PRÁCTICAS DE AMPLIACION DE ROBÓTICA

MARTA ROSA FLORES

DANIEL TRIANO MORENO

Índice:

[EJERCICIO 1 3](#_Toc65138959)

[Solución 1.1: 3](#_Toc65138960)

# EJERCICIO 1

Se desea simular el movimiento de un vehículo con tracción diferencial con los siguientes parámetros: distancia entre ruedas 0.8 m, radio de las ruedas 0.1 m, velocidad máxima de las ruedas 15 rad/s. Para ello se tendrán que implantar las ecuaciones de movimiento así como el modelo dinámico simplificado de los dos actuadores. Se considerará una constante de tiempo de 0.12 s, ganancia unidad. Se pide:

1. Simular el movimiento en tiempo discreto con periodo de T = 25 ms partiendo desde parado y aplicando actuaciones constantes. Considerar las siguientes situaciones: trayectoria recta, giro a la izquierda, giro a la derecha y velocidad lineal nula.

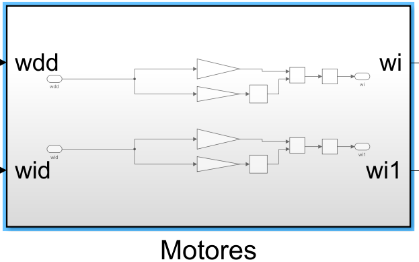
## Solución 1.1:

Antes de realizar alguna explicación de la solución obtenida, se van a enumerar las fórmulas necesarias para una correcta explicación:

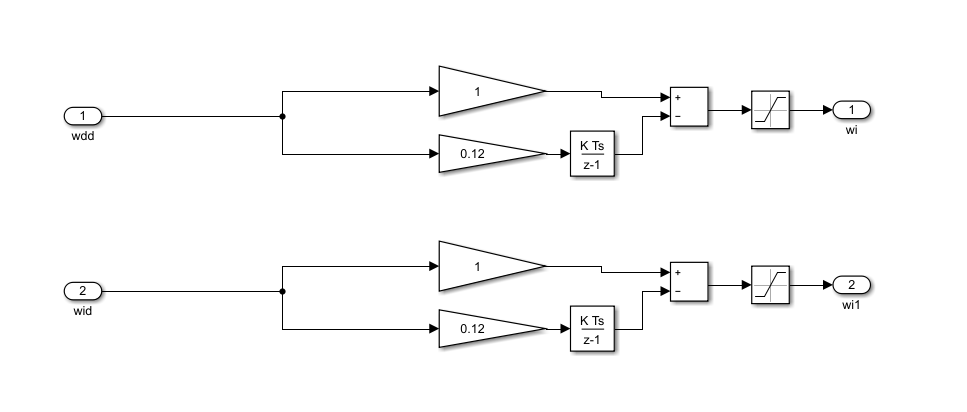
|  |  |
| --- | --- |
|  | () |
|  | () |
|  | () |

|  |  |
| --- | --- |
|  | () |
| 𝛥𝑠 = 𝑣 ⋅ 𝛥𝑡 | () |
| 𝛥𝜙 = 𝜔 ⋅ 𝛥𝑡 | () |
| 𝛥𝑥 = −𝑠𝑖𝑛(𝜙) ∙ 𝛥𝑠 | () |

Para poder resolver este apartado se han realizado distintos bloques. El primer de ellos serían los motores (ver **Figura 1**) y para ello se tratará de implementar las ecuaciones (3) y (4), tal y como muestra la **Figura 2**.



**Figura 1: Imagen del sub sistema “motores”**



**Figura 2: Interior del sub sistema “motores”**

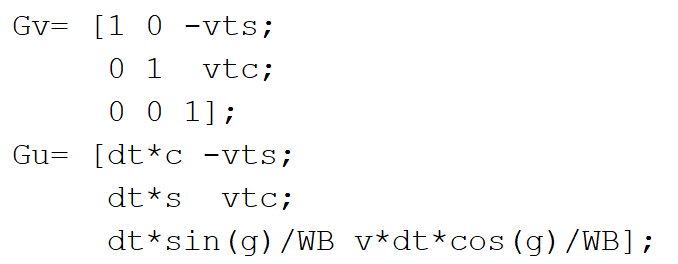
Tras una inspección superficial de las ecuaciones (3) y (4) se puede ver que son simétricas, donde simplemente varían las velocidades, izquierda para la primera y derecha para la segunda.

Para el caso de la rueda derecha se multiplica por una ganancia unitaria y se resta a la constante de tiempo discretizada. Se realiza el mismo procedimiento para ambos casos si bien al final del recorrido se le añade una saturación en 15 rad/s.

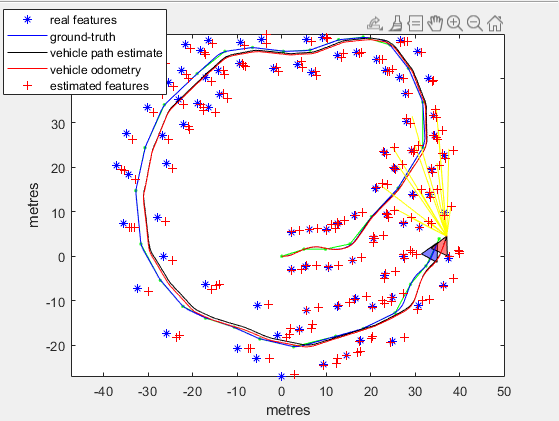
# EJERCICIO 2

1. **Completar la fase de predicción (función predict) con las matrices jacobianas correspondientes al modelo de conducción anterior.**

En primer lugar, se carga el mapa a recorrer tal y como se comenta en el enunciado. Si se acude a la función predict se puede observar como están definidas las drivadas parciales en función de u[V,G], tal y como muestra loa figura de abajo:



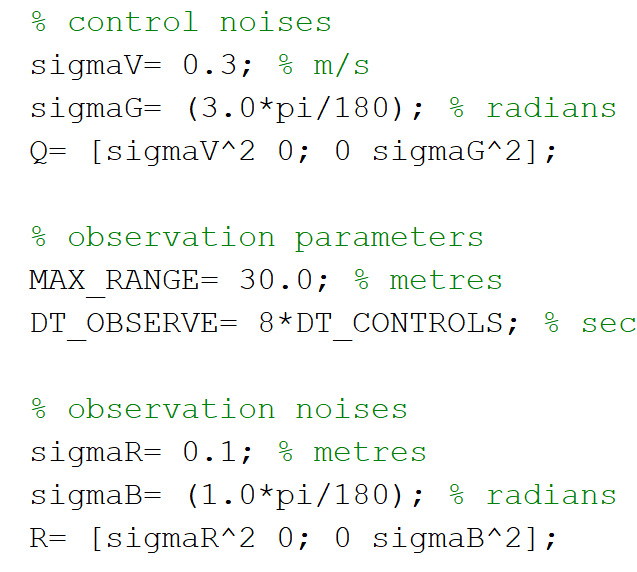
Tras ejecutar la función **ekfloc(lm,wp)** (ver figura de abajo), se analizan los siguientes resultados.



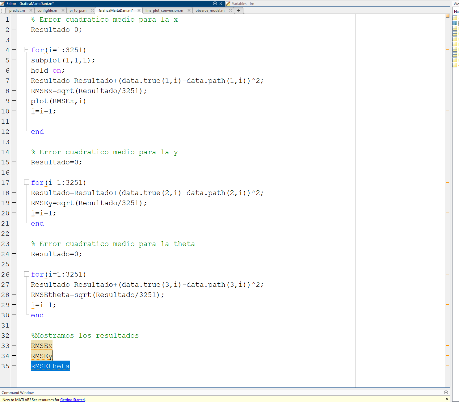
Se observa una cierta discrepancia entre la odometría y el camino recorrido, así como las marcas de entorno previstas y las reales.

1. **Calcular el error cuadrático medio en la distancia y ángulo de orientación a lo largo de la trayectoria, según la ecuación:**

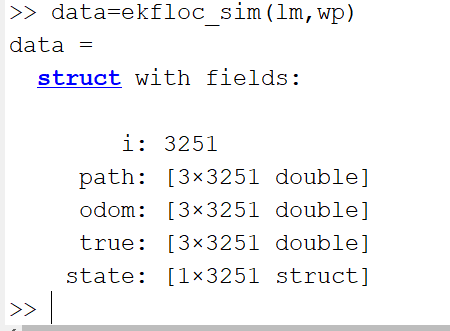
Si queremos analizar cómo afectas los distintos parámetros de ruido, ya sea de control o de observación, debemos acudir al archivo ***configfile,*** donde se modificar los parámetros correspondientes a las distintas sigmas (ver figura de abajo).



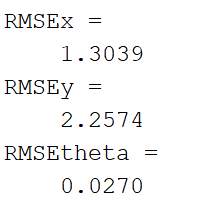
Para ver el error medio tanto en X, en Y como en Theta, se ha implementado la función llamada ***graficaMartaDani*** (ver figura de abajo),



Donde simplemente cogemos la columna correspondiente a cada variable para poder calcular su error. Se utiliza un bucle for de 3251 iteraciones dado que ese es el tamaño de nuestra función (ver figura de abajo):



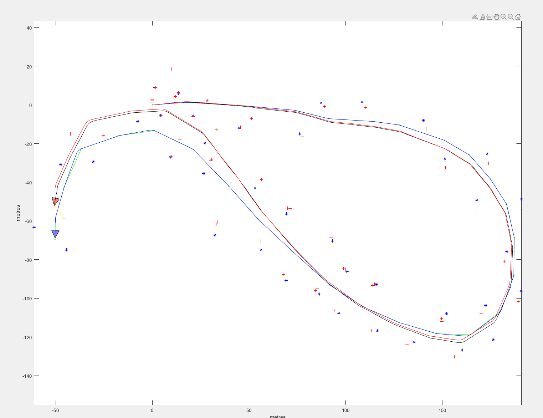
Así pues, se observa el error sin modificar ningún parámetro, obteniendo los siguientes resultados ver figura abajo:



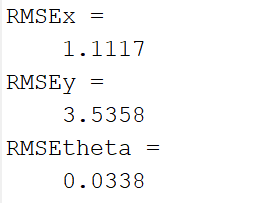
Se observa un mayor error en Y, y casi nulo en theta.

**3. Realizar simulaciones para distintos valores de incertidumbre en el control del vehículo y de la observación (al menos un cambio en cada uno de ellos). Modificar dichas incertidumbres en el fichero de parámetros (configfile). Estudiar su influencia en el resultado calculando el error cuadrático medio en cada caso y comentar las conclusiones.**

Si ahora se aumenta el valor de SigmaV por tres veces el valor anterior, se observa la figura de abajo:

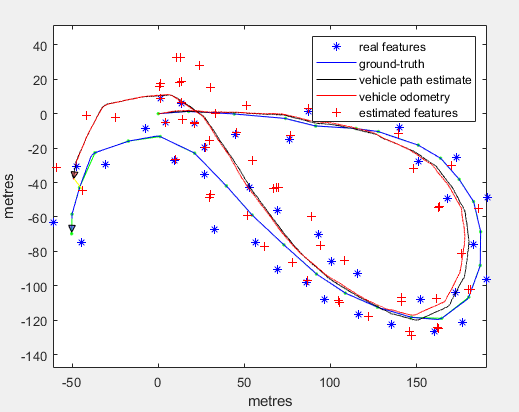


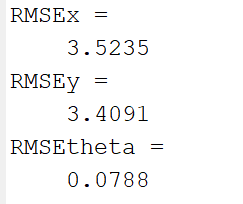
Se puede ver a simple vista y sin realizar cálculos que el error ha aumentado, sobre todo en el eje Y donde se dispersan más los puntos. Comprobamos eso en al calcularlo (ver figura de abajo):

pasamos de sigmaV 0.3 a 0.9 poner pie de foto

Confirmamos la previsión, el error en Y ha aumentado al aumentar el ruido de control SigmaV.

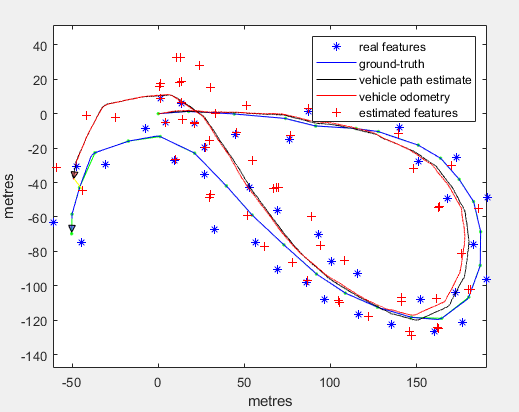
Si ahora modificamos SigmaG (ver foto de abajo), vemos que no solo aumenta el error en Y si no que también aumenta en X (ver figura de aun mas abajo xd).

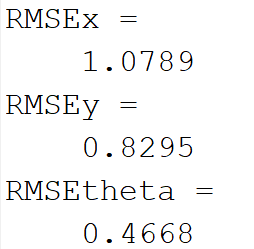
 sigma cambia de sigmaG= (3.0*pi/180); % radians POR sigmaG= (12.0*pi/180



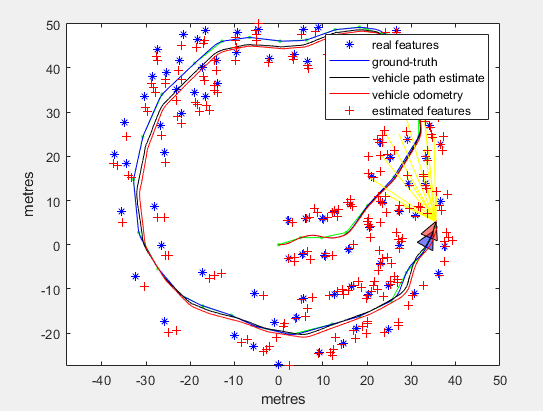
A su vez el error en theta duplica con respecto al anterior caso.

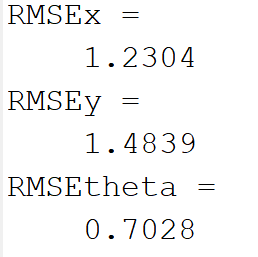
Modificamos a continuación los ruidos de observación, pasando de un SigmaR=0.1 a SigmaR=0.5 metros. Se observa la siguiente gráfica (ver abajo) donde claramente el error es menor respecto a las anteriores. Todo esto se confirma al ver el error real (figura de mas abajo)



resultados

Al realizar la ultima modificación, pasando a una sigmaB tres veces mayor, se observa un aumento del error en theta bastante notable. El error en X y en Y también aumenta pero en menor proporción. Todo esto se refleja en las foto de abajo



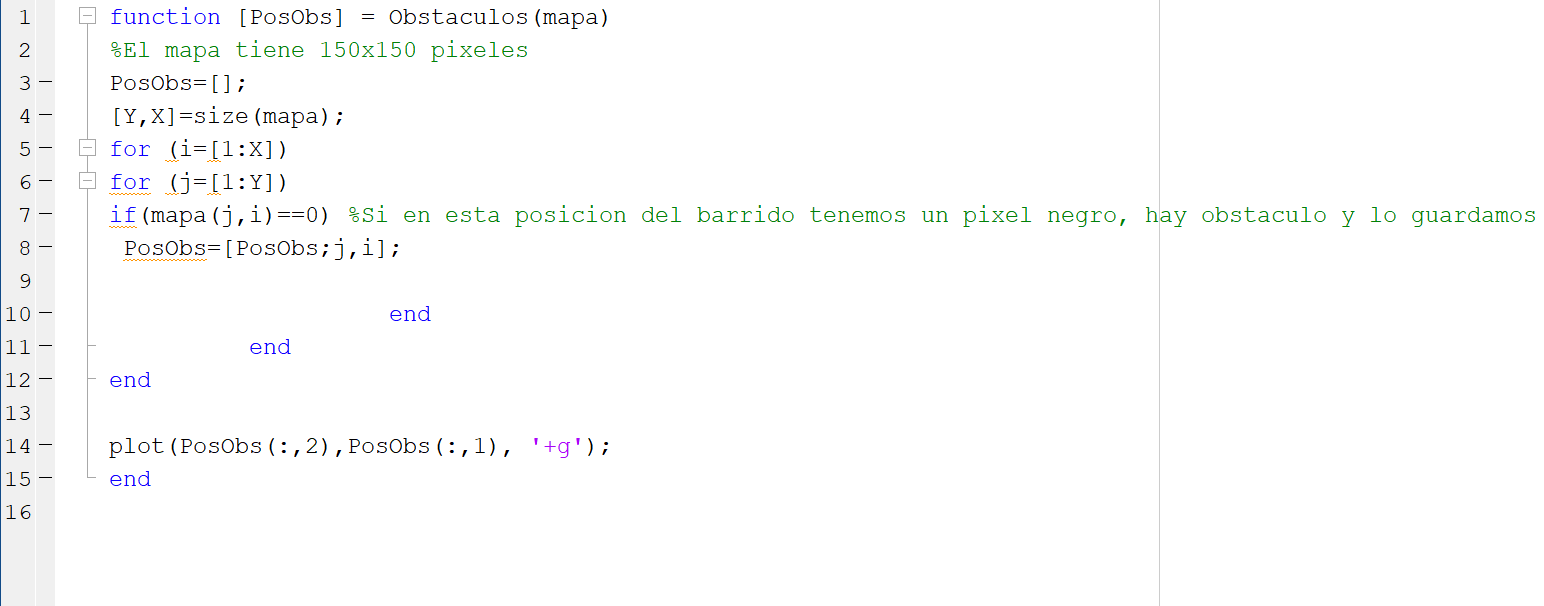


Como conclusión podemos ver que al aumentar el ruido la posición real del robot varía más que el camino que sigue la odometria. Si disminuimos el error se acercará por lo tanto al camino de la odometría, aumentando así la exactitud.

# EJERCICIO 3

**1. Completar el script denominado plantilla para conseguir la navegación reactiva del robot según el método de campos potenciales.**

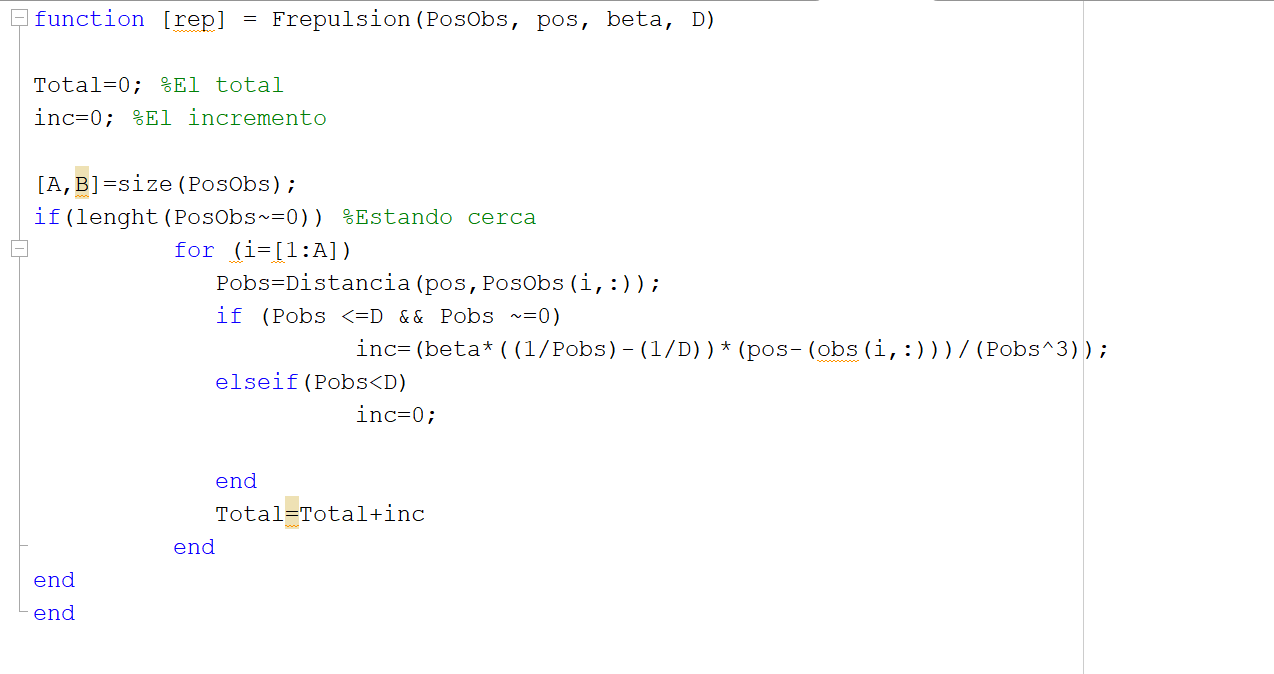
Se ha hecho una función que se encarga de detectar los obstáculos del mapa, para ello hemos hecho dos bucles for de una amplitud X e Y, que serán las proporciones del mapa seleccionado. Si se detecta un pixel de valor de valor 0, significará que se ha encontrado un pixel negro y por lo tanto un obstáculo (ver figura de abajo poner ref):



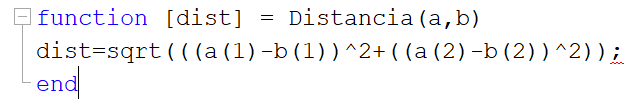
Mediante las siguientes ecuaciones, proporcionadas en las diapositivas de clase, se han implementado dos funciones, para calcular la fuerza de atracción y de repulsión respectivamente (ver ecuaciones 8 y 9)

|  |  |  |
| --- | --- | --- |
|  |  | () |
| = |  | () |

Dado que la fuerza de repulsión vale cero si no se cumple la condición, debemos emplear un condicionante IF, como se detalla en figura de abajo:



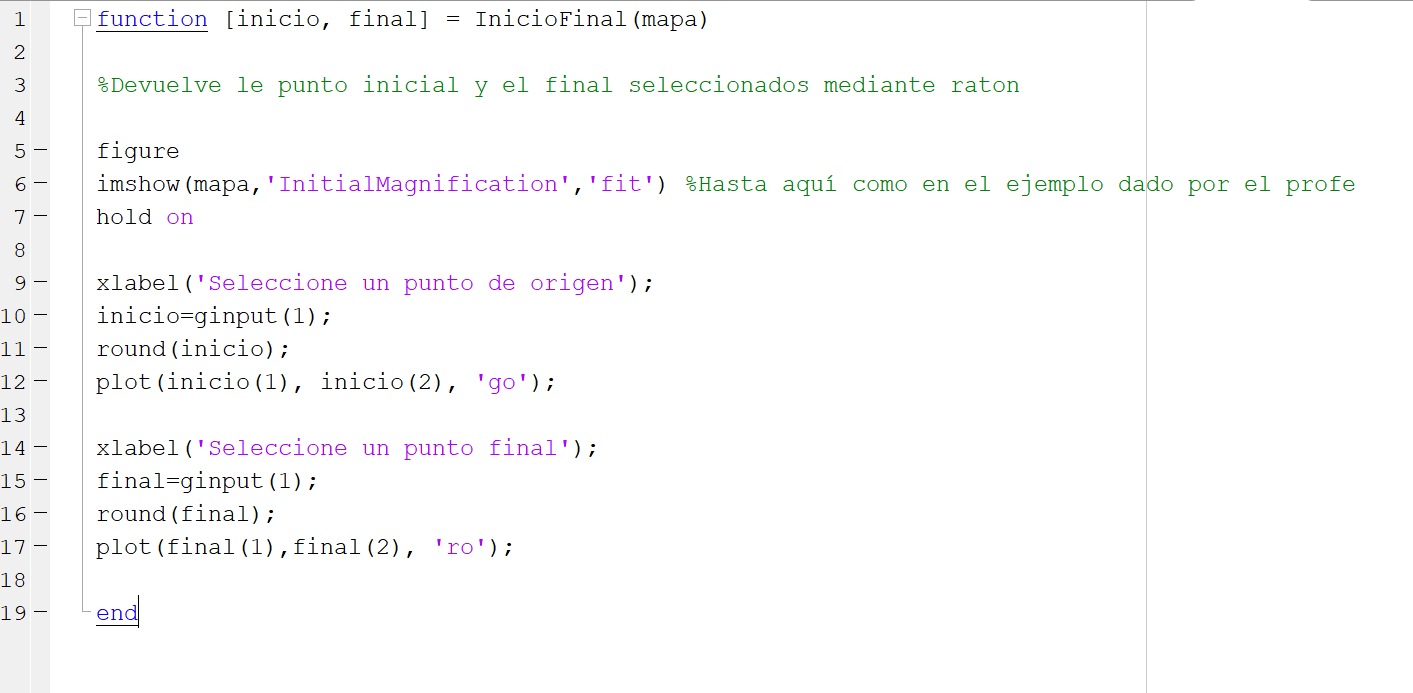
Se implementa una sencilla función llamada Distancia que calcula la distancia que hay entre dos puntos, como se puede observar en foto abajo:



NO RECUERDO COMO IMPLEMENTAMOS ESTO PARA QUE SE LEYERA BIEN

**2. Modificar adecuadamente el programa de Matlab anterior para permitir poder elegir cualquier origen y destino sobre el mapa, empleando para ello la función ginput().**

Dado que en este aparto se pide poder elegir de manera manual la posición de inicio y de final, lo que se ha realizado es una función que muestre en pantalla un mensaje para que el usuario elija un punto de inicio y al hacer click se guardará es punto como entrada. De la misma manera se hace con el punto final. Esto se puede ver desarrollado en la figura de abajo:



**3. Realizar varios experimentos con diferentes orígenes y destinos. Cambiar los parámetros del método, como α y β, o cualquier otro, para tratar de mejorar la situación. Poner el destino y origen en una situación de mínimo local (situación de trampa local), ¿es posible que el método pueda encontrar la solución sin modificarlo y cambiando solo los parámetros anteriormente mencionados?**