Tecnológico de Monterrey

Campus Puebla

Materia

Fundamentación de Robótica TE3001B

Tema

Challenge III

Integrantes

José Jezarel Sánchez Mijares A01735226

Frida Lizett Zavala Pérez A01275226

Antonio Silva Martínez A01173663

Fecha

Mayo 25 2023

Contenidos

Resumen

El documento de investigación propone desarrollar las habilidades de los estudiantes en el manejo de ROS (Robot Operating System) y Gazebo, a través de una actividad que involucra la interacción con un robot llamado Puzzlebot. El objetivo de la actividad es aumentar el nivel de interacción y la implementación de funciones disponibles en ROS, así como diseñar un controlador P, PI o PID para minimizar el error de posición del robot. Igualmente se añadió un sistema de navegación a través de visión por computadora implementado opencv en ROS, para detectar señales de tránsito.

Objetivos

1. Desarrollar en el alumno las capacidades de manejo del framework ROS, Gazebo, y con los conocimientos adquiridos en la actividad previa, incrementar el grado de interacción y la implementación de funciones disponibles en ROS.
2. Este reto permitirá al alumno interactuar con el Puzzlebot, conocer en todo momento las coordenadas de la posición del robot, introducir una posición deseada y calcular en tiempo real el error de posición. Además, el alumno podrá diseñar un controlador P, PI o PID para minimizar el error y lograr que el robot llegue a la posición deseada lo más exacto posible a través de ROS.
3. Asimismo, pretende mostrar el comportamiento de los sistemas de visión artificial en la robótica móvil. Particularmente detección de formas y colores a través de una cámara empotrada.

Introducción

Una de las tareas principales en la robótica móvil, es controlar con precisión el movimiento de un robot en un entorno cambiante. El control de retroalimentación es una técnica utilizada para mantener el movimiento del robot dentro de los parámetros deseados y corregir errores de posicionamiento en tiempo real. En este sentido, el controlador proporcional-integral-derivativo (PID) es uno de los métodos más utilizados para implementar el control de robots móviles diferenciales.

El controlador PID se basa en calcular el error entre la posición deseada del robot y su posición real y ajusta los parámetros del controlador para minimizar este error. Sin embargo, al implementar un controlador PID, se deben considerar varios problemas relacionados con el cálculo de errores y la confiabilidad del controlador para garantizar un control preciso y confiable del robot.

Este reto resuelve el problema del control de retroalimentación de un robot móvil con un enfoque en el uso de un controlador PID para un robot móvil diferencial. Se discuten los problemas relacionados con el cálculo del error y la confiabilidad del controlador y se proponen soluciones para resolver estos problemas. El objetivo principal es proporcionar una visión general de los conceptos y métodos clave para controlar robots móviles y explorar los últimos avances en el desarrollo de controladores robustos para mejorar el rendimiento de los robots móviles en diferentes condiciones.

Así mismo se implementó la visión por computadora y el procesamiento de imágenes, lo cual al implementarlo en la tarjeta embebida significa hacer uso de la potencia de cómputo y los recursos disponibles en la tarjeta para realizar las tareas relacionadas con el procesamiento de las imágenes. El objetivo de la visión por computadora en robots como éste puede variar desde tareas sencillas como la captura de imágenes en tiempo real, hasta la detección de objetos, en este caso se usó la detección de bordes para generar la detección de objetos, así como la detección de colores. En cuanto a la robustez en los sistemas de procesamiento de imágenes se refiere a la capacidad de un sistema para funcionar de manera confiable y precisa a pesar de la variación de las condiciones son las que cuente. Un sistema robusto en procesamiento de imágenes es capaz de manejar y adaptarse a diversos problemas, tales como

como ruido en las imágenes, variaciones en la iluminación, cambios en las perspectivas y deformaciones geométricas.

Solución del problema

Partiendo de la solución del reto anterior (Week_2_robótica_móvil) donde se diseñó un controlador PID, se generaron diversas mejoras en el controlador analizando cada una de las variables y los resultados a partir de sus cambios.

Para la parte del procesamiento de imágenes se generó un código por separado que generara el video en tiempo real y a partir de la detección de bordes se implementó una función para detectar círculos, aplicando paŕametros específicos, los cuales fueron definidos a partir de las pruebas realizadas en diversas condiciones de luz y distancias. A partir de ello se hizo el proceso de la detección de colores, donde sí el circulo era verde, rojo u amarillo lo encerrara de dicho color y mostrara el nombre del color en la pantalla. A partir de las respuestas generadas se definieron acciones, tales como seguir, disminuir la velocidad o frenar. Estas acciones se fueron delimitando con más condiciones, para obtener el resultado que se esperaba, esto ya en conjunto con los nodos del controlador y los del recorrido y coordenadas de la trayectoria.

A continuación se describe una bitácora de implementación donde se anotaron los principales desafíos y hallazgos presentados durante el proceso de realización del reto con la finalidad de tenerlos presentes para generar respuestas de solución y proponer mejoras en el diseño de solución.

Problemas presentados:

1. Al ir a puntos que están “atrás” de su posición, es decir, si estamos en 0,0 viendo a x y ponemos -1,0 el robot no es capaz de girar correctamente y pierde su “posición ” por lo que provoca que el tópico “/pose” no mande los datos correctamente, aunque esto no pasa con el ángulo, si solo ponemos el ángulo y dejamos de mandar velocidad al cont_vl =0 lo hace correctamente entonces, ¿qué ocasiona el problema?. Quizá la velocidad con la que se mueve cont_vl hace que el cont_ang no le de tiempo de reaccionar y también por eso se observa como un sobretiro.

Posibles soluciones:

- Podemos tratar de reducir la kp del cont_vl que como detecta distancias grandes al inicio da un impulso muy rápido, sin embargo esto puede hacer que el sistema sea demasiado lento, entonces integrar las variables ki, kd puede ayudar a que el "rebote" no sea demasiado grande y el sistema no sea demasiado lento.
- Seguir aumentando las constantes PID del cont_ang así en teoría al ser más rápido podríamos reducir el sobretiro que genera al inicio que es generada por el cont_vl sin, y no podemos hacer que primero empiece con un control y después con otro, deben ser al mismo tiempo.

  Constantes Utilizadas:

  Prueba 1:

  kp_l = 0.9

  ki_l = 0.00

  kd_l = 0.0

  kp_ang = 1.5

  ki_ang = 0.03

  kd_ang = 0.3

  R: (Demasiado alto las ki_ang, no reacciona el controlador)

  Prueba 2:

  kp_l = 0.9

  ki_l = 0.00

  kd_l = 0.0

  kp_ang = 1.5

  ki_ang = 0.01

  kd_ang = 0.0

  R: (mismo error, pero no se si es debido a la constante ki)

  Prueba 3:

  kp_l = 0.25

  ki_l = 0.0

  kd_l = 0.0

  kp_ang = 1.5

  ki_ang = 0.01

  kd_ang = 0.0

R: (error diferente, se acerca un poco más, se puede observar un pequeño "sobretiro" lo cual se entiende por la naturaleza de los controladores, bajar la kp_l funcionó, ¿qué tan lento hará el sistema si seguimos bajando?)

Prueba 4:

kp_l = 0.10

ki_l = 0.0

kd_l = 0.0

kp_ang = 1.5

ki_ang = 0.01

kd_ang = 0.0

R: Mejor resultado obtenido, sin embargo sacrificamos un poco la velocidad del sistema, ¿cómo podemos hacer que sea más rápido sin sacrificar su precisión?

- Se cambió la función wrap_to pi al ángulo de salida del error como el de la posición angular lo que provocó que al momento de ejecutar el siguiente punto no se alterara. (**funcionó**)

2. El robot interactúa con el semáforo dependiendo del color
3. El robot no ejecuta el código como debería ya que al llegar al segundo punto empieza a mandar erróneamente los valores y empieza a moverse sin ningún sentido.

Posibles soluciones:

Lo que parece causar el problema es la dt ya que llega un momento en el que se pierde completamente, pero ¿esto será problema de la jetson? es decir pierde la señal, sin embargo creemos que es más por el hecho de que no está corriendo todo el paquete de ROS como tal y solo una parte, al intentar correr únicamente con la jetson surge el mismo problema pero esto puede ser ocasionado por el archivo de launch donde al momento de desplegarlo en la terminal terminal, se muestra un error ya que no puede desplegar otras terminales dentro del ssh, como no creen los params.yaml entonces el robot pierde la dirección.

Actualización:
Si se debe a que no son parte del mismo paquete y aunque si están conectados por los tópicos ya cuando piden los parámetros o los archivos de mensajes estos no logran reaccionar, se intentará probar con el puzzlebot directo.

Actualización:
A pesar de que se probó en la jetson surgió el mismo problema, esto debido a la dt sin embargo no se sabe porqué sucede esto, es por eso que solo se puso que el dt fuera de 0.1 (lo que es de acuerdo al rospy) pero se declaró como constante para que no fallará por este error.

Resultados:

Enlace al vídeo de la simulación y demostración:
https://youtu.be/BNkWC-XNIvc

El robot sigue la trayectoria, mientras que reacciona a las señales de semáforo que se muestran en la cámara. Al hacer los giros la velocidad va disminuyendo para aumentar la precisión del ángulo y evitar sobretiros.

Conclusiones

Con el proceso de realización del reto se integraron los diversos conocimientos aprendidos a lo largo del curso que en principio se manejaron de manera aislada pero finalmente se combinaron para generar un proyecto en común, lo que nos ayudó a ver de manera tangible y práctica la implementación de cada uno de estos componentes trabajando en conjunto para lograr un objetivo en común. La robótica al ser un área multidisciplinaria permite acciones como éstas.
Con este reto en específico se logró un robot con un sistema de navegación funcional y usable, ya que el movimiento que se genera a partir del posicionamiento y el control implementado resulta satisfactorio, a partir de los resultados del reto anterior se hicieron mejoras en el control que se diseñó, generando análisis más profundos de los resultados de cada prueba, mientras que el procesamiento de imágen implementado realiza la detección

tanto de los colores como de las figuras específicas, esto se logró después de generar varias pruebas ya que las diferentes condiciones del entorno, por ejemplo de luz, cambiaban los resultados, igualmente se tomaron otros factores, como decisiones que tomaría el robot en cuanto a su distancia, etc.

Consideramos que la solución de este reto sirve como una buena base para generar en un futuro un sistema de visión y navegación mucho más robusto al implementar más funciones de detección y procesamiento de imágenes, siempre y cuando se adapte dicho código a las nuevas condiciones y especificaciones. Un área de oportunidad detectada es un error que recibimos a partir de la dt del sistema, y es que en determinado momento el robot pierde la conexión y no recibe la señal adecuada lo cuál se solucionó definiendo la dt como una constante.

Referencias

ManchesterRoboticsLtd. (2023). GitHub - ManchesterRoboticsLtd/TE3002B_Intelligent_Robotics_Implementation. Retrieved 26 April 2023, from https://github.com/ManchesterRoboticsLtd/TE3002B_Intelligent_Robotics_Implementation

Sánchez, J. J., Silva, A., & Zavala, F. L. (2023, April 25). *Reto semanal II*. Retrieved May 11, 2023, from https://docs.google.com/document/d/1nvoLrCuhpt6HTk-QKYXddcWD2YCVugQ2mxF_XolIlEM/edit#