Navegación de Robot con ArUcos: Estado del Arte Simplificado

Esta presentación describe un sistema de navegación robótica que utiliza marcadores ArUco para guiar a un robot omnidireccional a través de un laberinto.

1. Introducción

El objetivo es desarrollar un algoritmo que permita a un robot móvil navegar autónomamente por un laberinto utilizando información visual de una cámara. El robot utiliza marcadores ArUco de diferentes IDs colocados estratégicamente en el laberinto. Cada ID indica una acción específica (giro a la izquierda, derecha, 180° o 360°).

2. Arquitectura del Sistema

- **PyBullet:** Proporciona un entorno virtual para el laberinto y el robot.
- **Robot:** Capaz de moverse en cualquier dirección del plano XY.
- **Cámara:** Montada en el robot para capturar imágenes del entorno.
- **ArucoHunting:** Procesa las imágenes de la cámara para identificar y localizar los ArUcos.
- **MoveOmni:** Controla el movimiento del robot hacia las posiciones objetivo.

3. Flujo del Algoritmo

1. **Actualizar Cámara:** Obtiene una nueva imagen de la cámara.
2. **Actualizar Movimiento:** Mueve el robot hacia la posición objetivo actual, si existe.

3. Flujo del Algoritmo

3. **Calcular Siguiente Posición (get_next_pose):**
    - Detecta ArUcos en la imagen.
    - Si se detecta un ArUco:
        - Calcula la posición objetivo (x, y) y la orientación (rz) en función del ID del ArUco.
    - Si no se detecta un ArUco:
        - Realiza un pequeño movimiento de búsqueda.
4. **Actualizar Posición Objetivo:** Envía la nueva posición objetivo al controlador de movimiento.

4. Detección de ArUcos y Cálculo de la Posición Objetivo

La función `get_next_pose()` es el núcleo del algoritmo. Utiliza la clase `ArucoHunting` para detectar ArUcos en la imagen de la cámara. Si se detecta un ArUco, calcula la siguiente posición objetivo (`next_pose = [x, y, rz]`) en función de la posición del ArUco y la acción asociada a su ID.

Si no se detecta un ArUco, realiza un pequeño movimiento de búsqueda para intentar encontrar uno.

5. Control de Movimiento

La clase `MoveOmni` se encarga de controlar el movimiento del robot. Recibe la posición objetivo (`next_pose`) y mueve el robot suavemente hacia esa posición.

6. Optimizaciones

- **Parámetros de Detección Dinámicos:** Ajusta los parámetros de detección de ArUcos en función de la distancia estimada al siguiente marcador.
- **Movimiento Basado en Objetivos:** Mueve el robot directamente a las posiciones objetivo en lugar de realizar movimientos incrementales, lo que mejora la eficiencia y evita giros infinitos.

7. Conclusión

Este sistema de navegación robótica demuestra cómo se pueden utilizar marcadores ArUco para guiar un robot a través de un laberinto. El uso de parámetros de detección dinámicos y movimiento basado en objetivos mejora la robustez y eficiencia del sistema.