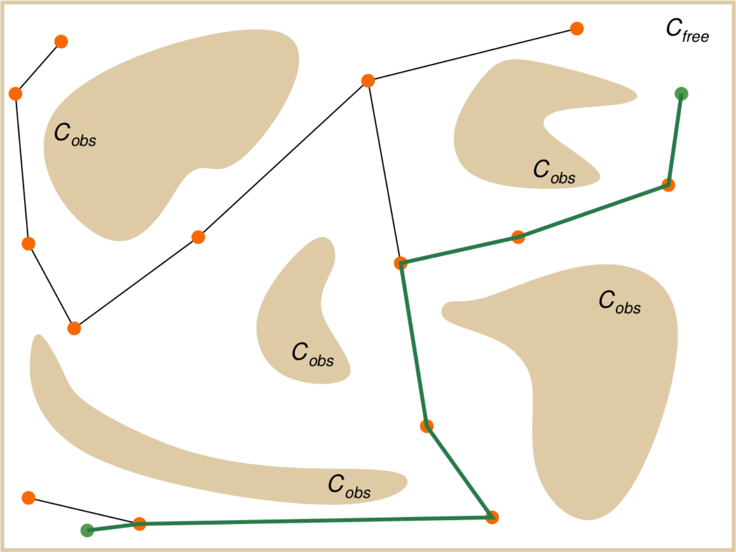
Roadmap probabilístico (PRM)

4) Unimos el punto inicial y el punto final al roadmap



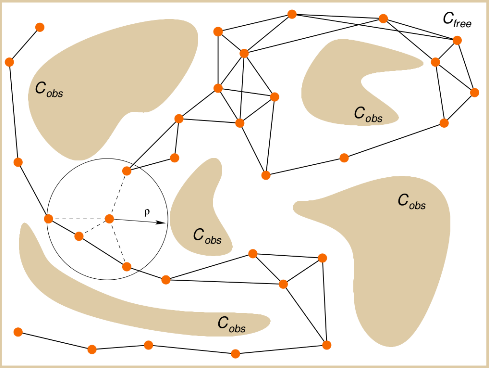
Roadmap probabilístico (PRM)

5) Obtenemos un camino en el roadmap.



PRM en la práctica

- Construimos el grafo de manera incremental.
- Conectamos los nodos en un radio ρ.
- El planificador local testea colisiones dentro de ese radio ρ.
- El camino mínimo puede ser encontrado con el algoritmo de Dijkstra.



Algoritmo PRM

Algorithm 1 PRM(q_{init}, q_{goal}, número de muestras n, radio ρ) $\rightarrow G = (V, E)$

$V \leftarrow \{q_{init}, q_{goal}\} \cup \{muestra \in C_{free}\}_{i=1,\ldots,n-1}$
$E \leftarrow \emptyset$
for each $v \in V$ **do**
  $U \leftarrow Neighbors(G = (V, E), v, \rho) \setminus \{v\}$
  for each $u \in U$ **do**
    if $CollisionFree(v, u)$ **then**
      $E \leftarrow E \cup \{(v, u), (u, v)\}$
    end if
  end for
end for
return $G = (V, E)$

Existen muchas formas de conectar los vércticos del conjunto U

- Clique de radio ρ
- Los k vecinos más cercanos a v
- Un número variable dentro del radio ρ que depende del n.

Algoritmo PRM: pros y contras

Pros:

- PRM es un algoritmo probabilisticamente completo
- Se puede aplicar fácilmente a espacios de configuraciones de gran dimensionalidad.
- Una vez que construimos el roadmap es muy rápido poder calcular el camino mínimo entre el punto inicial y el punto final.

Contras:

- PRM no funciona bien en algunos casos como por ejemplo cuando hay pasajes angostos entre dos $\mathcal{C}_{obs}$.
- Tenemos que muestrear mucho para encontrar soluciones en algunos problemas.
- No queda claro de qué manera muestrear: uniforme, en los bordes de los $\mathcal{C}_{obs}$, gaussiana, etc.

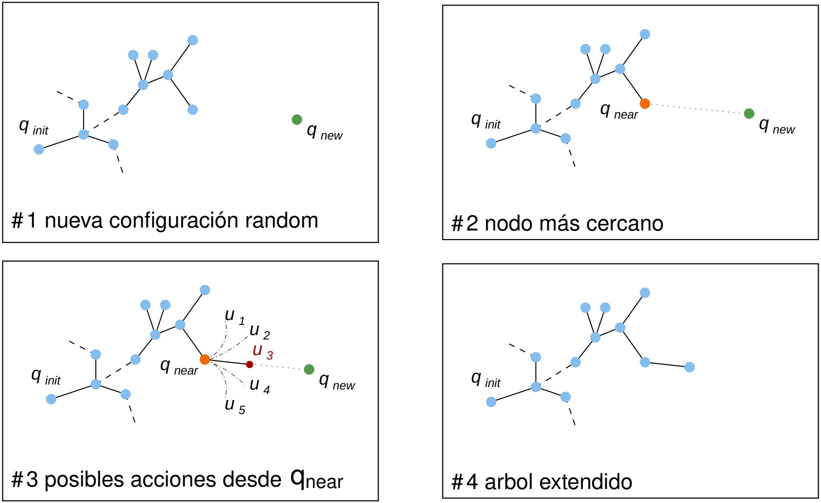
Rapidly Exploring Random Tree (RRT)

- La motivación es el uso de un solo *query* y una planificación basada en control
- Construye un grafo (árbol) en forma **incremental** hacia el *goal*

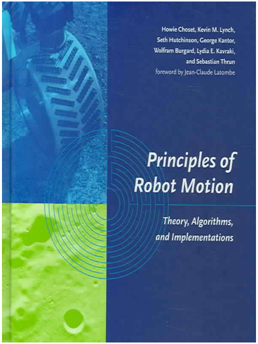
Algorithm 2 RRT(q_{init},n : samples) $\rightarrow G = (V, E)$

$V \leftarrow \{q_{init}\}$; $E \leftarrow \emptyset$
for $i = 1 \ldots n$ **do**
    $q_{rand} \leftarrow$ SampleFree
    $q_{nearest} \leftarrow$ Nearest($G = (V, E), q_{rand}$)
    $q_{new} \leftarrow$ Steer($q_{nearest}$, q_{rand})
    if CollisionFree($q_{nearest}$,q_{new}) **then**
        $V \leftarrow V \cup \{q_{new}\}$
        $E \leftarrow E \cup \{(q_{nearest}, q_{new})\}$
    end if
end for
return $G = (V, E)$

Rapidly Exploring Random Tree (RRT)



La expansión es repetida hasta que q_{goal} es alcanzado o la cantidad máxima de iteraciones es alcanzada.



"Principles of robot motion: theory, algorithms, and implementation", Howie Choset, Kevin Lynch, Seth Hutchinson, George Kantor, Wolfram Burgard, Lydia Kavraki, Sebastian Thrun, MIT press, 2005