Universidad de Panamá
Facultad de Informática,Electrónica y Comunicación
Laboratorio de Inteligencia Artificial

Agentes que resuelven problemas

Roderick Aparicio 8-916-593
Isaac Bethancourt 8-917-2263
Alan Castro 8-912-1890
Jesús De Gracia 8-1086-1646
Guillermo Espino 8-925-2235
Daniel González 8-907-1404

Allan Marín EC-20-12127
Elvin Marín EC-20-12126
David Morán 2-741-87
Lourdes Moreno 8-920-640
Kaiser Obaldía 8-898-703
Yeny Ortega 8-923-1263

Fermín Povaz 8-932-1661
Kevin Ruedas 8-910-1800
Efraín Samudio 8-772-1697
Krisbel Sanjur 2-738-644
Alexander Santana 8-933-1167

Stephanie Tejeira 8-924-2239
Víctor Valenzuela 8-931-2246
Ana Villarreal 8-917-2049

***Abstract*—**The following article presents the project: Agents who solve problems, it is based on an agent who has the ability to reach a location (Twitter), using the A * search algorithm to choose the fastest route.

RESUMEN

En el siguiente artículo se presenta el proyecto: ***Agentes que resuelven problemas,*** el mismo se basa en un agente que tiene la capacidad de llegar a una ubicación (Twitter), utilizando el algoritmo de búsqueda A* para elegir la ruta más rápida.

PALABRAS CLAVES

Python, Unity, Algoritmo de búsqueda A*, Agente, Simulador.

I - INTRODUCCIÓN

El entorno de simulación LGSVL proporcionado por LG nos da la capacidad de interactuar en un ambiente virtual con agentes como vehículos donde podemos implementar nuestro propio sistema de manejo por medio de algoritmos de para el funcionamiento de inteligencia artificial.

II - MARCO TEÓRICO

Ilustración 1 Logo de LGSVL Simulator

¿Qué es LGSVL Simulator?

LGSVL Simulator es un software para la simulación de entornos que facilita la prueba y desarrollo de sistemas de software de conducción autónomo.

Permite a los desarrolladores simular miles de millones de millas y escenarios arbitrarios para acelerar el desarrollo de algoritmos y la integración del sistema. El simulador está en desarrollo activo en Github y cuenta con la integración de plataformas como Apollo y Autoware.

Especificaciones para el uso de LGSVL:

- CPU Quad core de 4 GHz
- Nvidia GTX 1080 (8GBde memoría)
- Windows 10 64 Bit

Especificaciones a la hora de la implementación:

- Intel Core i5-9400F(6) 2,90 GHz - 4,10 GHz (TurboBoost)
- Tarjeta gráfica Nvidia gtx 1660 - 6GB
- 32GB RAM DDR4
- Ubuntu - linux 18.04LTS

*Algoritmo de búsqueda A**

A * es un algoritmo de búsqueda informado , o una búsqueda de primer orden , lo que significa que está formulado en términos de gráficos ponderados : a partir de un nodo inicial específico de un gráfico, su objetivo es encontrar una ruta al nodo objetivo dado que tenga el más pequeño costo (menor distancia recorrida, menor tiempo, etc.). Lo hace manteniendo un árbol de rutas que se originan en el nodo de inicio y extendiendo esas rutas un borde a la vez hasta que se cumpla su criterio de terminación.

Ilustración 2. Algoritmo utilizado en este laboratorio

Filtro de Kalman

El filtro de Kalman es un algoritmo desarrollado por Rudolf E. Kalman en 1960 que sirve para poder identificar el estado oculto de un sistema dinámico lineal, al igual que el observador de Luenberger, pero sirve además cuando el sistema está sometido a ruido blanco aditivo.

III - METODOLOGÍA

INSTALACIÓN DE LGSVL

Debemos tener la última versión del Simulador LGSVL para nuestro sistema operativo (Windows o Linux) aquí: *https://github.com/lgsvl/simulator/releases/latest y* verificar la instalación del último controlador de NVIDIA:

a. En Windows: Descargar controladores por medio NVIDIA GeForce Experience o desde el sitio web de NVIDIA.
b. En Ubuntu: En Configuración deberemos añadir los controladores de NVIDIA a la lista y en una terminal instalar libvulkan con sudo apt install libvulkan1.

Luego descargamos el .zip con los archivos del simulador y ejecutarlo.

EJECUCIÓN

Utilizamos el Python Api del Lgsvl. Pudimos utilizar las funcionalidades, como sensores, cambios de coordenadas, lectura de choques, iluminación y NPCs. La parte de del spawn fue más sencilla ya que al correr el simulador nos generaba una ruta específica inicial, que fue la utilizada para empezar las pruebas de la simulación. Para la transformación de coordenadas a puntos xy se podía utilizar Pandas y PyProj que generan posiciones por coordenadas. Que recorta el tiempo de análisis y transformación. Decidimos utilizar una variante del algoritmo para el seguimiento de objetivos, en donde decidimos manejar la posición como objetivo. Si lo alcanza genera una nueva posición objetivo y así sucesivamente.

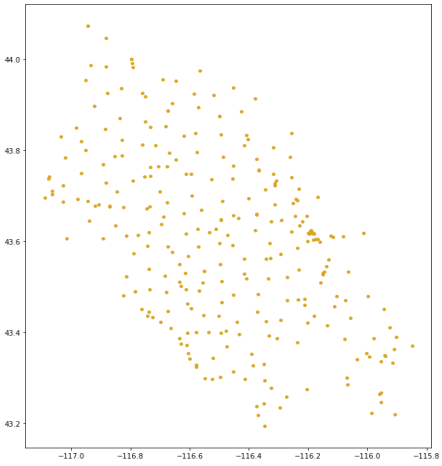

Ilustración 3.Referencia transformación de coordenadas con Pandas y Pyproj.

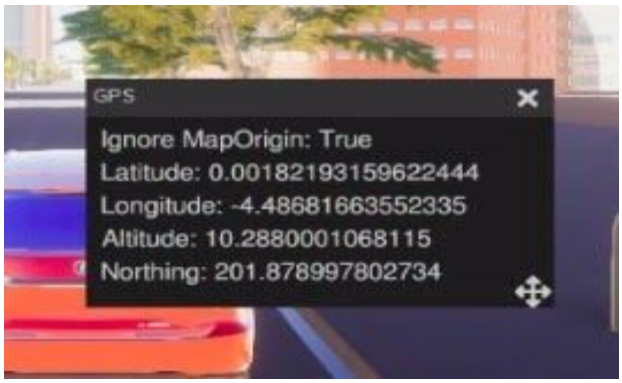

Ilustración 4 . Posición de Spawn (fija)

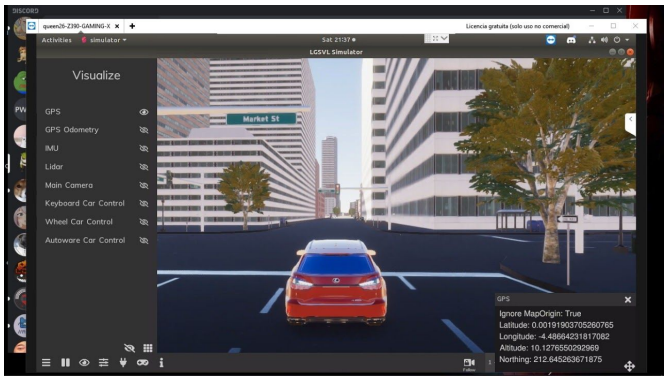

Ilustración 5. Ubicación más cercana a twitter (real).

Utilizamos Unity para recrear los asset y poder tratar de cambiar las posiciones, al hacer una evaluación

Para la ejecución de este proyecto, nos apoyamos con el uso de una API (código adjunto en el repositorio con comentarios de implementación)

Nota: Estuvimos buscando la ubicación basada en el simulador de twitter. Esta coordenada propia del LGSVL no era compatible con la posición real de twitter. Decidimos Hacer la conversión de coordenadas y nos salió un lugar aproximado del punto real, por lo que fue la opción utilizada como ruta final.

```
random.seed(0)
# Inicio del Simulador mediante localhost (lgsvl)
sim = lgsvl.Simulator(os.environ.get("SIMULATOR_HOST", "127.0.0.1"), 8181)
#Seleccion del mapa de scene
if sim.current_scene == "SanFrancisco":
  sim.reset()
else:
  sim.load("SanFrancisco")

spawns = sim.get_spawn()
#se inicia el estado inicial (spawn)
state = lgsvl.AgentState()
state.transform = spawns[0]
a = sim.add_agent("Lincoln2017MKZ (Apollo 5.0)", lgsvl.AgentType.EGO, state)
forward = lgsvl.utils.transform_to_forward(spawns[0])
```

Ilustración 6. Vista miniatura del código recolectando el websocket del rosbridge.

Durante las pruebas se utilizan NPCs como ayuda para generar nodos, pero por la cantidad de recursos que gastaba la máquina se desistió y preferimos utilizar posiciones. El funcionamiento de las posiciones se realizaron con matrices utilizando filtro de kalman y algunos otros procesos complementando el A*.

Posicionamiento

El mapa generado por unity nos confirmó la desigualdad en las posiciones. Decidimos tomar cada punto extremo del mapa, como un punto de estadía para recorrer la mayor cantidad de posiciones dentro del simulador.

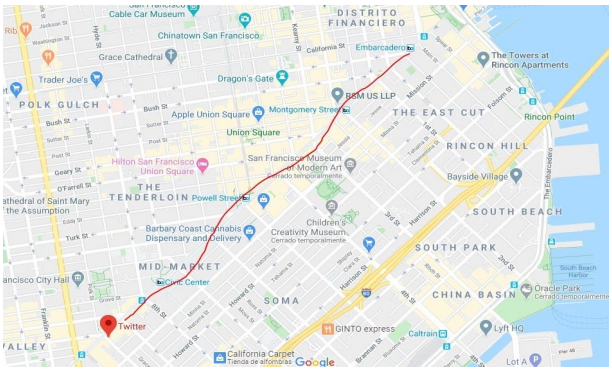

Ilustración 7 . Ruta Imaginaria contradictoria del simulador.

Desde el extremo más cerca de twitter pudimos encontrar “The embarcadero”, que generaba una línea casi perfecta hacia twitter. Al momento de manejar el automóvil a dicha posición, el mapa quedaba en otro extremo. y no dejaba avanzar.

Para solucionar este problema, reducimos el área de manejo, para recorrer solo posiciones válidas, el algoritmo encontraba los extremos (nodos) y recorría de forma simétrica las calles. Encontrando el punto de referencia más cercano a twitter.

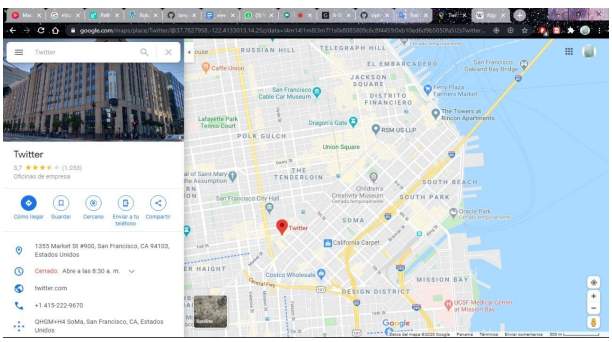

Ilustración 8. Posición exacta de twitter

Al momento de llegar a la conversión del punto exacto, logramos generar la ruta señalada, pero basada en el simulador. Durante el proceso de llegada se utilizó el reset entre nodos, mediante un script en el agente, para redireccionar el vehículo cuando detectara un choque de extremos.

Ilustración 9. Posición más cercana del auto en el punto asociado a twitter.

IV - CONCLUSIONES

Este trabajo fue muy difícil de realizar ya que se necesitaban demasiados recursos para poder ejecutarlo , pero al final lo logramos, no de la manera que se esperaba, pero se pudo cumplir el objetivo del laboratorio el cual era crear un agente que a través de un algoritmo pudiera llegar a un lugar. Otro problema que pudimos encontrar fue la falta de congruencia entre los mapas. Pero la experiencia de manejar un simulador y poder recrear un movimiento autónomo por inteligencia artificial, fue una gran forma de implementar algoritmos con agentes utilizando recursos dentro de nuestra etapa universitaria.

V - REFERENCIAS

N. Swift, "Nicholas Swift", *Easy A* (star) Pathfinding,* 2017. [Online]. Available: https://medium.com/@nicholas.w.swift/easy-a-star-pathfinding-7e6689c7f7b2.[Accessed: 26- Jul- 2020].

L. Simulator, "LGSVL Simulator", *An Autonomous Vehicle Simulator*, 2020. [Online]. Available: https://www.lgsvlsimulator.com/. [Accessed: 26- Jul- 2020].

"lgsvl/simulator", *GitHub*, 2020. [Online]. Available: https://github.com/lgsvl/simulator. [Accessed: 26- Jul- 2020].

"Algoritmo de búsqueda A*", *Es.wikipedia.org*, 2020. [Online]. Available: https://es.wikipedia.org/wiki/Algoritmo_de_b%C3%BAsqueda_A*. [Accessed: 26- Jul- 2020].

"U. Technologies, "Unity - Unity", *Unity*, 2020. [Online]. Available: https://unity.com/. [Accessed: 26- Jul- 2020].

S. Montoya, "Saul Montoya*", Tutorial para transformar Sistemas de Coordenadas de Puntos XY con Python Pandas y Pyproj,* 2020, [Online]. Available: https://gidahatari.com/ih-es/tutorial-para-transformar-sistemas-de-coordenadas-de-puntos-xy-con-python-pandas-y-pyproj. [Accessed: 26- Jul- 2020].