Proyecto Robótica
Parte II

Enrique José Padilla Terrones, *Estudiante, ITAM*

Abstract

En el siguiente documento se reporta el procedimiento y resultados de la implementación de un programa ejecutado en ROS y apoyado en Gazebo, diseñado para completar uno de tres retos especificados más adelante en el documento.

Index Terms

Robótica, ROS, turtleSim, LaTeX, reporte.

I. Introducción

EL segundo proyecto del curso de Robótica consistió en completar exitosamente uno de tres retos. Partiendo del conocimiento y las habilidades adquiridas en la realización del primer proyecto, se busca crear un programa que pueda hacer uno de los siguientes tres retos.

1) Seguir una carretera curva definida por lneas en sus extremos.
2) Seguir a otro robot en una carretera recta.
3) Estacionarse en paralelo entre dos objetos fijos.

La resolución de estas tareas implica implementar en el programa conceptos de poses, trayectorias y modelos cinemáticos de robots que se abordaron en la parte teórica de la clase. La conjunción de estos elementos junto con habilidades de programación son fundamentales para el desarrollo de este tipo de proyectos.

II. Marco Teórico

Como fue establecido anteriormente, este proyecto requisa el conocimiento de conceptos fundamentales para el desarrollo del mismo. A continuación se exponen brevemente aquellos que son indispensables para comprender el proyecto.

1) **Poses** - Una pose es la descripción matemática de la posición y la orientación que tiene un robot respecto al cierto marco de referencia. Un robot puede ser descrito por muchas poses correspondientes a distintos marcos de referencia. Esta información es fundamental cuando se pretende crear un programa que haga que el robot tenga cierta interacción con el mundo en el que se encuentra; sea ir hacia un objetivo o evadir cierto objeto u obstáculo.

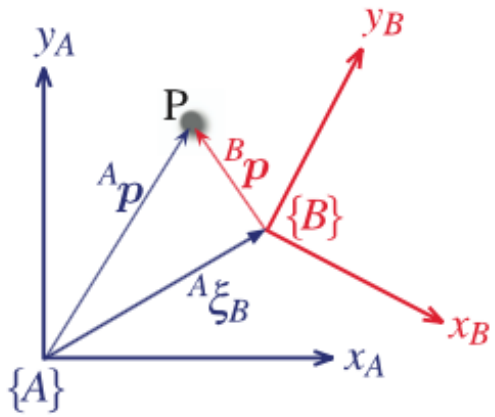

Fig. 1. Representación de la pose de P respecto a dos marcos de referencia distintos (A y B).

2) **Trayectorias** - Una trayectoria es una sucesión de poses, revisadas anteriormente. Estas poses caracterizan el movimiento que realiza un cuerpo cuando se mueve de un punto en el espacio a otro. Es importante notar que una trayectoria también tiene un tiempo definido como característica; en caso contrario, se estaría hablando de un camino solamente. Un requisito de las trayectorias mencionado por autores de libros de robótica, es que estas sean *suaves*, es decir, que los cambios de pose no sean muy marcados al pasar del tiempo. Es así como surge la necesidad de emplear métodos de derivación e interpolación para lograrlo. Uno de estos métodos fue desarrollado en el proyecto anterior.

3) **Modelo cinemático** - El modelo cinemático de un robot se refiere a sus propiedades físicas, mismas que limitan y determinan su movimiento e interacción con el espacio. En el modelo cinemático se especifican los grados de libertad del robot así como las articulaciónes y características que determinan su movimiento.

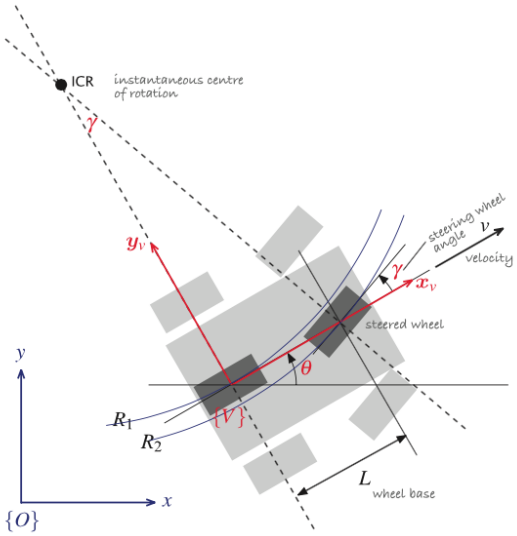

Fig. 2. Modelo cinemático de bicicleta, en donde un robot de cuatro ruedas se comporta mecánica y físicamente como si tuviera solo dos.

4) **Gazebo** - En adición a los conceptos teóricos fundamentales, es necesario conocer de Gazebo para poder entender el desarrollo del proyecto. Gazebo es un simulador que permite probar robots en un espacio tridimensional que permite probar algoritmos, diseñar robots y entornos además de contar con una interfaz gráfica para añadirle comodidad y experiencia de usuario a los proyectos.
5) **AutoNOMOS** - AutoNOMOS es un tipo de robot diseñado para el estudio y desarrollo de aplicaciones. Desafortunadamente, al ser una plataforma relativamente nueva, hay poca información de referencia. En este proyecto nos limitamos a referirnos al AutoNOMOS como el robot de la simulación.

III. Desarrollo/Experimentos

El desarrollo del proyecto se basó en un códigos preexistentes que ilustró el camino a seguir, proporcionado por Edgar Granados, estudiante del Instituto Tecnológico Autónomo de México (ITAM) y responsable del laboratorio de robótica de la institución. Los códigos mencinados fueron nodos que corren en ROS y Gazebo, permitiendo que los equipos se enfocaran directamente en la programaciíon y resolución de los retos sin tener que preocuparse por crear los nodos y los archivos necesarios para que estos fueran compilados y ejecutados. A continuación se detalla el proceso seguido para lograr cumplir el primer de los retos especificados al principio del documento. En algunos casos el código proporcionado ya realizaba la tarea correspondiente por lo que sólo hizo falta comprender la funcionamiento general y sobre todo cómo se enlazaba ese nodo con los siguientes.

A. Nodo de visión

El primer nodo que necesario para la ejecución del proyecto fue *vision node.cpp*. Este nodo toma el robot situado en Gazebo (diseñado y colocado por Edgar G.) y regresa lo que el robot ve. Es decir, regresa las lecturas del sensor que le permiten ”ver” al robot. Esto se logra por medio de datos de tipo PointCloud que este nodo pasará al siguiente.

B. Nodo de percepción

El siguiente nodo proporcionado fue el *perception node.cpp*. En este archivo, se cuenta con dos funciones principales que aportan la funcionalidad necesaria.

1) Función de filtrado: En primera instancia el equipo agregó un proceso de filtrado para las PointClouds que entregó el nodo de visión. La PointCloud que el AutoNOMOS entrega al programa está compueta en gran parte por elementos NaN, *Not a Number*. El primer proceso fue diseñado para filtrar los NaNs y así tener lecturas exclusivamente de números reales.

2) Función de cálculo de puntos medios: El segundo proceso implementado fue diseñado para tomar dos lecturas con números reales que se hayan hecho y por medio de un código sencillo, calcula el punto medio entre la recta que pasa por esos dos puntos. Esa línea recta generada será la que determine la trayectoria que el robot debe seguir en el espacio. El punto calculado, al cual el robot debe dirigirse es publicado en el tópico correspondiente para seguidamente lograr el movimiento en el siguiente nodo.

C. Nodo de control

El siguiente nodo empleado fue *autonomos ctrl node.cpp*. Este nodo es la piedra ángular del proyecto. El código está compuesto por tres tipos de funciones fundamentales que son los que implementan la funcionalidad requerida para el proyecto.

1) Función para conocer poses: Dentro de este rubro caen dos de las funciones del nodo. Estas funciónes son simples; se suscriben al tópico apropiado para traer al nodo información de interés como son la pose actual del robot y la pose que se desea alcanzar.

2) Funciones para generar movimiento: Una vez que conocemos la pose actual y la deseada del robot, calculamos la velocidad por medio de una constante (arbitrariamente asignada) y el ángulo necesario para que el robot se alinee con la pose deseada. La primera función diseñada, *isGoalFar* determina si la pose final está ”lejos” (determinado arbitrariamente también). En caso de estarlo, la constante para generar la velocidad es mayor a aquella que designaríamos si no lo estuviera. Todas estas magnitudes calculadas son publicadas en el tópico correspondiente.

3) Funciones de límite: La última clase de funciones implementadas en el nodo de control están diseñadas para limitar las magnitudes que puede alcanzar la velocidad o el ángulo de giro. Esto es necesario ya que las características y limitantes físicas del AutoNOMOS pueden generar un problema en la ejecución cuando estas magintudes rebasan cierto umbral.

D. Nodo de velocidad

El último nodo empleado fue *a robor velocity node.cpp*. Este nodo final es el que implementa el modelo cinemático adoptado (modelo de bicicleta) en el AutoNOMOS. Además de este fundamental rasgo, el nodo de velocidad se encarga de suscribirse al tópico de velocidad para seguidamente convertir estas velocidades calculadas anteriormente en velocidades para los motores (Joints) del AutoNOMOS.

La acción conjunta de todos los nodos explicados logra que el AutoNOMOS, corriendo sobre Gazebo, siga exitosamente una carretera con curvas, delimitada a los lados por líneas.

IV. Conclusiones

Después de mucho trabajo, tiempo y esfuerzo, se logró completar uno de los retos presentados al inicio del reporte. La simulación conduce con éxito al AutoNOMOS por la carretera trazada en Gazebo. La segunda parte del proyecto semestral del curso de Robótica fue una experiencia en extremo retadora, pero igualmente gratificante y enriquecedora. El proyecto permitió un profundo aprendizaje no sólo de conceptos de robótica, si no que también propició el aprendizaje de otro lenguaje de programación (c++), el uso de otro sistema operativo (Ubuntu), e inclusive de programación en consola, sin poder contar con una interfaz gráfica. Un posible mejora que cabe en el reto y el programa, sería lograr que el robot se detuviese una vez que la carretera ha finalizado. Sin embargo, el trabajo realizado hasta ahora deja al equipo sumamente satisfecho.

Agradecimientos

El autor extiende una agradecimiento a Diego Fernando Cifuentes Jimenez y a Alejandro Terminel por su dedicada ayuda, sin la cual la realización de este proyecto no hubiera podido ser posible.

References

[1] J.M. O'Kane, *A Gentle Introduction to ROS*, Abril 2016.
[2] Koubaa A., [Anis Koubaa]. (2015, Septiembre 29). *[CS460] ROS Tutorial 4.4: Go-To-Goal Location (Turtlesim Cleaner)*. URL: *https://www.youtube.com/watch?v=Qh15Nol5htM*
[3] Peter Corke, *Robotics, Vision and Control*, Septiembre 2011.

Enrique Jose Padilla Terrones Enrique nació en Knoxville, Tennessee, EEUU el 15 de agosto de 1995. En el verano del 2013 entraria al Instituto Tecnologico Autónomo de México a estudiar Ingeniería Mecatrónica e Ingeniería Industrial. Actualmente cursa el séptimo semestre de sus estudios en la Ciudad de México.