Instituto Tecnológico y de Estudios Superiores de Monterrey Campus Puebla

TE3003B.501

Integración de robótica y sistemas inteligentes

Reto Semanal 4 | Manchester Robotics

Frida Lizett Zavala Pérez A01275226

Diego Garcia Rueda A01735217

Alejandro Armenta Arellano A01734879

3 de Mayo del 2024

Mediante el proceso de obtención de incertidumbres gaussianas para *Puzzlebot*, un sistema no lineal, mediante la técnica de linealización alrededor de un punto operativo específico". Los mecanismos se utilizan para calcular la covarianza, considerando mediciones de incertidumbres gaussianas, proporcionando una herramienta esencial para comprender y predecir la variabilidad y precisión del sistema. Salida: Además, se describe la arquitectura ROS utilizada, incluyendo la implementación de una arquitectura de bucle abierto para obtener las constantes necesarias y la publicación de posiciones entre el modelo ideal y real de *Puzzlebot*. Los resultados obtenidos muestran la eficacia de la aproximación y la validez de las incertidumbres calculadas, destacando la importancia de este proyecto como base para futuros desarrollos.

Matriz de covarianza para localización

La matriz de covarianza es una herramienta estadística que permite entender cómo se relacionan diferentes variables entre sí, podría verse como una tabla que muestra las covarianzas entre las parejas posibles de variables, ayuda a entender la relación entre más de dos variables de manera eficiente.

En el contexto de localización la matriz de covarianza juega un papel importante para estimar la incertidumbre asociada a la posición y orientación del robot. Describe como la incertidumbre en una variable se relaciona con la incertidumbre de otras, por ejemplo la posición en el eje x con con la posición en el eje y o la orientación.

Solución del problema

Obtención de incertidumbres Gaussianas:

Gracias a que la linealización nos permite simplificar un sistema no lineal alrededor de un punto de operación específico es posible aproximar el comportamiento no lineal del *puzzlebot* mediante un modelo lineal en el entorno de la posición y orientación. Gracias a la linealización y a las fuentes de error en las mediciones mejor conocidas como incertidumbres gaussianas , es posible calcular la covarianza mediante la siguiente ecuación:

$$\Sigma_K = H_K \cdot \Sigma_{K-1} \cdot H_K^T + Q_K \tag{1}$$

$$\Sigma_K = H_K \cdot \Sigma_{K-1} \cdot H_K^T + \nabla_{wk} \cdot \Sigma_{\Delta k} \cdot \nabla_{wk}^T \tag{2}$$

Las ecuaciones (1) y (2) representan el cálculo para obtener la covarianza en una matriz 3x3 que representa una suma de dos multiplicaciones de 3 matrices siendo la principal diferencia que en la ecuación (1) se pone Q_K como variable y en la ecuación (2) se desglosa esa variable representando que $Q_K = \nabla_{wk} \cdot \Sigma_{\Delta k} \cdot \nabla_{wk}^T$, siendo el resto de variables matrices de diferentes dimensiones generando la siguiente operación de matrices:

$$\Sigma_K = (3x3 \cdot 3x3 \cdot 3x3 + 3x2 \cdot 2x2 \cdot 2x3) \tag{3}$$

La ecuación (3) representa las dimensiones de las matrices de cada variable cada una representando los siguientes valores:

$$H_K = \begin{bmatrix} 1 & 0 & -\Delta t * V * sin(\Theta)) \\ 0 & 1 & \Delta t * V * cos(\Theta)) \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

$$\Sigma_{\Delta k} = \begin{bmatrix} k_r\,|\Delta s_r| & 0 \\ 0 & k_l\,|\,\Delta s_l \end{bmatrix} \tag{5}$$

$$\nabla_{wk} = \frac{1}{2} r\Delta t \begin{bmatrix} \cos(s_{\theta,k-1}) & \cos(s_{\theta,k-1}) \\ \sin(s_{\theta,k-1}) & \sin(s_{\theta,k-1}) \\ \frac{2}{l} & -\frac{2}{l} \end{bmatrix} = \begin{bmatrix} \frac{1}{2}\cos\left(\theta+\frac{\Delta\theta}{2}\right) - \frac{\Delta s}{2b}\sin\left(\theta+\frac{\Delta\theta}{2}\right) & \frac{1}{2}\cos\left(\theta+\frac{\Delta\theta}{2}\right) + \frac{\Delta s}{2b}\sin\left(\theta+\frac{\Delta\theta}{2}\right) \\ \frac{1}{2}\sin\left(\theta+\frac{\Delta\theta}{2}\right) + \frac{\Delta s}{2b}\cos\left(\theta+\frac{\Delta\theta}{2}\right) & \frac{1}{2}\sin\left(\theta+\frac{\Delta\theta}{2}\right) - \frac{\Delta s}{2b}\cos\left(\theta+\frac{\Delta\theta}{2}\right) \\ \frac{1}{b} & -\frac{1}{b} \end{bmatrix} \tag{6}$$

La matriz (4) es el sistema cinético linealizado en el reto previo así como H_K^T representa esa misma matriz transpuesta esas dos matrices se multiplican con el valor previo de Σ_K que se inicializa en una matriz 3x3 con únicamente ceros , del otro lado de la suma tenemos la matriz (6) ∇_{wk} , esa matriz es proporcionada por el socio formador, siendo ∇_{wk}^T la matriz transpuesta , esas dos matrices se multiplican a la matriz (5) donde los valores *kr* y *kl* deben de ser tuneados mediante pruebas a lazo abierto, este proceso es explicado en la siguiente sección. por lo que el primer valor de cada matriz debería de quedar como se muestran en la matriz (7) y la matriz (8):

$$\boldsymbol{\Sigma_1} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0.1 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0.1 & 1 \end{bmatrix} + \begin{bmatrix} 0.5 & 0.01 & 0.01 \\ 0.01 & 0.5 & 0.01 \\ 0.01 & 0.01 & 0.2 \end{bmatrix} \tag{7}$$

$$\boldsymbol{\Sigma_1} = \begin{bmatrix} 0.5 & 0.01 & 0.01 \\ 0.01 & 0.5 & 0.01 \\ 0.01 & 0.01 & 0.2 \end{bmatrix} \tag{8}$$

Siendo la matriz (8) el primer valor resultante de Σ_K para posteriormente incorporar ese resultado a la ecuación y ahora multiplicar ese valor al resto de variables lo que genera que conforme avanza el tiempo la matriz crezca representando el error que se va acumulando como se muestra a continuación:

$$\boldsymbol{\Sigma_1} = \begin{bmatrix} 0.5 & 0.01 & 0.01 \\ 0.01 & 0.5 & 0.01 \\ 0.01 & 0.01 & 0.2 \end{bmatrix} \qquad \boldsymbol{\Sigma_2} = \begin{bmatrix} 0.998 & 0.0207 & 0.0180 \\ 0.0207 & 1.0040 & 0.0399 \\ 0.0180 & 0.0399 & 0.4000 \end{bmatrix} \tag{9}$$

Representando la matriz (9) los valores que se obtienen con respecto a (x,y, Θ) como se muestra a continuación:

$$\boldsymbol{\Sigma}_k = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{x\theta} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{y\theta} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta\theta} \end{bmatrix} \tag{10}$$

Los valores de la matriz (10) se utilizaron sustituyendo los mismos valores en la matriz que se manda a odometría como se ve en la siguiente matriz:

$$\Sigma = \begin{bmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} & \sigma_{x\varphi} & \sigma_{x\psi} & \sigma_{x\theta} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} & \sigma_{y\varphi} & \sigma_{y\psi} & \sigma_{y\theta} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} & \sigma_{z\varphi} & \sigma_{z\psi} & \sigma_{z\theta} \\ \sigma_{\varphi x} & \sigma_{\varphi y} & \sigma_{\varphi z} & \sigma_{\varphi\varphi} & \sigma_{\varphi\psi} & \sigma_{\varphi\theta} \\ \sigma_{\psi x} & \sigma_{\psi y} & \sigma_{\psi z} & \sigma_{\psi\varphi} & \sigma_{\psi\psi} & \sigma_{\psi\theta} \\ \sigma_{\theta x} & \sigma_{\theta y} & \sigma_{\theta z} & \sigma_{\theta\varphi} & \sigma_{\theta\psi} & \sigma_{\theta\theta} \end{bmatrix} \tag{11}$$

La matriz (10) es la que se manda a odometría que contiene 6 valores (x,y,z,φ,ψ, Θ) , al realizar movimientos solo en los valores de la matriz (10) el resto de valores se manda el valor de 0.

Todo lo previamente explicado es representado en el nodo llamado kinematic_puzzlebot que es el encargado de realizar los cálculos en la función *elipse_covarianza* (apéndice **Funcion_Covarianza**) para posteriormente mandarlos al nodo de localización y poder publicarlos mediante un mensaje de tipo */pose* modificando su atributo *covariance* a la odometría.

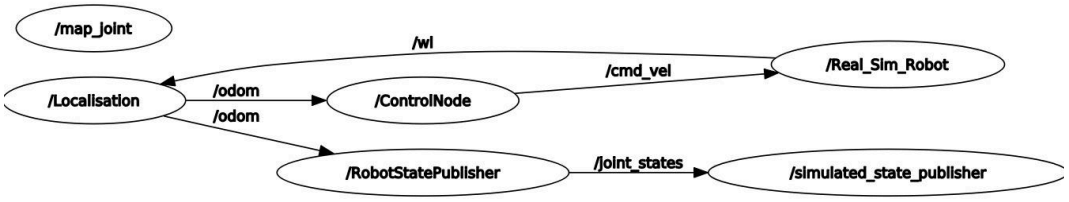

Imagen 1. Gráfico de la arquitectura de ROS del proyecto

Obtención kr y kl:

Para la obtención de estas constantes era necesario la elaboración de una arquitectura de lazo abierto, la arquitectura que se había trabajado durante el desarrollo de los retos del socio formador había sido de lazo cerrado siendo el mensaje */odom* que al conectarse al nodo *ControlNode* el que genera la retroalimentación del sistema de lazo cerrado (Imagen 1). Teniendo esto en mente es que se optó a la creación de una nueva arquitectura únicamente para la obtención de las constantes requeridas, la nueva arquitectura puede apreciarse en la Imagen 2.

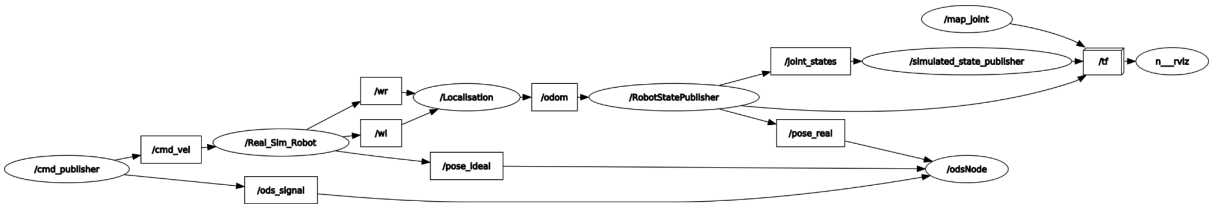

Imagen 2. Gráfico de la arquitectura de ROS de lazo abierto

- CMD Publisher:

Las principales diferencias que se pueden observar son la eliminación del nodo de control que deja de ser requerido debido a la nula necesidad de retroalimentación, sin embargo en la arquitectura original el nodo *ControlNode* era requerido ya que este generaba un mensaje de tipo *Twist* que permite al nodo *Real_Sim_Robot* generar las velocidades angulares de cada una de las llantas, al no contar con este nodo se tuvo que generar uno nuevo encargado de generar el mensaje *Twist*.

Este nodo no solamente se limita a la creación del mensaje, recibe parámetros ingresados por el usuario de velocidad(puede ser lineal o angular) y el tiempo que se quiere mantener esta velocidad, esto es necesario para la parametrización de los errores del robot sin retroalimentación, el nodo maneja un rango de tiempo siguiendo el parámetro ingresado,

mientras el tiempo transcurrido de la simulación sea menor al tiempo designado, el nodo publicará la velocidad ingresada, al pasar el tiempo designado, se publica un alto asignando a la velocidad un valor de 0. La última funcionalidad de este nodo es la publicación de un mensaje Booleano que se envía posteriormente al alto del robot, esta bandera es la condicional que indica que el robot se ha detenido,es en este momento que se tiene que realizar el guardado de la posición para la comparación de posición ideal y posición real (este proceso se desarrolla en la siguiente sección).

- Publicación de posiciones:

Es importante destacar que para la incertidumbre es necesario contar con un modelo ideal (simulación) y un modelo real (simulación con físicas, robot físico) ya que el principal objetivo de obtener la incertidumbre es poder contemplar factores externos al robot que puedan llegar a ser perjudiciales para el robot y su desempeño, sin embargo para la elaboración de este reto se decidió implementar unicamente el modelo simulado en *RVIZ* que se considera un modelo ideal, esto es debido a que actualmente no se cuenta con una simulación en *Gazebo* o el robot en físico para pruebas.

Debido a la anterior es que se utiliza para mediciones la posiciones del robot obtenidas en el nodo del robot simulado *Real_Sim_Robot* que calcula las posiciones mediante el uso de la velocidades lineal y angular (v,w), este se decidió tomar como el modelo ideal mientras que la posición que se tomó como “real” era la posición que se simulaba en *RVIZ*, esta es publicada por el nodo *RobotStatePublisher*, entre ambas posiciones suele haber diferencias, se intuye que es debido al cálculo y la limitación de decimales de los datos *Float32* de las velocidades de las llantas (wl, wr) que podrían generar una diferencia de milésimas, este acercamiento nuevamente se recuerda que no es el adecuado debido a la falta de un modelo real propio pero como aproximación para el desarrollo de los siguientes retos funciona positivamente.

Se mencionó cómo se involucran 2 nodos, estos se encargan de publicar mensajes de tipo *Pose* y *PoseStamped* al nodo *odsNode* (se explican sus funcionalidades en la siguiente sección) siendo estas la posición ideal y la posición "real" respectivamente, la publicacion correspondiente del nodo *Real_Sim_Robot* es en la función *update_Pose* (genera las posiciones y orientaciones a partir de la velocidad lineal y angular) y la función *publish_robot_state* (encargada de publicar todos los mensajes necesarios del Nodo para la arquitectura). La publicación correspondiente del nodo *RobotStatePublisher* se encuentra en la función *publish_joints* (genera las posiciones y velocidades para las juntas, necesarias para la simulación en *RVIZ*), al tener los valores de posicion y orientación es que se podían utilizar nuevamente para generar la posición "real".

- Nodo almacenador de datos:

Se agrega un nuevo nodo a la arquitectura siendo el nodo *ODSNode*, este nodo es el encargado de recibir los mensajes de tipo *Pose* mediante 2 callbacks, sin embargo, espera la señal enviada por el nodo *CMDNode*, esta señal marca cuando el robot se ha detenido para que sea únicamente esta posición final la que se almacene en txt que el nodo se encarga de escribir.

- Captación de datos y procesamiento:

Tras la obtención de las diferencias de las posiciones, estos son registrados en hojas de datos, con las condiciones utilizadas en cada caso, recordar que cada caso tiene como variables la velocidad (lineal o angular) y el tiempo de simulación. A continuación se muestra la tabla (1) que muestra todos los casos de simulación utilizados.

Velocidad Lineal	v	Tiempo
Caso 1	0.5 m/s	2 s
Caso 2	0.5 m/s	5 s
Caso 3	0.5 m/s	10 s
Caso 4	0.7 m/s	2 s

Caso 5	0.3 m/s	2 s
Caso 6	0.3 m/s	10 s
Velocidad Angular	w	Tiempo
Caso 1	0.5 rad/s	3 s
Caso 2	0.5 rad/s	5 s
Caso 3	0.5 rad/s	10 s

Tabla 1. Casos utilizados en el proyecto

A todos los datos se les aplicó medidas de covarianza y de desviación estándar, todavía se sigue definiendo si es que hay diferencia considerable entre ambas operaciones, para el desarrollo de este proyecto se decidió utilizar los resultados de la desviación estándar, al tener un resultado de desviación estándar por cada caso y debido a su similaridad es que se decidió promediar estos valores (promedio de 5 elementos para la velocidad lineal y promedio de 3 elementos para la velocidad angular) que serán los valores de las constantes para la simulación de este entregable.

Resultados obtenidos

En el siguiente video se muestran los resultados del *Puzzlebot* volviendo a utilizar la arquitectura de lazo cerrado (Imagen 1) agregando la incertidumbre utilizando las constantes desarrolladas previamente.

https://youtu.be/LmRsL9l8ph0

Conclusión y análisis de resultados

Como conclusión para este proyecto es que a pesar de que el proceso de obtención de la incertidumbre no fue el adecuado, con las consideraciones actuales del proyecto se pudo obtener un resultado bastante satisfactorio logrando graficar correctamente la elipse de confidencia además de obtener un crecimiento limitado conforme pasa el tiempo mostrando que la incertidumbre es bastante acertada. En este reto se presenta una aproximación adecuada a los siguientes retos en los que si se llegaba a incorporar correctamente un modelo "real" el procedimiento es el mismo.