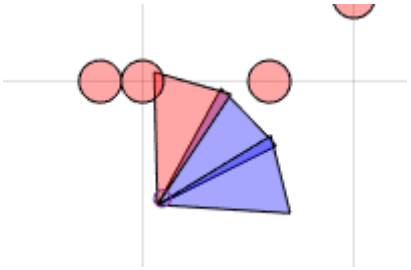
Tp2 - Resolución de problemas por lógica difusa

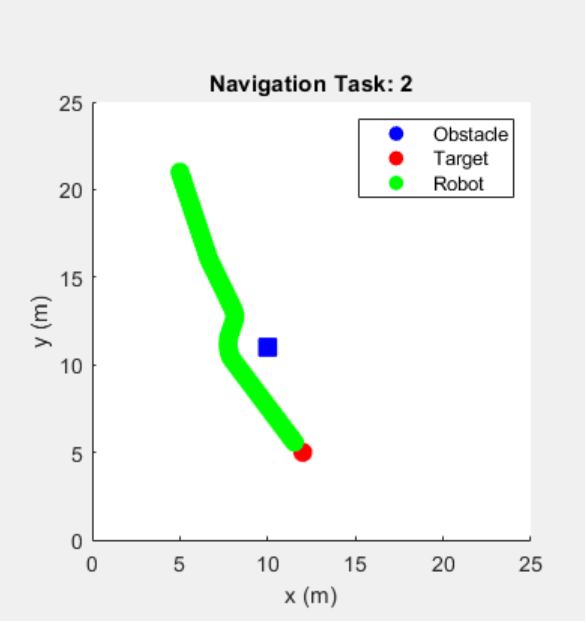
El problema elegido para este trabajo práctico es la simulación de un móvil (“robot”) que se dirige a un destino (“target”) tratando de evadir los obstáculos que encuentra en el camino y controlando su velocidad. La detección de obstáculos se realiza mediante tres sensores de distancia con un ángulo determinado y distancia máxima. El robot se puede mover en cualquier dirección.



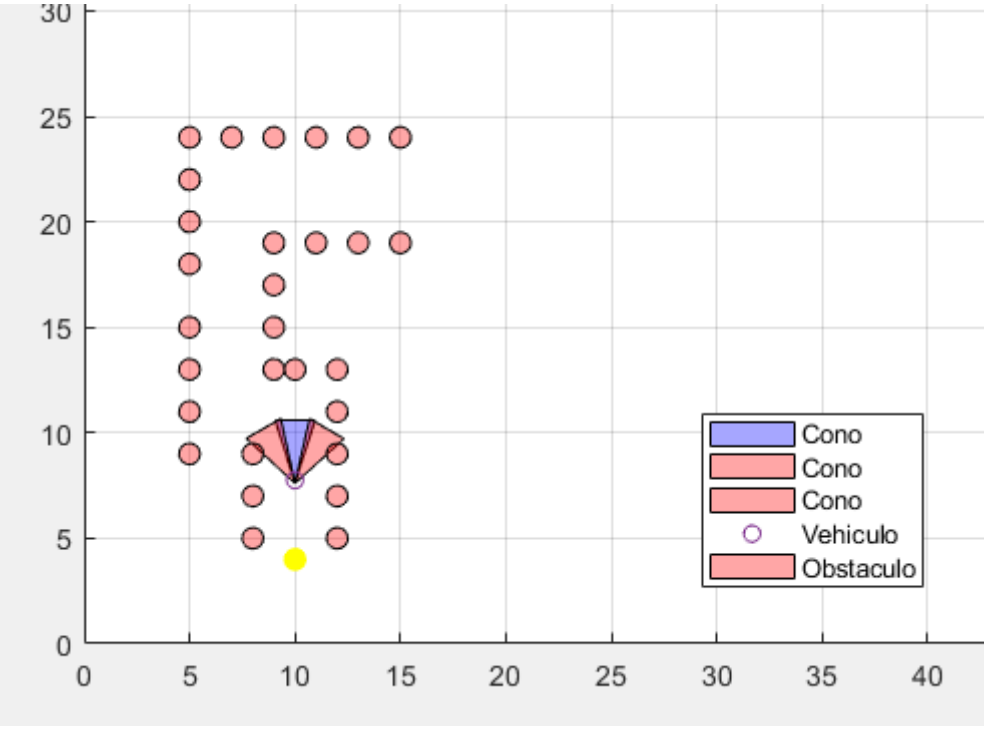
Implementación general

El programa está desarrollado en matlab. En el archivo enviado se encuentran distintas versiones.

- MySimAvoidObs1: Primera versión que probamos como idea base. Referencia https://www.mathworks.com/help/fuzzy/tune-fuzzy-systems-using-custom-cost-function.htm

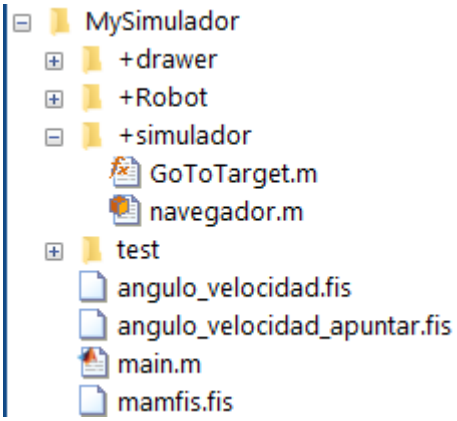


- MySimAvoidObs2: Prueba de laberinto sin target definido



- MySimReachTarget: Prueba de laberinto con target definido
- MySimReachTarget2: Prueba sin laberinto con target definido
- MySimReachTargetVelControl: Prueba con control de velocidad

dentro de la carpeta de cada versión se encuentran los siguientes archivos



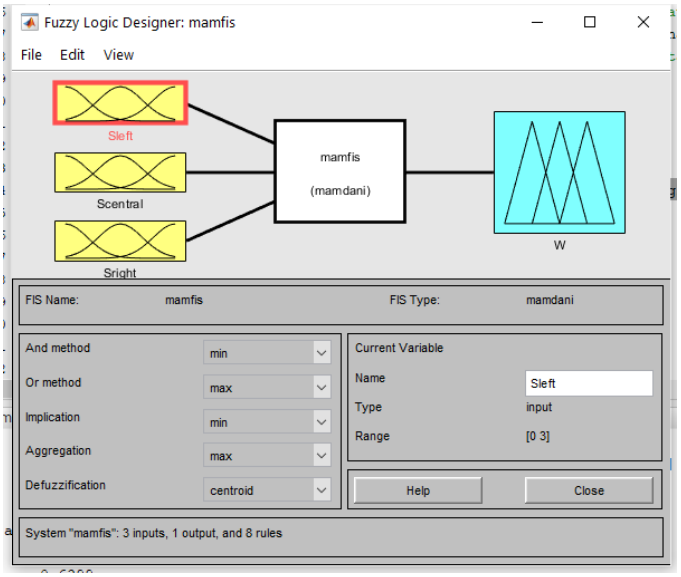
Desde main.m se ejecuta la simulación. Las distintas configuraciones y pruebas se pueden hacer desde el archivo navegador.m que se encuentra en la carpeta +simulador.

Dentro de navegador.m señalamos las líneas más relevantes para cambiar los parámetros de la simulación, Ejemplo:

```
% Nomre de archivo del Sistema de control difuso
nav.fis = readfis('angulo_velocidad_apuntar');

% Inicializo obstaculos para cada grafico
% posiciones [x y] de cada uno (-1 => ningun obstaculo en la simulacion)
nav.obstacles = [ [10 10] [9 10] [13 10] [15 12]];
% Posicion inicial del robot [x y angulo]
% El tercer argumento es la orientacion absoluta del robot al iniciar
nav.robot = [10 4 0];
% Posicion inicial del target
nav.target = [10 15];
```

En la función readfis se carga la configuración del control difuso. Este se puede modificar con la herramienta de lógica difusa de matlab (Introducir fuzzy en la terminal de Matlab).



Al generar una modificación se exporta con extensión fis, si se eligió otro nombre de archivo debe modificarse el nombre en la función readfis('nombre_archivo').

Las líneas siguientes definen las posiciones iniciales de los obstáculos, el robot y el target respectivamente. Para agregar obstáculos se agregan elementos [x y] al vector nav.obstacles.

Durante el trayecto el trayecto la evaluación del sistema difuso se realiza en la función:

evalfis(nav.fis,[distR_O,distC_O,distL_O],nav.options);

La salida de esta línea según la versión pueden ser distintas variables a controlar por el sistema de control difuso dependiendo en cada versión de los parámetros de la ecuación a controlar (Ver en "Diferencias entre los sistemas").

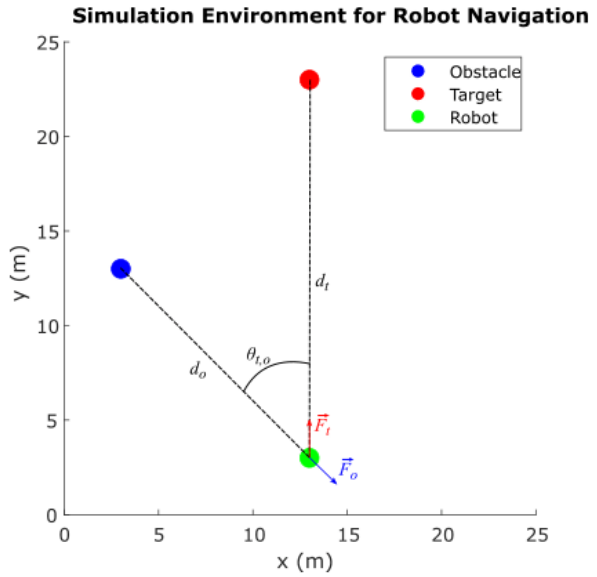
Más adelante se encuentran definidos los parámetros físicos de cada sensor, se puede modificar la apertura, el ángulo y la rotación del sensor respecto del eje hacia donde mira el robot.

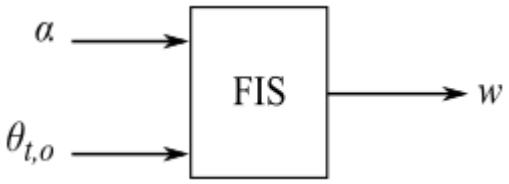
```
92
93          % Parametros fisicos del sensor central
94 -        nav.triangle1.ubicacion = [0,0];
95 -        nav.triangle1.apertura = 35/2;
96 -        nav.triangle1.largo = 3;
97 -        nav.triangle1.rotacion = 0;
98
99          % Parametros fisicos del sensor izquierdo
100 -       nav.triangle2.ubicacion = [0,0];
101 -       nav.triangle2.apertura = 35/2;
102 -       nav.triangle2.largo = 3;
103 -       nav.triangle2.rotacion = -10;
104
105         % Parametros fisicos del sensor derecho
106 -       nav.triangle3.ubicacion = [0,0];
107 -       nav.triangle3.apertura = 35/2;
108 -       nav.triangle3.largo = 3;
109 -       nav.triangle3.rotacion = +10;
```

Diferencia entre versiones

MySimAvoidObs1:
Primera versión que probamos como idea base.

Referencia
https://www.mathworks.com/help/fuzzy/tune-fuzzy-systems-using-custom-cost-function.htm





Ecuación a controlar con el sistema difuso:

$$\vec{F} = w\vec{F}_o + (1-w)\vec{F}_t, \text{ where } 0 \le w \le 1$$

La salida del sistema difuso es un parámetro llamado W el cual pesa entre las dos direcciones posibles del robot, la dirección al target y la evasión del obstáculo. Cuando la salida del sistema difuso es W=1 el robot se concentra en evadir el obstáculo y cuando la salida del sistema difuso es W=0 se dirige al target. Obviamente la salida del sistema difuso no es 1 o 0 si no una salida difusa en el rango [1 - 0].

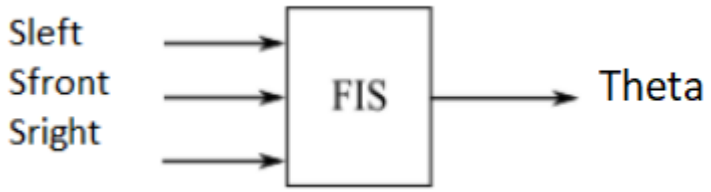
Las entradas al sistema de control difuso son:

- El ratio “α” que es el ratio de distancia. α = distancia entre robot-obstáculo/ distancia robot-target.
- Diferencia entre angular target-obstáculo θ

MySimAvoidObs2

Primera modificación de nuestra idea base, ahora nuestra ecuación a controlar con el sistema difuso es cuanto modificar el ángulo actual del robot para evadir un obstáculo.

Así que nuestro sistema de control difuso tiene estas entradas y salidas.



En donde Sleft, Sfront, y Sright representa las distancias al obstáculo y Theta el angulo que debera girar el robot para evadirlo.

Ecuación a controlar con el sistema difuso:

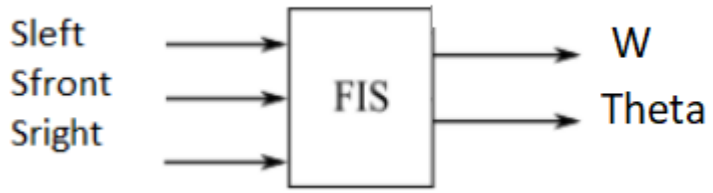
```
nav.robot(3) = nav.robot(3) - Theta;
```

Tal que:

- nav.robot(3) representa la dirección de avance actual del robot.
- Theta la salida del sistema difuso, evasión del obstáculo.

MySimReachTarget

En este caso la modificación consiste en agrupar ambas ideas de las versiones anteriores, la salida del sistema difuso ahora son dos, por un lado el ángulo “Theta” utilizado para corregir la dirección futura del robot para evadir el obstáculo y por el otro un parámetro llamado W el cual pesará la dirección al target y la evasión del obstáculo.



Ecuación a controlar con el sistema difuso:

```
nav.robot(3) = (1-w)*(nav.robot(3) - Theta) + w*(ang);
```

Tal que:

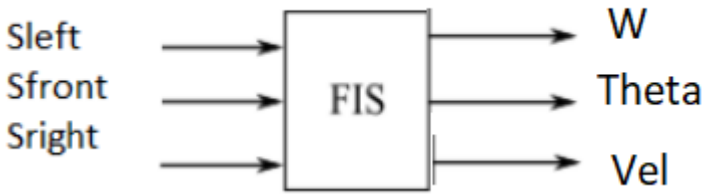
- nav.robot(3) representa la dirección de avance actual del robot.
- Theta la salida del sistema difuso, evasión del obstáculo.
- W: Parámetro [0-1].
- ang: ángulo al target con respecto a la horizontal en cada ubicación del robot.

MySimReachTarget2

Ejemplo con la misma ecuación que el anterior, sin laberinto y con distinto archivo fis, se agrega la variable velocidad a la salida del sistema difuso.

MySimReachTargetVelControl

En este caso lo que hicimos fue agregarle como salida al sistema difuso el control de la velocidad del robot.



En este caso nuestra ecuación a controlar la dirección del avance del robot no cambio pero, si cambia la velocidad del robot dinámicamente en función de la salida del sistema difuso llamada “Vel”.

Ecuación a controlar con el sistema difuso:

nav.robot(3) = (1-w)*(nav.robot(3) - Theta) + w*(ang);

Tal que:

- nav.robot(3) representa la dirección de avance actual del robot.
- Theta la salida del sistema difuso, evasión del obstáculo.
- W: Parámetro [0-1].
- ang: ángulo al target con respecto a la horizontal en cada ubicación del robot.

nav.Velocidad = evalout(1,3);