CUARTA PRÁCTICA

EVITAR OBSTÁCULOS MEDIANTE CAMPOS POTENCIALES

MIGUEL IAN GARCÍA POZO

GIERM

ESCUELA DE INGENIERÍAS INDUSTRIALES

UNIVERSIDAD DE MÁLAGA

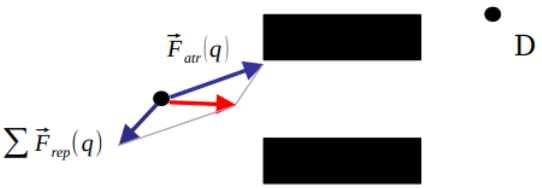
Índice

Navegación Reactiva

- Explicación

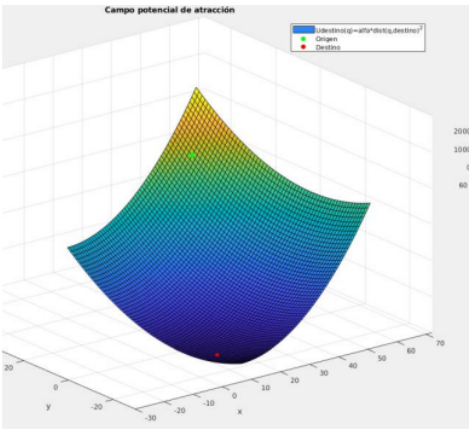
El objetivo de esta práctica es simular la navegación reactiva del robot por el mapa de un punto inicial a un punto final, usando para ello el método de campos potenciales. Este método consiste en definir unas fuerzas que van a afectar al movimiento del robot según el entorno. Veámoslo en un esquema:

$$\vec{F}_{res}(q)=\vec{F}_{atr}(q)+\sum \vec{F}_{rep}(q)$$

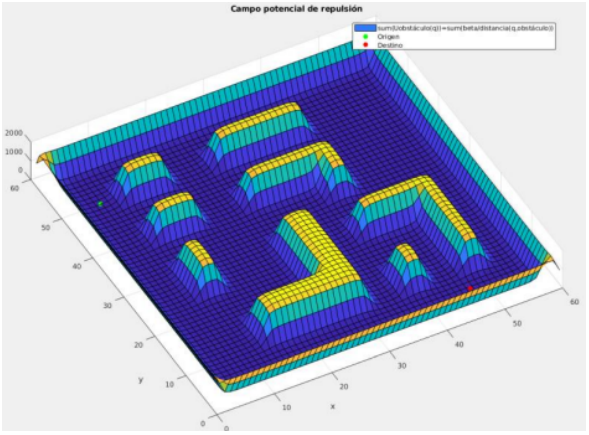


Vamos a definir de donde provienen cada una de estas fuerzas:

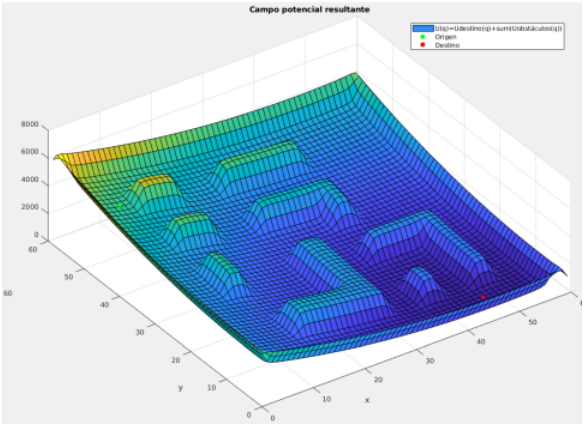
- La fuerza de atracción consiste en una fuerza que va a tomar la dirección de la función gradiente que apunta hacia el punto destino desde el punto en el que está el robot.



- La fuerza de repulsión consiste en la fuerza que va a tomar la dirección contraria al punto más cercano de un obstáculo, es decir, cuando el robot detecta un obstáculo, este va a generar una fuerza en la dirección contraria al objeto.



- La fuerza resultante consiste en la suma de ambas fuerzas.



- Desarrollo

Ahora que hemos explicado la idea de los campos potenciales, empecemos a calcular las fuerzas que van a actuar en el robot.

- Fuerza de atracción:

$$U_{destino}(q) = \frac{1}{2}\cdot\alpha\cdot\rho_{destino}^{\ 2} \longrightarrow F_{atr}(q) = -\nabla U(q) = -\alpha\cdot\rho_{destino}\cdot\nabla\rho_{destino}$$

$$F_{atr}(q) = \alpha\cdot(q - q_{destino})$$

Siendo:

α el factor de escalado

$$\rho_{destino} = dist(q, destino) = \left\|q - q_{destino}\right\|$$

q la posición del robot y $q_{destino}$ la del destino

- Fuerza de repulsión:

$$U_{obstáculos}(q) = \begin{cases} \frac{1}{2}\cdot\beta\cdot\left(\frac{1}{\rho_{obs}} - \frac{1}{D}\right)^2 & si\ \rho_{obs} \le D \\ 0 & si\ \rho_{obs} > D \end{cases}$$

$$F_{rep}(q) = -\nabla U_{obstáculo}(q) = \begin{cases} \beta\cdot\left(\frac{1}{\rho_{obs}} - \frac{1}{D}\right)\cdot\frac{q - q_{obstáculo}}{\rho_{obs}^{\ 3}} & si\ \rho_{obs} \le D \\ 0 & si\ \rho_{obs} > D \end{cases}$$

Siendo:

β el factor de escalado

$$\rho_{obs} = dist(q, obstáculo) = \left\|q - q_{destino}\right\|$$

D distancia de influencia del objeto

- Fuerza resultante:

$$F_{res}(q) = F_{atr}(q) + F_{rep}(q)$$

Teniendo ya todos los cálculos planteados, vamos a implementarlo en Matlab.

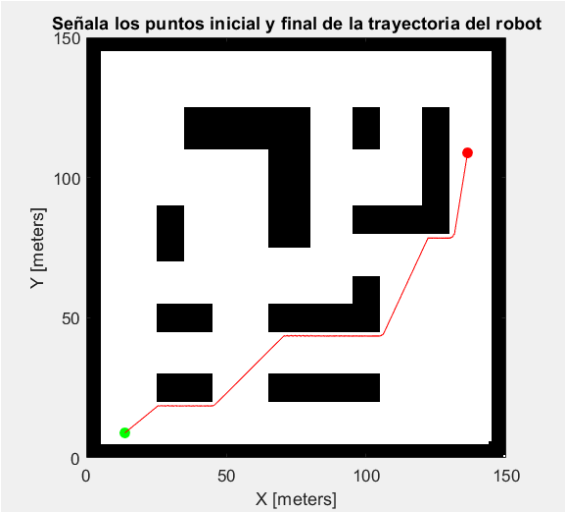
```
% Definimos a 0 el vector con la fuerza de repulsión
Frep = [0 0];
% Calculamos con la ecuación el valor de la fuerza de atracción
Fatr = alfa.*(destino - robot(1:2));
% Usamos la función que nos devolverá la distancia a los obstáculos cercanos
obs = SimulaLidar(robot,mapa,angulos,max_rango);
% Comprobamos si tenemos un objeto cercano recorriendo la matriz "obs"
for i = 1:length(obs)
    % Comprobamos que el valor que estamos viendo no es nan, que indica
    % fuera de rango
   if (not(isnan(obs(i,1))))
       % calculamos la distancia entre el robot y el punto del
       % obstáculo
       rho = (robot(1:2)-obs(i,:));
       d = sqrt(rho(1)^2+rho(2)^2);
       % Comprobamos si la distancia es menor a la umbral definida
       % anteriormente
       if (d<=D)
           % Sumamos a la fuerza de repulsión acumulada, la fuerza de repulsión que
           % genera el punto que estamos comprobando
           Frep = Frep + beta*((1/d)-(1/D))*((robot(1:2)-obs(i,:))/d^3);
       end
   end
end
% Calculamos la Fuerza resultante
Fsol = Fatr + Frep;

% Hacemos unitario el vector fuerza para que no influya en la velocidad
unitario = Fsol/sqrt(Fsol(1)^2+Fsol(2)^2);
% Definimos la nueva posición del robot tras moverse
robot = [robot(1)+unitario(1)*v robot(2)+unitario(2)*v atan2(unitario(2),unitario(1))];
path = [path;robot];    % Se añade la nueva posición al camino seguido
plot(path(:,1),path(:,2),'r'); % Pintamos el movimiento
drawnow
```

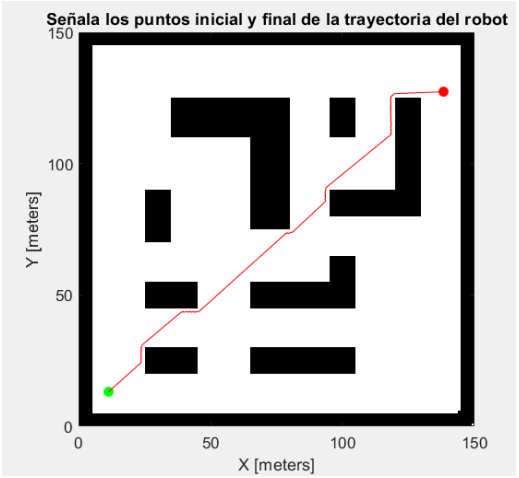
En la imagen vemos como el código calcula la fuerza de atracción y la sumatoria de las fuerzas de repulsión de cada punto del obstáculo para sumarlas y darnos la dirección en la que se va a mover el robot.

- Simulación

Para simular nuestro código, basta con que pongamos el nombre del archivo en la ventana de comandos. La simulación nos da los siguientes resultados:

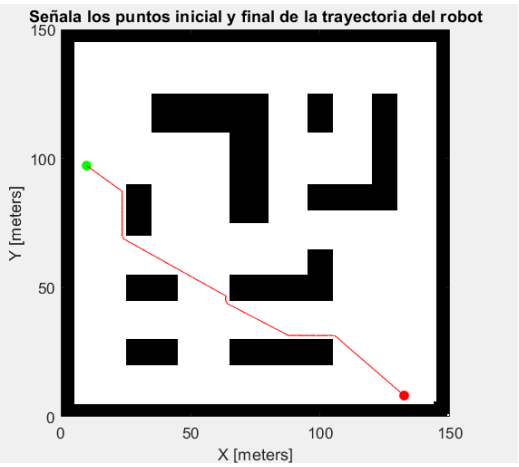
```
plantilla_campos_potenciales
```

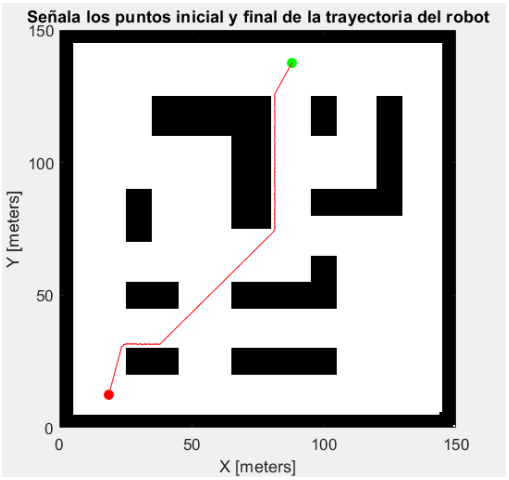




Pruebas cambiando los parámetros

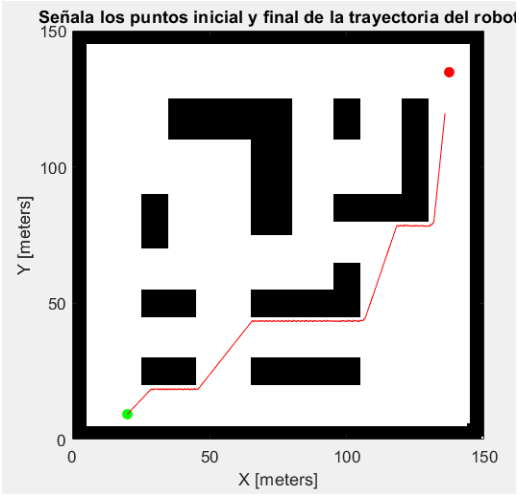
Primero probemos distintos puntos iniciales y finales para probar que el algoritmo funciona correctamente:





Ahora vamos a comprobar cómo afectan a la navegación reactiva del robot el que cambiemos los valores del escalado de las fuerzas de atracción y repulsión.
Si cambiamos el valor de α de 1 a 0.1 y el valor de β de 100 a 800, obtenemos el siguiente resultado:

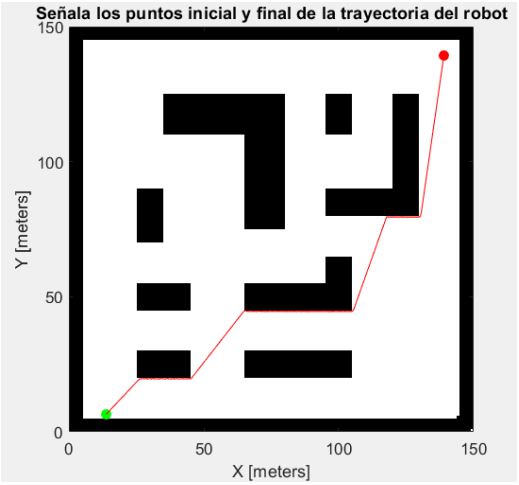
```
alfa=0.1;
beta=800;
```



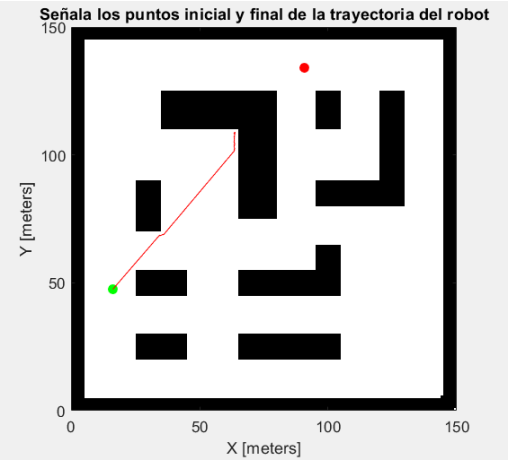
Vemos que la navegación no cambia prácticamente en nada, con la diferencia de que al ser α muy pequeño, el robot se mueve más lento y es por eso que no ha llegado al destino antes de que se acabase el tiempo.

Por otro lado, si ahora cambiamos el valor de α para ser mayor a β, vemos que el robot no deja prácticamente ningún espacio con la pared.

```
alfa=20;
beta=1;
```



Probemos ahora a guiar al robot hacia un mínimo local.



Podemos ver que se quedará ahí para siempre hasta que se termine el tiempo. Esto ocurrirá para cualquier valor de α y β, ya que estos solo influyen en cuánto atrae o cuanto repelen el destino y los obstáculos, no la dirección que se debe tomar.

Enlace a Github

https://github.com/MiguelIlan/AmpliacionRobotica/tree/main/Rob%C3%B3tica%20m%C3%B3vil/EvitarObstaculos