Proyecto 1: Generación de trayectorias en ROS

J.Carlos Urteaga Reyesvera
C.U. 10912
Maestría en Ciencias en Computación

Resumen—

I. Introducción

La investigación en el área de robótica ha aumento en los últimos años, pues gran parte del desarrollo ha sido la integración entre el hardware (circuitos) y el software (algoritmos) con los que se han evolucionado los robots que cómo los robots móviles, cuadropteros y humanoides . A lo largo de los años, han existido diferentes herramientas para la integración de circuitos y algoritmos [1] y algunas de ellas son MRDS (Microsoft Robotics Studio), Urbi, OROCOS, MRPT y ROS.

MRDS es un ambiente de ventanas desarrollado por Microsoft cuya objetivo es de tipo académico, entretenimiento y comercial con una amplia variedad de hardware [3]. MRDS es una librería de procesos que permite intercambiar mensajes a través de un DSS (Decentralized Software Service, por sus siglas en ingles) que permite organizar múltiples servicios. Desafortunadamente la integración de nuevo hardware fue una limitación, pues el agregar hardware requería de un proceso complicado.

Por otra parte, Urbi es una plataforma de programación para la robótica y sistemas complejos [4]. Esta plataforma cuenta con un lenguaje interpretado, concurrente y dirigido por eventos. Además está enfocada en la creación de controladores y de robots reales como el humanoide NAO, NXT de Lego Mindstorms y Roomba. A diferencia de MRDS esta plataforma cuenta con un mercado específico para la implementación de controladores y es de propósito particular.

En el caso de OROCOS es una herramienta específica de tiempo real enfocado a manos robóticas y herramientas de control [5]. A diferencia de otros proyectos, es de propósito particular que soporta distintos proveedores de componentes eléctricos pero enfocados al área de control.

Otra herramienta de programación movil es MRPT (Mobile Robot Programming Toolkit) es una paqueteria multiplataforma con el objetivo de ayudar a los investigadores a diseñar eimplmentar algoritmos relacionados a SLAM (Simultaneous Localization and Mapping) y planeación de movimientos (esquivar obstáculos).

A diferencia de los herramientas anteriores ROS permite la integración con los diferentes mecanismos del robot con conectividad. Esta herramienta permite una fácil integración de actuadores y software de forma a través de mensajes planos permitiendo independencia entre el hardware y software. Además, esta plataforma permite integrar la generación de movimientos, percepción y planificación de tareas.

El presente documento muestra la generación de Trayectorias utilizando ROS y su implementación. La organización de del documento es la siguiente. En la segunda sección se presenta el marco teórico con los conceptos fundamentales de ROS, la generación de trayectorias y una descripción de los tipos de robots de acuerdo a su tipo de desplazamiento. En el Tercer capítulo se detalla el procesamiento que se utilizo para la generación de trayectorias y de que forma se implemento en ROS . La experimentación se muestra en el cuarto, finalmente en el quinto los resultados obtenidos y las conclusiones.

II. Marco Teórico

ROS es un programa de código abierto diseñado para ser un software intermediario entre componentes y algoritmos que cumple con las tareas básicas de un sistema operativo. Este sistema operativo provee de herramientas de abstracción de hardware, comunicación entre diferentes procesos y la posibilidad de comunicarse con distinas computadoras. Además provee herramientas y paquetes para obtener, desarrollar, escribir y ejecutar código de forma modular para replicar el trabajo en difernetes entornos.

Un sistema distribuido como ROS permite interconectar computadoras con actuadores y sensores. Algunos de los ejemplos que enfatizan esto es la idea de un conjunto de robots que cooperan para fin común. Además, gran parte de la comunidad de investigación y desarrollo han creado y apoyado la distribución de paquetes con diferentes objetivos en el área de robótica, tal es el caso de `rviz` (ROS computer vision package) y `turtlesim` (simulador de un robot) para la programación. El programa principal que se ejecuta en ROS `roscore` que permite la comunicación entre procesos (nodos).

Los conceptos fundamentales para el uso de ROS son nodos, mensajes, tópicos y servicios [2]. La idea principal es suscribirse (obtener) información en los nodos mediante mensajes que son publicados por tópicos y servicios. Los nodos se suscriben a los tópicos para obtener la información. En el caso estos son los procesos que se ejecutan y requieren o envian información de sensores o de decisiones. ROS está diseñado para ser modular a escala de grano fino, es decir, cada nodo se puede ver como un programa. La comunicación entre nodos se realiza a través de mensajes, estos son enviados por el sistema operativo. Los mensajes cuentan con un tipo de dato y estructura específica de tipo primitivos (enteros, punto flotante, booleanos, etc.) o compuestos cómo arreglos de cadenas. Un nodo envía un mensaje al publicar en un tópico específico (*topic*) que puede ser una simple cadena de caracteres. En el caso de que un nodo desee recibir información de otro, este se suscribirá al tópico específico para obtener la información. Este tipo de arquitectura permite que los suscriptores y los

generadores de información sólo reciban y envíen información sin tomar en cuenta a quien o de quien toman información.

Una de las tareas principales que tiene que realizar un robot es el desplazarse a un punto, esto se realiza a partir de la planificación de trayectorias para determinar el camino a seguir de acuerdo a las restricciones (ambiente o del robot).

La generación de trayectorias es el desplazar un robot del punto A a B sin coliciones en un tiempo específico que puede ser calculado de forma discreta o continua. La planeación de trayectorias es una área de gran importancia en la robótica pues permite a los robots en autónomos.

Un robot es holonómico es la relación entre los grados controlables de libertad y los grados de libertad del robot. Si el número de grados de libertad controlados son mayores o iguales a los del robot se dice que es holonómico. Usar este tipo de robots permite realizar las tareas. Un ejemplo de un caso no holonómico es el automóvil, ya que no puede girar sobre su eje y sólo se puede desplazar hacia adelante.

Existen diferentes algoritmos para la planeación de movimientos que va desde identificar todos los caminos posibles, (potencial Field Planning), planeación a través de muestro y a través de cuadrículas. El que usará en este trabajo será el primero pues suponemos que no existen obstáculos entre el punto de origen y destino y puede tomar cualquier ruta.

La herramienta que permite la simulación de un robot holonómico con una planeación de movimientos candidatos es el simulador `turtlesim`. Este simulador permite al robot girar sobre su propio eje pero sólo moverse hacia enfrente, es decir los grados de libertad de acuerdo a plano de coordenadas es x y θ.

III. Desarrollo

El desarrollo de la práctica se realizo de acuerdo a los tres primeros capítulos del libro [**?**]. Los primeros ejercicios fueron el generar una trayectoria aleatoria del simulador, el segundo escuchar las trayectorias y finalmente en el capitulo 7 se realizar De acuerdo a las restricciones del robot, se plantean las siguientes ecuaciones para poder desplazar al robot, donde $v*$ es la velocidad lineal. $\theta*$ es el ángulo de dirección, el controlador proporcional es γ y la velocidad está dada por k_v

$$v* = k_v\sqrt{(x*-x)^2 + (y*-y)^2} \tag{1}$$

$$\theta* = \tan^{-1}\frac{y*-y}{x*-x} \tag{2}$$

$$\gamma = k_h(\theta*\theta), K_h > 0 \tag{3}$$

$$k_v = d/t \tag{4}$$

Estas ecuaciones se usaron para poder trazar la trayectoria. El ejercicio se acoto a que la tortuga llegara a un punto.

IV. Implementación

La implementación fue realizada con `C++`, se utilizaron las librerías de ROS. El problema se dividío en Cuatro puntos, los cuales se resolvieron por etapas.

- Obtener información del usuario. Los datos del usuario se dividieron en dos. Por un lado se obtienen lel punto deseado al generar un tópico de `PositionCommand`. Este tópica toma un mensaje de tipo `geometry_msgs`. En el caso del tiempo se toma como argumento al momento de iniciar el programa.
- Calcular la velocidad. El calculo de la trayectoria se realizo con las ecuaciones descritas en la sección anterior.
- Trazar la trayectoria. Esta sección permité al alumno pensar un gran número de soluciones, en este caso se tomo el de retroalimentación Este método permite calcular la distancia cada tiempo de muestro, por lo que se pueden generar trayectorias suaves. Otras posibles soluciones hubiera sido girar la tortuga sobre el mismo eje, despalzarse y rotar.
- Rotar. Esta sección no se pudo implementar, ya que la tortuta recalculaba la ruta y se perdía de la ruta. Lo que se planteava era ir girando mientras que llegara o incluso girar sobre su eje. Desafortuvadamente la velocidad que se manejo no permiitio este punto

V. Demostración

Para visualizar el ejercicio realizado, es necesario descargar el archivo del repositorio, iniciar el roscore, turtlesim, iniciar nuestro programa y generar un tópico.

- `$roscore`
- `$rosrun turtlesim turtlesim_node`
- `$rosrun tarea01 tarea01 tiempo`
- `$rostopic pub /turtle1/PositionCommand geometry_msgs/Pose2D "x: 1.0 y: 2 theta: 0.01`

VI. Conclusión

Es posible utilizar ROS para simular trayectorias en un robot, este tipo de trabajos permite ver la facilidad de integrar los componentes para resolver problemas a otro nivel. Cómo trabajo futuro se queda generar un lunch file para ejecutar el ejemplo más facil

Referencias

[1] Harris, A., & Conrad, J. M. (2011, March). Survey of popular robotics simulators, frameworks, and toolkits. In Southeastcon, 2011 Proceedings of IEEE (pp. 243-249). IEEE.

[2] Quigley, M., Conley, K., Gerkey, B., Faust, J., Foote, T., Leibs, J., ... & Ng, A. Y. (2009, May). ROS: an open-source Robot Operating System. In ICRA workshop on open source software (Vol. 3, No. 3.2, p. 5).

[3] Morgan, S. (2008). Programming microsoft® robotics studio. Microsoft Press.

[4] Baillie, J. C., Demaille, A., Hocquet, Q., Nottale, M., & Tardieu, S. (2008, November). The Urbi universal platform for robotics. In First International Workshop on Standards and Common Platform for Robotics.

[5] Bruyninckx, H. (2001). Open robot control software: the OROCOS project. In Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference on (Vol. 3, pp. 2523-2528). IEEE.

[6] O'Kane, Jason M. .A gentle introduction to ROS."(2014): 1564.