Turtlebot 3 - Autonomous Driving

Universidad de Ingeniería y Tecnología UTEC - Curso: Deep Learning

1st Romero Diaz, Gabriel
201810423
Computer Science
Lima, Perú
gabriel.romero@utec.edu.pe

2nd Riveros Soto, Andres
201810017
Computer Science
Lima, Perú
andres.riveros@utec.edu.pe

3nd Vásquez Auqui, Fabrizio
201810205
Computer Science
Lima, Perú
fabrizio.vasquez@utec.edu.pe

Abstract*—Este trabajo explora la integración de un modelo lenguaje-visión (VLM), específicamente ChatGPT, con datos proporcionados por un sensor LIDAR para mejorar las capacidades de navegación autónoma del robot TurtleBot3. La combinación de información visual obtenida a través de una cámara RGB-D con datos espaciales del LIDAR permite al robot comprender mejor su entorno y tomar decisiones de navegación más informadas. Se describe la metodología de integración, incluyendo la configuración del entorno con ROS Noetic y la conexión a la API de ChatGPT. Se presenta un esquema de prompting diseñado para que el VLM genere comandos de movimiento en formato JSON (avanzar, girar a la izquierda o a la derecha), considerando las distancias proporcionadas por el LIDAR en diferentes ángulos y la necesidad de evitar colisiones. Se realizaron experimentos en escenarios de prueba que simulan un circuito con obstáculos, analizando la respuesta del robot ante diferentes configuraciones del entorno. Los resultados demuestran el potencial de esta integración para la navegación autónoma, aunque se observa un tiempo de respuesta promedio de la API de 8 segundos. Se discuten los resultados obtenidos y se proponen futuras líneas de investigación, incluyendo la incorporación de aprendizaje por refuerzo, la extensión del sistema a entornos dinámicos, la optimización de la integración para reducir latencias, la evaluación de arquitecturas VLM alternativas y la implementación de mapeo 3D en tiempo real.

I. Introducción

El TurtleBot3, una plataforma robótica de código abierto, está equipado con sensores LIDAR y cámaras que pueden ser integrados con modelos de lenguaje-visión (Vision Language Models, VLM) como ChatGPT. Esta combinación permite al robot comprender el entorno y tomar decisiones inteligentes basadas en datos visuales y espaciales. Este trabajo explora el uso de un modelo VLM junto con datos LIDAR para navegar de forma autónoma.

La estructura del documento es como sigue: en la Sección II, se introduce el marco teórico. En la Sección III, se describe la metodología de integración del modelo de lenguaje-visión con los datos LIDAR. En la Sección IV, se presentan los resultados experimentales y análisis. Finalmente, en la Sección V, se ofrecen conclusiones y posibles extensiones futuras.

II. Marco Teórico

A. Modelos de Lenguaje-Visión

Los modelos de lenguaje-visión son sistemas que pueden procesar y razonar sobre información visual y textual de manera combinada. En este trabajo, ChatGPT es utilizado para interpretar datos visuales provenientes de una cámara RGB-D y proporcionar comandos de navegación en tiempo real.

B. LIDAR en Robótica

El sensor LIDAR genera datos tridimensionales del entorno utilizando rayos láser. Esta información es crucial para mapear el entorno, detectar obstáculos y calcular trayectorias seguras. En este trabajo, los datos LIDAR se combinan con la salida del modelo VLM para mejorar la toma de decisiones.

III. Trabajos Relacionados

La navegación autónoma y el control de robots móviles son áreas de investigación activas que han sido exploradas con diversas metodologías y algoritmos. El TurtleBot3, una plataforma de bajo costo y programable basada en ROS, ha sido ampliamente utilizado en investigaciones relacionadas con la navegación autónoma, la detección de obstáculos y el aprendizaje por refuerzo.

Marchesini y Farinelli (2021) propusieron un enfoque basado en Double Deep Q-Networks (DDQN) para resolver el problema de navegación sin mapa en TurtleBot3. Su método reduce significativamente el tiempo de entrenamiento en comparación con algoritmos de espacio continuo como PPO y DDPG. Además, demostraron la capacidad del modelo entrenado en simuladores Unity para ser transferido al TurtleBot3 real sin necesidad de entrenamiento adicional [4].

En otro trabajo, Aydemir et al. (2021) evaluaron algoritmos de Deep Q-Learning (DQN) y Deep DQN para la planificación de rutas locales en robots móviles. Los resultados mostraron que estos métodos permiten evitar obstáculos de manera eficiente en entornos dinámicos, superando algunos de los desafíos de los enfoques tradicionales, como A* y RRT, que requieren replanificación constante en sistemas offline [3].

Dobriborsci y Osinenko (2021) realizaron un estudio experimental comparando métodos de aprendizaje por refuerzo predictivo con control predictivo de modelos (MPC) en TurtleBot3. Descubrieron que los métodos de aprendizaje por refuerzo, como el Q-Learning apilado, superaron a MPC en términos de costo acumulado, destacando el potencial del aprendizaje por refuerzo para mejorar el rendimiento mientras mantienen las propiedades de MPC [2].

Finalmente, Patil y Gunale (2021) desarrollaron algoritmos de procesamiento de imágenes y aprendizaje automático para conducción autónoma en TurtleBot3. Su trabajo incluyó detección de carriles, detección de señales de tráfico y evasión de obstáculos, probados inicialmente en simuladores Gazebo y luego en el robot físico [1]. Esto subraya el potencial del TurtleBot3 como una plataforma versátil para pruebas de algoritmos autónomos.

IV. Metodología

A. Configuración del Entorno

El sistema fue configurado con Ubuntu 20.04, ROS Noetic y un TurtleBot3 Burger. Se integraron sensores LIDAR y cámaras RGB-D para recolectar datos. El modelo ChatGPT se conectó al sistema mediante una API personalizada para interpretar datos visuales y textuales.

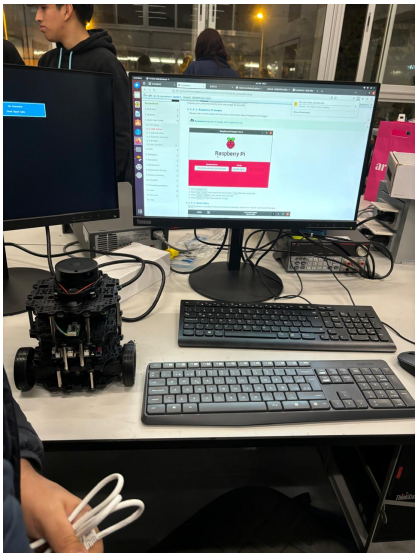

Fig. 1: Instalando Sistema Operativo ROS Noetic

B. Experimentos

En esta sección se mostrará los experimentos que se realizaron usando la API de OpenAI (GPT-4o mini) y los datos de las ditancias obtenidas por el Lidar en tres ángulos que son 0 grados, 60 grados y 300 grados.

1) Prompt: Se te va a compartir una foto que es sobre una pista de carrera en la que participas y eres un Turtlebot 3, donde las cajas de derecha y izquierda forman un camino o circuito por donde debes recorrer. Además, tienes que moverte por el camino entre las cajas de cartón y evadir obstáculos como otros Turtlebot3 o cajas en el medio del camino.

De la imagen compartida el LIDAR nos da la siguiente información: para un ángulo de 0 grados la distancia es {distance_0:.2f} m, para un ángulo de 60 grados la distancia es {(distance_60):.2f} m, para un ángulo de 300 grados la distancia es {distance_300:.2f} m. Es importante saber que no podemos recorrer toda la distancia que nos da el LIDAR porque si no chocaríamos con una pared de cartón.

Necesito que tomes una decisión sobre qué movimiento realizar a partir de la imagen y las distancias de los ángulos: avanzar hacia delante, girar a la derecha o girar a la izquierda. Además, la distancia que debo tomar para no chocar y la duración de la acción. Es importante saber que la velocidad del Turtlebot es de 0.22 m/s y la velocidad de giro es de 0.5 rad/s.

Adicionalmente, si giro a la derecha o izquierda no debo moverme en distancia, solo en ángulo, y la variable duración debe ser 1.

Por favor, devuelve solo el JSON con la acción a realizar. El formato debe ser el siguiente, y sólo eso:

Es importante que avancemos hacia la dirección que tiene mayor distancia, y si la distancia de izquierda o derecha es menor a 0.3, no gires. Si la distancia de izquierda y derecha es mayor a la distancia de 0 grados, gira hacia la dirección de mayor distancia entre derecha e izquierda.

Formato del JSON:

```
{
  "actions": [
    {"movement_type": "FORWARD", "distance":
    <num>, "duration": <num>},
    {"movement_type": "TURN_LEFT", "angle":
    <num>, "duration": <num>},
    {"movement_type": "TURN_RIGHT", "angle":
    <num>, "duration": <num>},
  ]
}
```

Nada más que este JSON debe contener solo una sola acción.

Fig. 2: El resultado fue FORWARD, con una distancia de avance de 1.1 metros y duración de 6 seg. Los datos provenientes del lidar fueron distancia_0 (1.35 m), distance_60 (1.1 m) y distance_300 (1.21 m).

Listing 1: Integración de ChatGPT con LIDAR

```
import openai
import rospy
from sensor_msgs.msg import LaserScan
from cv2 import imread

def process_lidar(data):
    lidar_ranges = [r for r in data.ranges if r < float('inf')]
```

Fig. 3: El resultado fue FORWARD, con una distancia de avance de 2.1 metros y duración de 12 seg. Los datos provenientes del lidar fueron distancia_0 (3.1 m), distance_60 (0.9 m) y distance_300 (1.2 m).

Fig. 4: El resultado fue FORWARD, con una distancia de avance de 1.5 metros y duración de 8 seg. Los datos provenientes del lidar fueron distancia_0 (2.1 m), distance_60 (1.5 m) y distance_300 (1.2 m).

Fig. 5: El resultado fue TURN RIGHT, con una distancia de avance de 0.2 metros y duración de 1.5 seg. Los datos provenientes del lidar fueron distancia_0 (2.7 m), distance_60 (0.45 m) y distance_300 (0.15 m).

```
    return "LIDAR data processed: {} obstacles detected.".format(
        len(lidar_ranges)
    )

def send_to_chatgpt(image_path, lidar_data):
    openai.api_key = "YOUR_API_KEY"
    response = openai.ChatCompletion.create(
        model="gpt-4",
        messages=[
            {"role": "system", "content":
             "You are a navigation assistant."},
            {"role": "user", "content":
             f"Image: {image_path}, LIDAR: {lidar_data}"}
        ]
    )
    return response["choices"][0]["message"]["content"]

if __name__ == "__main__":
    rospy.init_node('turtlebot3_vlm_lidar', anonymous=True)
    rospy.Subscriber('/scan', LaserScan, process_lidar)
    image = imread('path/to/image.png')
    lidar_info = process_lidar(LaserScan())
    decision = send_to_chatgpt('path/to/image.png', lidar_info)
    print("Navigation Command: ", decision)
```

C. Modelo de Decisión

El modelo VLM interpreta las entradas visuales y LIDAR para generar comandos de navegación. La fusión de datos se realiza mediante la siguiente ecuación:

$$C = \alpha V + \beta L, \tag{1}$$

donde C es el comando final, V es la salida del modelo visual, L es la información LIDAR, y α, β son factores de ponderación. Estos factores se calibran experimentalmente para optimizar el rendimiento.

D. Detección y Evitación de Obstáculos

La información del sensor LIDAR se analiza para identificar regiones libres de obstáculos. El robot calcula un vector direccional basado en:

$$\mathbf{d} = \frac{1}{n} \sum_{i=1}^{n} \mathbf{r}_i, \tag{2}$$

donde $\mathbf{r}_i$ representa los vectores libres detectados por el LIDAR. Este vector direccional se combina con la salida del modelo de lenguaje para ajustar la trayectoria.

V. Resultados

Las figuras 2 y 3 ilustran escenarios con espacios abiertos que permiten el avance del Turtlebot. Como se observa, en ambos casos la decisión fue avanzar hacia adelante, lo cual resulta predecible. Sin embargo, la distancia de avance es menor a la proporcionada por el LiDAR, lo que evita posibles colisiones con las paredes de cartón.

En contraste, las figuras 4 y 5 representan situaciones más desafiantes. En la figura 4, se identifica una estructura similar a un marco de puerta frente al robot. Ante esta configuración, el modelo decide avanzar, una acción que, si bien no es incorrecta, implica un riesgo potencial de colisión con las paredes de cartón. Por otro lado, en la figura 5, el modelo opta por girar a la derecha, probablemente debido a la proximidad a la pared izquierda.

VI. Conclusiones y Trabajos Futuros

La combinación de un modelo lenguaje-visión con datos LiDAR demostró mejorar significativamente las capacidades de navegación autónoma del TurtleBot3, permitiéndole completar exitosamente el circuito de prueba, a pesar de un tiempo de respuesta promedio de la API de 8 segundos. Este resultado valida el potencial de la integración de modalidades para la navegación robótica.

Como trabajos futuros, se proponen las siguientes líneas de investigación:

- **Aprendizaje por Refuerzo:** Implementar aprendizaje por refuerzo utilizando la salida combinada del modelo lenguaje-visión y datos LiDAR para optimizar la toma de decisiones y la adaptación a entornos complejos.
- **Entornos Dinámicos:** Extender el sistema para operar en entornos dinámicos que incluyan la presencia de personas y vehículos, abordando los desafíos de predicción de movimiento y evitación de colisiones en tiempo real.
- **Optimización de la Integración:** Optimizar la integración del sistema para reducir las latencias y lograr un rendimiento en tiempo real, lo que permitirá una navegación más fluida y reactiva.
- **Arquitecturas VLM Alternativas:** Evaluar diferentes arquitecturas de modelos VLM (Vision-Language Models) con el objetivo de mejorar el tiempo de inferencia y la eficiencia computacional.
- **Mapeo 3D en Tiempo Real:** Implementar la generación de mapas 3D en tiempo real para proporcionar al robot una representación más completa del entorno, lo que permitirá una navegación más precisa y robusta.

VII. Repositorio

Incluimos el enlace al entorno en donde ejecutamos la experimentación: https://github.com/Andres20-Utec/TurtleBot3_VLM.

VIII. Referencias

Referencias Bibliográficas

[1] Patil, S.G., & Gunale, K.G. (2022). Implementation of Autonomous Driving Algorithms on a Miniature Robot. International Journal of Electrical and Electronic Engineering & Telecommunications. [Online]. Available: https://www.ijeetc.com/index.php?m=content&c=index&a=show&catid=222&id=1601

[2] D. Dobriborsci and P. Osinenko, "An experimental study of two predictive reinforcement learning methods and comparison with model-predictive control," arXiv preprint arXiv:2108.04857, 2021. [Online]. Available: https://arxiv.org/abs/2108.04857.

[3] M. Gok, M. Tekerek, and H. Aydemir, "Reinforcement learning based local path planning for mobile robot," arXiv preprint arXiv:2403.12463, 2023. [Online]. Available: https://arxiv.org/abs/2403.12463.

[4] E. Marchesini and A. Farinelli, "Discrete Deep Reinforcement Learning for Mapless Navigation," 2020 IEEE International Conference on Robotics and Automation (ICRA), Paris, France, 2020, pp. 10688-10694, doi: 10.1109/ICRA40945.2020.9196739.