Tecnológico de Monterrey

RETO: AUTONOMOUS NAVIGATION

Programación

Resumen del reto

Este documento contiene un breve resumen de lo realizado en el reto.

Daniela Pamelín Álvarez Guarneros
A01026164@tec.mx

Reto Quantum

A lo largo de la historia, la humanidad ha buscado entender y proteger la existencia de la vida, lo que ha motivado la exploración y estudio del espacio exterior. No obstante, los viajes al espacio son costosos y requieren mucho tiempo debido a su complejidad. Según la NASA (2021), las misiones espaciales se centran en dos prioridades: la búsqueda de vida extraterrestre y la identificación de recursos que puedan preservar la vida humana. En los últimos años, Marte ha captado un creciente interés debido a su proximidad y que además es el cuerpo planetario más similar a la Tierra. Esto implica un alto potencial para haber albergado o albergar vida, convirtiendo a Marte en un objetivo clave en el campo de la astrobiología (Centro de Astrobiología, s.f).

Desde las pioneras misiones espaciales realizadas en la mitad del siglo XX, se ha consolidado una comprensión inequívoca de que la exploración de Marte implica una serie de desafíos tecnológicos sumamente complejos y variados. Es por esta razón que la comunidad científica y los ingenieros han forjado una vía eficaz para desentrañar los misterios de la superficie marciana: la utilización de ingenios robóticos que funcionan como auténticos laboratorios itinerantes. Esta innovadora perspectiva resalta la imperante necesidad de sumergirse en el desarrollo de agentes inteligentes capaces de sortear las complejidades que presenta ese planeta. La construcción y mejora de estos agentes constituyen un eje fundamental en la búsqueda de respuestas en Marte, y representa un hito trascendental en la búsqueda del conocimiento en el vasto cosmos.

El uso de rovers, o vehículos robóticos y la exploración espacial ha cobrado una importancia significativa en las últimas décadas. Estos rovers son máquinas diseñadas para operar en entornos hostiles o desconocidos, como la superficie de otros planetas, y son fundamentales para la investigación y exploración espacial. En específico University Rover Challenge es una de las competencias universitarias más destacadas en el uso de rovers. Se lleva a cabo anualmente y desafía a equipos de estudiantes universitarios de todo el mundo a diseñar, construir y operar rovers que puedan llevar a cabo tareas específicas en un entorno simulado similar a Marte. Estos rovers deben ser capaces de navegar terrenos accidentados, recoger muestras, realizar tareas de manipulación y comunicarse de manera efectiva con la estación base. (University Rover Challenge, s.f.)

En este contexto, la programación desempeña un papel de suma importancia en la construcción y operación de rovers, ya sea en competencias universitarias o en misiones de exploración espacial reales. Su relevancia radica en habilitar el control autónomo de estos vehículos, permitiéndoles tomar decisiones en tiempo real, evitar obstáculos, planificar rutas, y adaptarse a condiciones cambiantes en el entorno. Además, la programación es esencial para la navegación eficiente y segura de los rovers en terrenos desconocidos, así como para llevar a cabo tareas de manipulación, como la recolección y análisis de muestras. Facilita la comunicación entre el rover y la estación base, lo que es esencial para la transmisión de datos científicos y comandos, y también permite la visión por computadora, lo que habilita la identificación de objetos y características importantes en el entorno. En conjunto, la programación es un componente crítico en el éxito de las misiones de rovers y su capacidad para abordar los desafíos de la exploración espacial en entornos remotos y desconocidos.

El presente reporte analiza en detalle tres códigos que simulan la navegación autónoma de un rover en un plano bidimensional. Estos códigos emplean la biblioteca Turtle de Python para la visualización gráfica y, en el caso del tercer código, el framework simpleai para implementar algoritmos de búsqueda

avanzados. El objetivo principal en todos los códigos es permitir que el rover alcance una ubicación objetivo "ArUco" mientras evita obstáculos en su camino. A continuación, se proporciona un análisis pormenorizado de cada uno de los códigos.

Código 1: Reto_basico.py

Configuración Inicial

1. Este código comienza estableciendo los límites del plano de navegación en las variables `X_min`, `X_max`, `Y_min` y `Y_max`
2. Se genera una ubicación aleatoria para el "ArUco" utilizando `random.uniform`.
3. Se crea una ventana gráfica de Turtle con un tamaño de 800x600 píxeles
4. Se carga un fondo de plano ("superficie.png") en la ventana.
5. Se crea un objeto Turtle llamado "rover" que representa al vehículo de exploración. Su velocidad se configura en 50.

Funciones de Movimiento y Estado

1. Se definen dos funciones: `move_rover(x, y, theta)` para mover el rover a una ubicación y orientación específicas, y `print_rover_state(x, y, theta)` para mostrar las coordenadas y orientación actuales del rover.
2. Se marca el punto de inicio del rover en azul y se imprime su estado inicial.
3. Se utiliza Turtle para representar visualmente la ubicación del "ArUco" en rosa.
4. Se implementa un algoritmo simple de búsqueda para que el rover se desplace hacia el "ArUco". A medida que se mueve, se actualizan las coordenadas y se registran en la ruta.
5. El recorrido del rover se visualiza con puntos verdes en el plano.
6. La ventana de Turtle se cierra al hacer clic en ella.

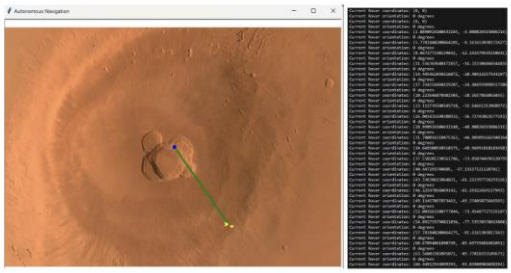

Figura1. Resultados de funcionamiento

Código 2: Reto_obstaculos.py

Este código extiende el código anterior al introducir obstáculos en el plano de navegación., la diferencia es que el rover debe evitar colisionar con estos obstáculos mientras se dirige al "ArUco".

Implementación de Obstáculos

1. Los obstáculos se representan como límites o áreas que el rover no puede atravesar.
2. Se implementan restricciones en las acciones del rover para garantizar que no supere los límites del entorno ni atraviese obstáculos.
3. La generación de la ubicación del "ArUco" y las funciones de movimiento y estado son las mismas que en el código anterior.
4. Se utiliza el mismo algoritmo de búsqueda que en el código anterior para encontrar el "ArUco".
5. El algoritmo se adapta para tener en cuenta los obstáculos y garantizar que el rover evite colisionar con ellos.
6. El recorrido del rover y la visualización en pantalla son similares al código anterior.

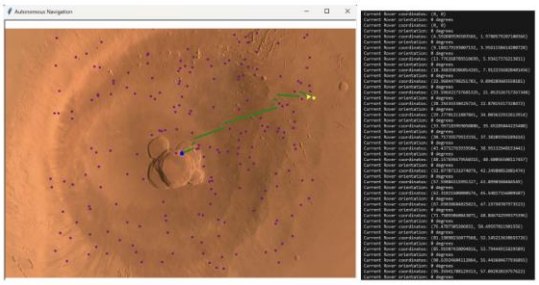

Figura2. Resultados de funcionamiento

Código 3: Reto_AI.py

Este código utiliza el framework simpleai para implementar un algoritmo de búsqueda más sofisticado y optimizado.

1. Se define una clase llamada "Exploreterrain" que hereda de "SearchProblem" de simpleai.
2. Esta clase modela el problema de búsqueda con un estado inicial, un estado objetivo y las acciones permitidas.
3. El estado inicial y el objetivo se establecen según las posiciones del rover y el "ArUco" en el plano.
4. Se toma en cuenta la existencia de obstáculos y se aplican restricciones a las acciones.
5. Se definen costos para las acciones del rover, lo que permite al algoritmo evaluar la eficiencia de las rutas.
6. Se implementa una heurística que estima la distancia al objetivo. Esta heurística es fundamental para el funcionamiento del algoritmo de búsqueda A*.
7. El código ejecuta el algoritmo de búsqueda, en este caso, se utiliza "depth_first" para la demostración.
8. Se encuentra una ruta desde el punto de inicio hasta el "ArUco" mientras se evitan obstáculos.
9. El código imprime la ruta encontrada y muestra visualmente el recorrido del rover en la pantalla.
10. Utiliza colores para resaltar el punto de inicio (azul), el "ArUco" (verde) y la ruta seguida (rojo).
11. La ventana gráfica es interactiva y permite cerrarla al hacer clic.
12. El código calcula el costo total de la ruta como la suma de los valores numéricos en las coordenadas visitadas.

Figura3. Resultados de funcionamiento

Conclusión

Estos tres códigos proporcionan una visión detallada de la navegación autónoma del rover en un plano bidimensional. Desde la navegación básica hasta la consideración de obstáculos y el uso de un algoritmo de búsqueda avanzado, representando así diferentes niveles de complejidad y capacidades. La elección de cuál utilizar depende de los requisitos específicos del proyecto y la complejidad de la tarea de navegación autónoma. Cada uno de estos códigos puede servir como punto de partida para el desarrollo de aplicaciones más avanzadas en la exploración autónoma de rovers.

Referencias

- University Rover Challenge (s.f.). The Mars Society. Recuperado de: https://urc.marssociety.org/home

- Centro de Astrobiología (s.f) ¿Por qué Marte?. Recuperado de: http://cab.inta-csic.es/rems/es/curiosidad/por-que-marte/

- NASA (2021) ¿Porque nos importa si en Marte tiene agua ? Recuperado de: https://spaceplace.nasa.gov/water-on-mars/sp/

- Universidad EIA (2020) Todo lo que debes saber sobre los agentes inteligentes. Recuperado de: https://www.eia.edu.co/wp-content/uploads/2020/09/agentes-inteligentes.pdf