Universidad de La Laguna

Informe estrategias de Búsqueda

El objetivo del desarrollo de la actividad práctica es la utilización de estrategias de búsqueda como propuesta de resolución en la determinación de la planificación de trayectorias para coches autónomos.

Daniel Dóniz García.
alu0101217277@ull.edu.es

Alberto Ravelo Cordobés.
alu0101136042@ull.edu.es

Introducción

- **Descripción del problema a resolver.**

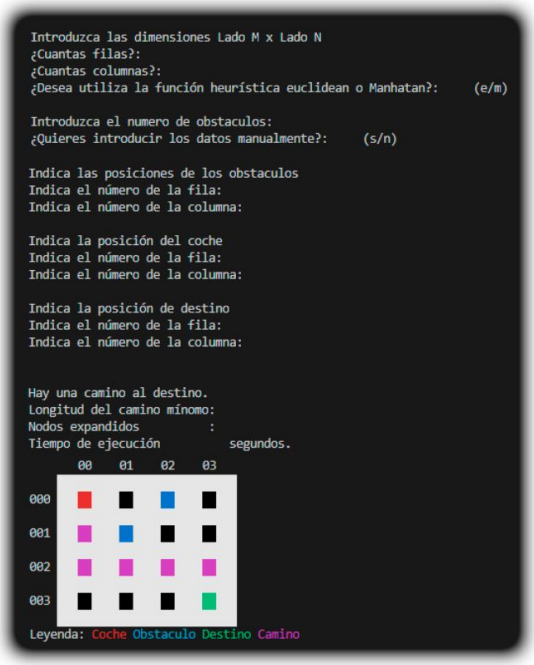
El propósito es la utilización de estrategias de búsqueda como propuesta de resolución en la determinación de la planificación de trayectorias para coches autónomos. El entorno que utilizaremos es una rejilla de dimensiones M x N, esta rejilla está constituida por casillas libres y ocupadas por obstáculos, donde el coche puede desplazarse únicamente por las casillas libres colindantes a la que se encuentra.

- **Formulación del problema como un Espacio de Estados, identificando el Estado Inicial, Estado Final, Estados y Operadores.**

Tenemos un tablero de MxN en principio todas las posiciones tienen el estado a 0, introduces los obstáculos, se ponen a 1, el coche, se pone a 2 y la meta, se pone a 3. Este es el estado inicial del problema, el coche solo puede comprobar los estados de las posiciones Norte Sur Este y Oeste, contamos con un mapa del entorno y con las posiciones inicio, donde aparece el coche por primera vez y meta, donde se encuentra el objetivo de nuestra búsqueda de un camino. Cuando se encuentra el camino se ponen los estados de las posiciones del camino a seguir a 4 y se imprime.

Entorno de Simulación y Programación.

- **Descripción de la interfaz y su implementación. Incluir captura de pantalla que muestre las funcionalidades de la interfaz.**



La interfaz consta de preguntas que se van mostrando por pantalla sucesivamente tras responder la pregunta anterior. Finalmente, antes de mostrar el entorno de simulación se muestra por consola los resultados de longitud de camino mínimo, nodos expandidos y tiempo de ejecución obtenidos. posteriormente se muestra el entorno junto a una leyenda en la cual se explica cada uno de los colores

que se muestran en pantalla.

- **Argumentación sobre el entorno de programación escogido.**

La razón de utilizar C++ es porque es un lenguaje que nos han estado enseñando en los últimos años y nos gustaba la idea de profundizar más en este lenguaje. Nos pareció interesante aprender a realizar una rejilla tan grande en este entorno de programación.

Metodología de Trabajo.

- **Si se ha trabajado en grupo, cada miembro debe indicar las tareas llevadas a cabo para el desarrollo e implementación de la práctica.**

Fundamentalmente hemos trabajado sincronizados y en equipo, nuestra dinámica ha sido cada uno desarrollar en su equipo almacenando los archivos del programa en una carpeta que tenemos ambos a través de la nube de Dropbox, de esta manera teníamos pleno acceso ambos a todo. A grandes rasgos hemos intentado los dos trabajar en todo un poco desde desde un principio así éramos conscientes de cómo continuar posteriormente. Al mismo tiempo que Dóniz comenzaba a buscar información sobre los algoritmos de búsqueda a utilizar, Ravelo se encargaba de buscar posibles interfaces a utilizar. Posteriormente Ravelo comenzó el simulador del entorno, posteriormente lo concluyó Dóniz, junto a la interfaz. Continuamos buscando información de los algoritmos de búsqueda conjuntamente, un algoritmo cada uno y finalmente desarrollamos conjuntamente las Heurísticas de Manhattan distance y Euclidean distance. Concluimos la práctica con Dóniz comenzando el informe que posteriormente terminó Ravelo.

Algoritmo de Búsqueda.

- **Pseudocódigo indicando cómo se ha llevado a cabo la implementación.**

```
Abiertos = [inicio]
Cerrado =  []
f'(inicial) = h'(inicio, meta)
current = inicio
```

repetir mientras abiertos != [] {

recorre Abiertos(*it) de principio a fin{
compara f(*it) con lowestF de manera que al acabar ahí está el menor costo y ese pasa a ser el nuevo current.
}
si current == meta entonces existe camino=true y sale del bucle sino{

vecinos [] = busca vecinos(current) //N S E W
recorre (vecinos[i], mientras i < vecinos.size() i++){
busca vecinos[i] en Cerrado, si no está {
gs = g + h (cur, vecinos[i])
busca vecinos[i] en Abiertos, si no está {
Abiertos.push_back(vecincos[i])
} else {va al final del while}
camefrom(vecinos[i]) = current
g(vecinos[i])= gs
f(vecinos[i]) = gs + h(vecinos[i], meta)
}}//fin del bucle for
}}//fin del else y del while

si existe camino ==true entonces{
path.pushback(meta)
repite mientras current != inicio{
path.pushback(cur)
estado(current)=4->para la impresión
cur = camefrom(cur)
}}

- **Estructuras de datos utilizadas para su implementación.**

-Un doble puntero a la clase casilla que actúa de mapa de casillas.

-3 vectores de size_t que sirven para calcular el camino a seguir

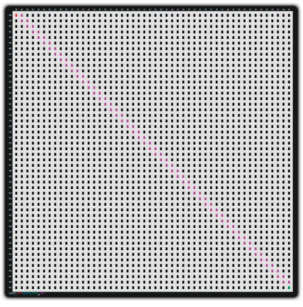
- Una clase entorno y otra casilla.

- Función heurística utilizada y justificación.

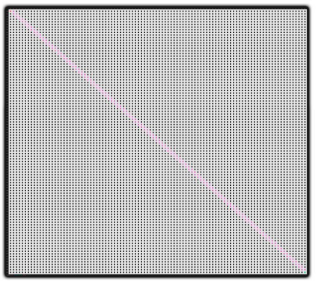
Hemos utilizado la funciones heurísticas de: la distancia euclídea y la distancia Manhattan.
La distancia euclídea ha sido utilizada por su sencillez de comprender gracias a que su definición coincide con el concepto de distancia. Gracias a que tenemos variables homogéneas con unas medidas de unidades iguales y se conoce la matriz de varianzas (el grid que contiene el inicio, destino y obstáculos).
La distancia Manhattan la hemos utilizado porque es la que más sentido tiene utilizarla dado que no se pueden atravesar los obstáculos del grid. Teniendo en cuenta que no podemos escoger caminos en diagonal la distancia Manhattan es semejante a la distancia euclídea, dado que esta última debe ir en “escalera”. Finalmente queremos destacar que la distancia Manhattan es mayor que la distancia euclídea, pero también es más real en la práctica. La distancia euclídea se expande en círculo, en cambio, la distancia Manhattan se expande en rombo.

Evaluación Experimental del Algoritmo de Búsqueda con dos Funciones Heurísticas:

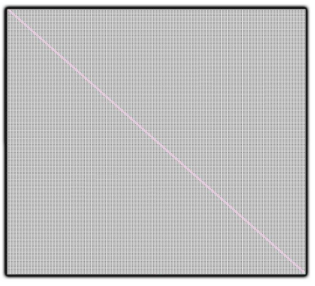
- Los resultados deben mostrarse en una tabla y deben incluir: el número total de nodos expandidos por el algoritmo, la longitud del camino mínimo y el tiempo para obtener el resultado (sin tener en cuenta la interfaz de visualización) por cada función heurística.
- Se considerarán como mínimo tres tamaños de escenarios diferentes (pequeño 3 , mediano 2 y grande 1) sin obstáculos. Dónde grande se refiere al tamaño máximo que es capaz de soportar el programa. Por tanto, habrá que generar como mínimo 3 escenarios diferentes, mostrando los resultados para cada función heurística.



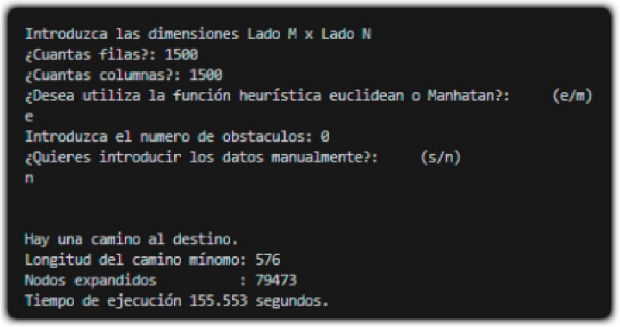
50 x 50



100 x 100



200 x 200



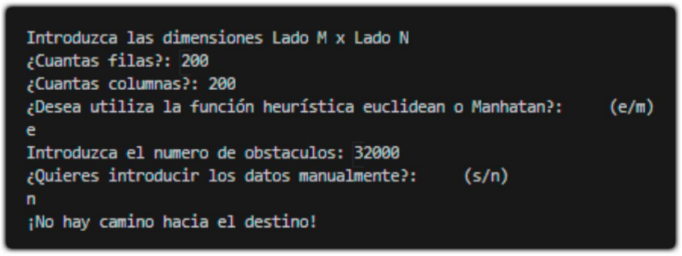
Aunque 1.500 filas x 1.500 columnas soporta el entorno teniendo en cuenta que se no se muestra correctamente el grid en pantalla y que tarda 65,243 segundos.

- Se considerarán como mínimo tres tamaños de escenarios diferentes (pequeño, mediano y grande) con un 25% de obstáculos, 50% de obstáculos y 80% de obstáculos generados de forma aleatoria. Por tanto, habrá que generar como mínimo 9 escenarios diferentes, mostrando los resultados para cada función heurística.

Grande 200 x 200

80% Obstáculos
32.000 Obstáculos

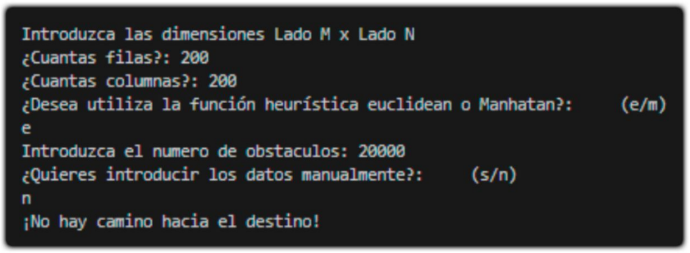
10 Pruebas.
10 Fallos.



Grande 200 x 200

50% Obstáculos
20.000 Obstáculos

10 Pruebas.
10 Fallos.



Grande 200 x 200

25% Obstáculos
10000 Obstáculos

5 Pruebas.
5 Éxitos.

```
Introduzca las dimensiones Lado M x Lado N
¿Cuantas filas?: 200
¿Cuantas columnas?: 200
¿Desea utiliza la función heurística euclidean o Manhatan?:     (e/m)
e
Introduzca el numero de obstaculos: 10000
¿Quieres introducir los datos manualmente?:     (s/n)
n


Hay una camino al destino.
Longitud del camino mínomo: 121
Nodos expandidos          : 2366
Tiempo de ejecución 0.09912 segundos.
```

Grande 100 x 100

80% Obstáculos
8000 Obstáculos

10 Pruebas.
10 Fallos.

```
Introduzca las dimensiones Lado M x Lado N
¿Cuantas filas?: 100
¿Cuantas columnas?: 100
¿Desea utiliza la función heurística euclidean o Manhatan?:     (e/m)
m
Introduzca el numero de obstaculos: 8000
¿Quieres introducir los datos manualmente?:     (s/n)
n
¡No hay camino hacia el destino!
```

Grande 100 x 100

50% Obstáculos
5000 Obstáculos

10 Pruebas.
10 Fallos.

```
Introduzca las dimensiones Lado M x Lado N
¿Cuantas filas?: 100
¿Cuantas columnas?: 100
¿Desea utiliza la función heurística euclidean o Manhatan?:     (e/m)
e
Introduzca el numero de obstaculos: 5000
¿Quieres introducir los datos manualmente?:     (s/n)
n
¡No hay camino hacia el destino!
```

Grande 100 x 100

25% Obstáculos
2500 Obstáculos

5 Pruebas.
5 Éxitos.

```
Introduzca las dimensiones Lado M x Lado N
¿Cuantas filas?: 100
¿Cuantas columnas?: 100
¿Desea utiliza la función heurística euclidean o Manhatan?:     (e/m)
e
Introduzca el numero de obstaculos: 2500
¿Quieres introducir los datos manualmente?:     (s/n)
n


Hay una camino al destino.
Longitud del camino mínomo: 39
Nodos expandidos          : 303
Tiempo de ejecución 0.002549 segundos.
```

Grande 50 x 50

80% Obstáculos
2000 Obstáculos

10 Pruebas.
10 Fallos.

```
Introduzca las dimensiones Lado M x Lado N
¿Cuantas filas?: 50
¿Cuantas columnas?: 50
¿Desea utiliza la función heurística euclidean o Manhatan?:     (e/m)
m
Introduzca el numero de obstaculos: 2000
¿Quieres introducir los datos manualmente?:     (s/n)
n
¡No hay camino hacia el destino!
```

Grande 50 x 50

50% Obstáculos
1250 Obstáculos

10 Pruebas.
10 Fallos.

```
Introduzca las dimensiones Lado M x Lado N
¿Cuantas filas?: 50
¿Cuantas columnas?: 50
¿Desea utiliza la función heurística euclidean o Manhatan?:     (e/m)
m
Introduzca el numero de obstaculos: 1250
¿Quieres introducir los datos manualmente?:     (s/n)
n
¡No hay camino hacia el destino!
```

Grande 50 x 50

25% Obstáculos
625 Obstáculos

5 Pruebas.
5 Éxitos.

```
Introduzca las dimensiones Lado M x Lado N
¿Cuantas filas?: 50
¿Cuantas columnas?: 50
¿Desea utiliza la función heurística euclidean o Manhatan?:     (e/m)
m
Introduzca el numero de obstaculos: 625
¿Quieres introducir los datos manualmente?:     (s/n)
n


Hay una camino al destino.
Longitud del camino mínomo: 62
Nodos expandidos          : 483
Tiempo de ejecución 0.004854 segundos.
```

Conclusiones

- **Mostrar las principales conclusiones obtenidas tras la realización de la práctica.**

Para concluir, cabe destacar que en esta demostración no se aprecia diferencia alguna en el coste de ambos algoritmos, siempre que no hayan obstáculos, ya que en este caso no podemos saber los estados de las posiciones vecinas diagonales. Debido a esto la distancia euclídea no resulta tan eficaz como podría, ya que de ser así las operaciones se reducirían a la mitad.

Bibliografía

1. https://en.wikipedia.org/wiki/A*_search_algorithm
2. https://en.wikipedia.org/wiki/Euclidean_distance
3. https://en.wikipedia.org/wiki/Taxicab_geometry