Robótica II

Tarea 1. RRT en sistemas no holonómicos
19 de septiembre de 2020

Marco Antonio Esquivel Basaldua

I. Introducción

La planeación de trayectorias en robótica es un problema clásico. Un robot de ciertas dimensiones pretende navegar desde un punto A hasta un punto B, evitando un conjunto de obstáculos en el ambiente, $\mathcal{C}_{obs}$.

Existen numerosos algoritmos para la planificación de trayectorias. El algoritmo *Rapidly-Exploring Random Tree* (RRT) es una opción muy común en la que se genera un grafo y una trayectoria dentro de éste. Esta trayectoria no es necesariamente óptima en la métrica que se utilice para medir la "distancia"para conectar dos puntos en el espacio de configuraciones $\mathcal{C}$.

En este trabajo se muestra la implementación del algoritmo RRT para la planeación de rutas en sistemas no holonómicos dentro de un espacio cartesiano de dos dimensiones. Los sistemas a considerar son un robot de control diferencial (DDR) y un robot tipo carro (Car-like robot). Se muestran los resultados obtenidos tras aplicar el algoritmo a distintos ambientes con y sin obstáculos.

II. Algoritmo RRT

La premisa de este algoritmo es generar puntos de manera aleatoria y conectarlos al nodo más cercano posible. Cada vez que se crea un vértice, se debe revisar que esté en un espacio libre de colisiones y que el arista que lo conecte con su respectivo nodo también lo esté. El algoritmo termina cuando se genera un vértice dentro de la región de meta, o cuando un máximo de nodos es generado. El algoritmo es muy sencillo de implementar, se presenta a continuación.

Algorithm 3: RRT

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

Este método requiere menor poder de cómputo, es más simple de implementar y es más rápido que otros métodos de planeación de trayectorias, sin embargo requiere más nodos para ser almacenados. Su parte más costosa es encontrar el vecino más cercano a una muestra ya que este proceso crece conforme aumenta el número de vértices generados.

III. Sistemas no holonómicos

Como es conocido en el ámbito de robótica móvil, sistemas que presentan restricciones definidas en términos de derivadas en el tiempo de variables de configuración y que no pueden ser eliminadas por integración, son conocidos como sistemas no holonómicos. En estos sistemas, lo mejor que se puede determinar es una relación diferencial entre los estados del sistema y las entradas, no es posible determinar en forma cerrada una relación geométrica. Contrario a estos sistemas, en un sistema holonómico el robot se puede mover en cualquier dirección en el espacio de configuraciones.

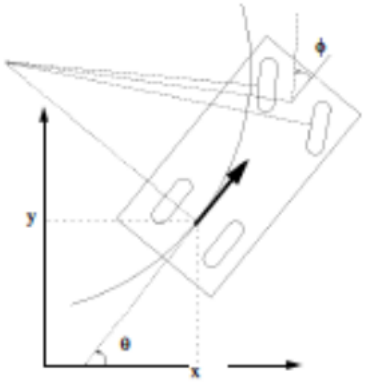

Figura 1. Car-like robot.

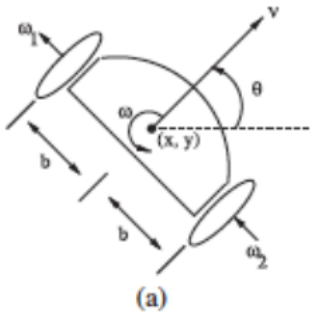

Figura 2. Differential drive robot DDR.

Dos de los sistemas no holonómicos más comunes son el robot tipo carro (Car-like robot), figura 1, y el robot de control diferencial (DDR), figura 2. En el primero de ellos se cuenta con cuatro llantas, en las que las dos traseras rotan con la misma dirección y velocidad mientras que existe un volante que se encarga de modificar el ángulo ϕ el cual indica la dirección del robot. Un primer modelo que se puede generar

a partir de la figura 1 es el siguiente

$$\begin{aligned}\dot{x} =& vcos(\theta)cos(\phi)\\ \dot{y} =& vsin(\theta)cos(\phi)\\ \dot{\theta} =& vsin(\phi)\end{aligned}$$

Definiendo

$$\begin{aligned}u_1 =& vcos(\phi)\\ u_2 =& vsin(\phi)\end{aligned}$$

Con motivo de simplificar el modelo generado por estas ecuaciones se añaden las siguientes restricciones

$$|u_1| \leq 1;\ |u_2| \leq 1$$

adicionalmente se debe tomar en cuenta que

$$-\frac{\pi}{4} < \phi < \frac{\pi}{4}$$

El modelo puede ser escrito como en la ecuación (1)

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}cos(\theta) & 0\\ sin(\theta) & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}u_1\\ u_2\end{bmatrix} \tag{1}$$

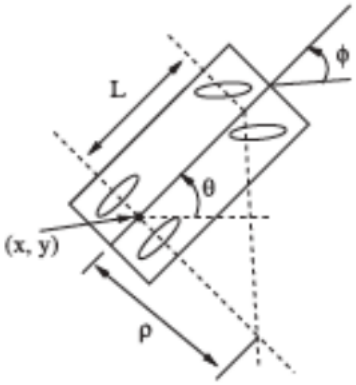

Figura 3. Car-like robot, segundo modelo.

Un modelo adicional para el sistema Car-like robot se obtiene a partir de la figura 3 como

$$\begin{aligned}\dot{x} =& vcos(\theta)cos(\phi)\\ \dot{y} =& vsin(\theta)cos(\phi)\\ \dot{\theta} =& v\frac{tan(\phi)}{L}\end{aligned}$$

Definiendo ahora

$$\begin{aligned}u_1 =& vcos(\phi)\\ u_2 =& v\frac{tan(\phi)}{L}\end{aligned}$$

Se obtiene el mismo modelo de la ecuación (1).

Por otro lado, un robot de control diferencial (Differential Drive Robot), figura 2, es un robot móvil cuyo movimiento se basa en el manejo de dos ruedas puestas a cada lado de la estructura móvil del robot. La dirección del robot puede ser variada de acuerdo a la rotación de cada una de las ruedas por lo que no es necesario el uso adicional de un volante.

Si las dos ruedas tienen la misma dirección y velocidad, el robot se mueve en línea recta . Si las ruedas se mueven con la misma velocidad pero con direcciones opuestas, el robot realiza una rotación en sitio. De otra forma, dependiendo de las velocidades y direcciones de cada una de las llantas, el robot describe un arco de círculo cuyo centro de rotación varia de acuerdo a las velocidades y direcciones de cada llanta, expresadas como ω_l y ω_r.

Considerando que el DDR se mueve en un espacio de dos dimensiones, las ecuaciones en espacio de estados no lineal están dadas por

$$\begin{bmatrix}\dot{x}\\ \dot{y}\\ \dot{\theta}\end{bmatrix} = \begin{bmatrix}cos(\theta) & 0\\ sin(\theta) & 0\\ 0 & 1\end{bmatrix}\begin{bmatrix}v\\ \omega\end{bmatrix} \tag{2}$$

Siendo x, y la posición del centro del eje del robot entre las rotaciones de las ruedas relativo al marco global, θ el ángulo relativo a la horizontal del marco de referencia global, v la velocidad lineal del robot, y ω la velocidad angular de la estructura.

Cada uno de los controles se puede expresar mediante la ecuación (3)

$$\begin{bmatrix}v\\ \omega\end{bmatrix} = \begin{bmatrix}\frac{\omega_l+\omega_r}{2}\\ \frac{\omega_l-\omega_r}{2}\end{bmatrix} \tag{3}$$

Sujeto a las restricciones

$$|\omega^{max}| \leq \frac{1}{b}(v^{max} - |v|)$$

IV. Implementación

Una forma de visualizar el funcionamiento del algoritmo RRT es generar una configuración x_{rand}, buscar el nodo de configuraciones más cercano x_{near} y generar un nuevo nodo x_{new} en la dirección $x_{near} \rightarrow x_{rand}$ a una distancia ϵ, tal como lo muestra la figura 4.

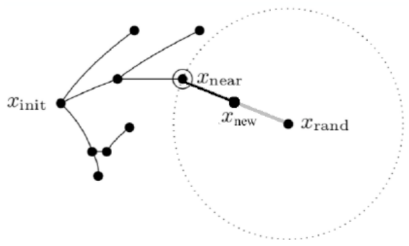

Figura 4. Se añade un nuevo nodo en el algoritmo RRT.

Sin embargo, este proceso de agregar nodos no es posible para el caso de los sistemas no holonómicos debido a las restricciones de velocidad de los mismos. Una alternativa es, en lugar de conectar dos configuraciones mediante una línea recta, generar un set de controles aplicados durante un tiempo Δ_t fijo a partir de x_{near} y conservar aquel control que resulte en la ubicación más cercana a x_{rand}, tal como lo muestra la figura 5.

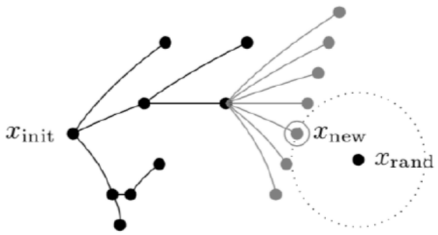

Figura 5. Se añade un nuevo nodo en el algoritmo RRT mediante la aplicación de controles.

La métrica usada para determinar la cercanía de un nodo a otro es la métrica de estados en $\mathbb{R}^2 \times S^1$ de la ecuación (4).

$$d(X, X') = \sqrt{(x - x')^2 + (y - y')^2 + \alpha^2} \tag{4}$$

$$\alpha = min\left\{|\theta - \theta'|, 2\pi - |\theta - \theta'|\right\}$$

Los algoritmos usados son los siguientes.

Algorithm 1: BuildRRT($x_{\text{init}}, \mathcal{X}_{\text{goal}}$)

$V \leftarrow \{x_{\text{init}}\}$;
$E \leftarrow \emptyset$;
while $V \cap \mathcal{X}_{\text{goal}} = \emptyset$ **do**
  $x_{\text{rand}} \leftarrow$ SampleState();
  $x_{\text{near}} \leftarrow$ NearestNeighbor(V, x_{rand});
  $(x_{\text{new}}, u_{\text{new}}, \Delta t) \leftarrow$ NewState($x_{\text{near}}, x_{\text{rand}}$);
  if CollisionFree($x_{near}, x_{new}, u_{new}, \Delta t$) **then**
    $V \leftarrow V \cup \{x_{\text{new}}\}$;
    $E \leftarrow E \cup \{(x_{\text{near}}, x_{\text{new}}, u_{\text{new}}, \Delta t)\}$;
return (V, E);

Algorithm 2: NewState($x_{\text{near}}, x_{\text{rand}}$)
(using fixed time step and best-input extension)

$\boldsymbol{u}_{\text{new}} \leftarrow \arg\min_{\boldsymbol{u} \in \mathcal{U}} \{\rho(\text{Simulate}(\boldsymbol{x}_{\text{near}}, \boldsymbol{u}, \Delta t), \boldsymbol{x}_{\text{rand}})\}$;
$\boldsymbol{x}_{\text{new}} \leftarrow \text{Simulate}(\boldsymbol{x}_{\text{near}}, \boldsymbol{u}_{\text{new}}, \Delta t)$;
return $(x_{\text{new}}, u_{\text{new}}, \Delta t)$;

V. Resultados en simulación

Los algoritmos y ecuaciones de la sección anterior se implementaron en el lenguaje de programación Python haciendo uso del módulo pygame para visualización. A continuación se muestran los resultados para cada uno de los sistemas no holonómicos descritos con anterioridad en un ambiente con y sin obstáculos.

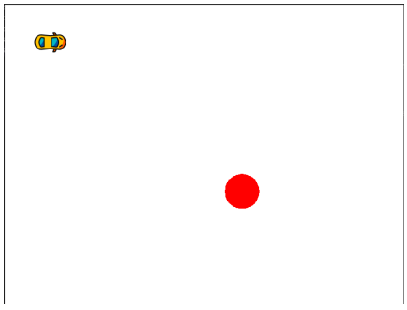

Figura 6. Posiciones inicial y final para un Car-like robot en un ambiente sin obstáculos.

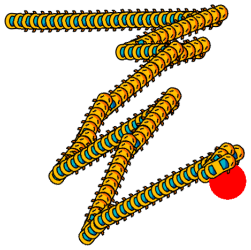

Figura 7. Trayectoria generada para el caso de la figura 6.

Figura 8. Posiciones inicial y final para un Car-like robot en un ambiente con obstáculos.

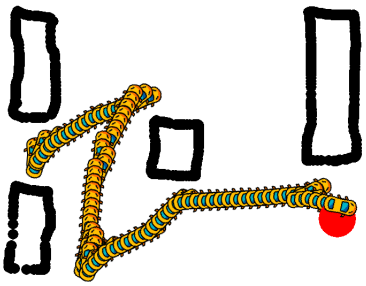

Figura 9. Trayectoria generada para el caso de la figura 8.

Figura 10. Posiciones inicial y final para un DDR en un ambiente sin obstáculos.

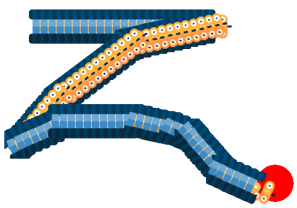

Figura 11. Trayectoria generada para el caso de la figura 10.

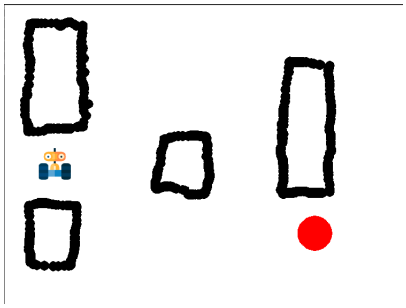

Figura 12. Posiciones inicial y final para un DDR robot en un ambiente con obstáculos.

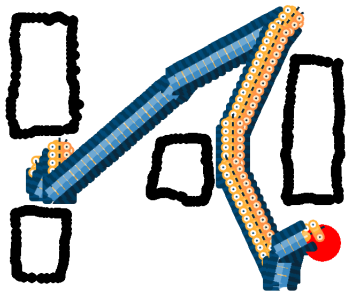

Figura 13. Trayectoria generada para el caso de la figura 12.

Adicional a estos resultados, se incluyen videos mostrando el movimiento de cada sistema a su destino. Estos videos no coinciden necesariamente con los resultados mostrados en este reporte.