Ignacio LLorente Díaz

Resumen del código RRT_kinodinamico:

Simulador del SIAR implementado un RRT sobre un mapa 2D de un alcantarillado predefinido.

Introducción

Este documento describe la implementación de un planificador de trayectorias basado en el algoritmo RRT (Rapidly-exploring Random Tree) aplicado al robot SIAR.
El SIAR es un robot de seis ruedas diseñado para operar en entornos subterráneos, con la particularidad de que puede variar su ancho para adaptarse a diferentes condiciones del entorno.
El objetivo principal del código es planificar trayectorias seguras y estables, garantizando que el robot evite colisiones y conserve su estabilidad.

Estructura general del código

El código se organiza en varios bloques que cumplen funciones específicas, lo cual favorece la modularidad y facilita la comprensión:

1. Parámetros del mapa y del robot: definen las dimensiones físicas, la escala y los umbrales de operación.
2. Tabla de configuraciones: relaciona el ancho del robot con la posición del centro de gravedad (CoG).
3. Dinámica y propagación: simulan el movimiento del robot bajo un conjunto de controles discretos.
4. Validación de configuraciones: verifica que las poses generadas sean viables y estables.
5. Planificador RRT: núcleo que expande el árbol de búsqueda hasta alcanzar el objetivo.
6. Interfaz gráfica: permite al usuario seleccionar visualmente el inicio y la meta, además de observar el crecimiento del árbol y la trayectoria final.

Funciones principales

A continuación se describen las funciones más relevantes del programa, clasificadas según su propósito:

1. Conversión de unidades: m2px / px2m convierten entre metros y píxeles en el mapa.
2. Geometría del robot: robot_polygon(state) genera el polígono del cuerpo; wheel_centers_pixels(state) calcula las posiciones de las ruedas.
3. Apoyos y estabilidad: support_points_pixels(...) determina qué ruedas se apoyan en zonas válidas; cog_pixel_from_table(state) obtiene la posición del CoG.
4. Validación: valid_configuration(smap, state) combina comprobación de colisiones, apoyo mínimo de ruedas y que el CoG esté dentro del polígono de soporte.
5. Dinámica: propagate(state, control, smap) simula la evolución del robot aplicando un control durante un tiempo discreto.
6. Métrica de distancia: state_distance(a, b) mide similitud entre estados considerando posición, orientación y ancho.
7. Planificación: plan_rrt(...) implementa el ciclo de muestreo, búsqueda del nodo más cercano, propagación y validación, hasta conectar con la meta.
8. Visualización: draw_state(...) dibuja el robot y sus apoyos; la clase Picker gestiona la selección de inicio y meta con el ratón.

Mecanismo de cambio de controles

Un aspecto central es el mecanismo de cambio entre conjuntos de control.
El planificador utiliza un conjunto de movimientos básicos (avanzar recto o con giros suaves) y otro específico para maniobras en caso de atasco.
El cambio entre conjuntos no depende de sensores, sino de un contador de iteraciones con fallos o éxitos, con histéresis y un periodo de enfriamiento.
De esta manera, se evita que el algoritmo oscile constantemente entre modos.

Conclusiones

En conclusión, el código implementa un planificador RRT kinodinámico adaptado al robot SIAR, integrando tanto la viabilidad geométrica como la estabilidad física en la validación de estados.
La estructura modular, junto con la visualización interactiva, lo convierten en una herramienta útil para experimentar con la planificación de trayectorias en entornos subterráneos restringidos.