Sistemas de control inteligente

Práctica de laboratorio Final

Grado en Ingeniería Informática

Raúl López Llana
Oscar Pozo de Cádiz

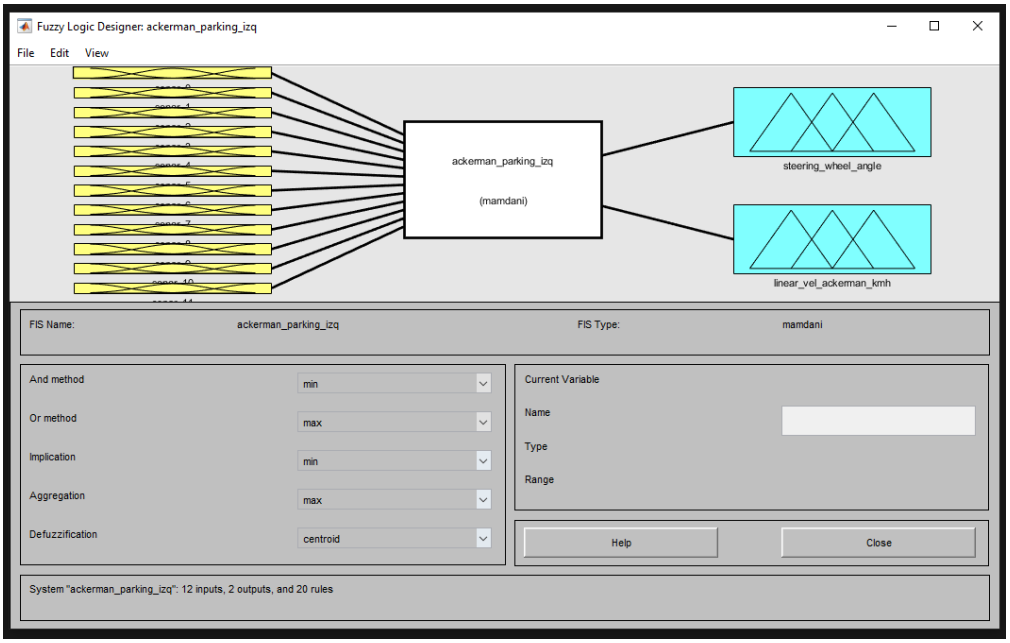
Contenido

1. Descripción de la práctica

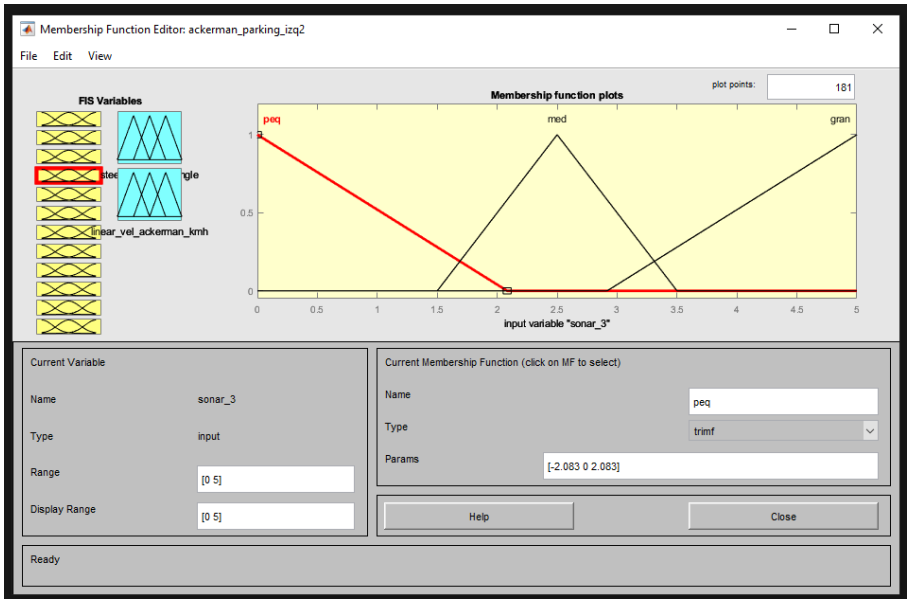
En esta práctica vamos a replicar el comportamiento que realizaría una persona para aparcar un vehículo en un hueco en paralelo. Esto lo realizaremos mediante 2 aproximaciones, con control borroso y con inteligencia artificial.

2. Control borroso

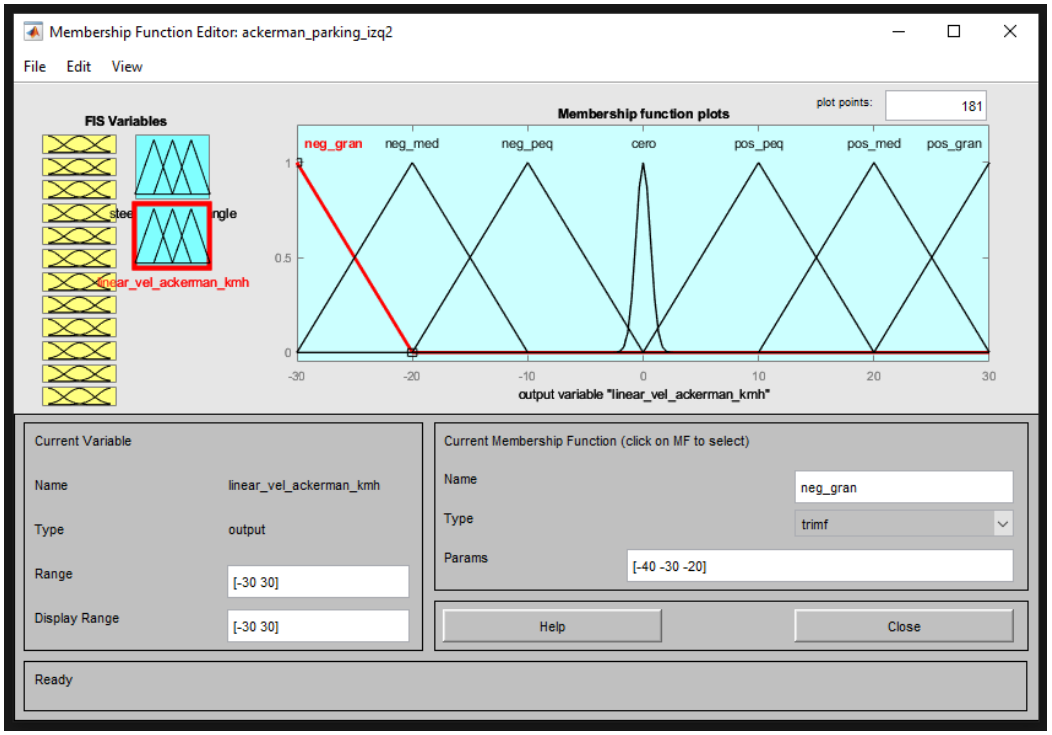
El vehículo que debemos controlar tiene instalados 12 sensores de proximidad con un alcance de hasta 5 metros. El número de sensores a utilizar dependerá de cada persona, pero en nuestro caso hemos decidido hacer uso de todos ellos desde el principio.



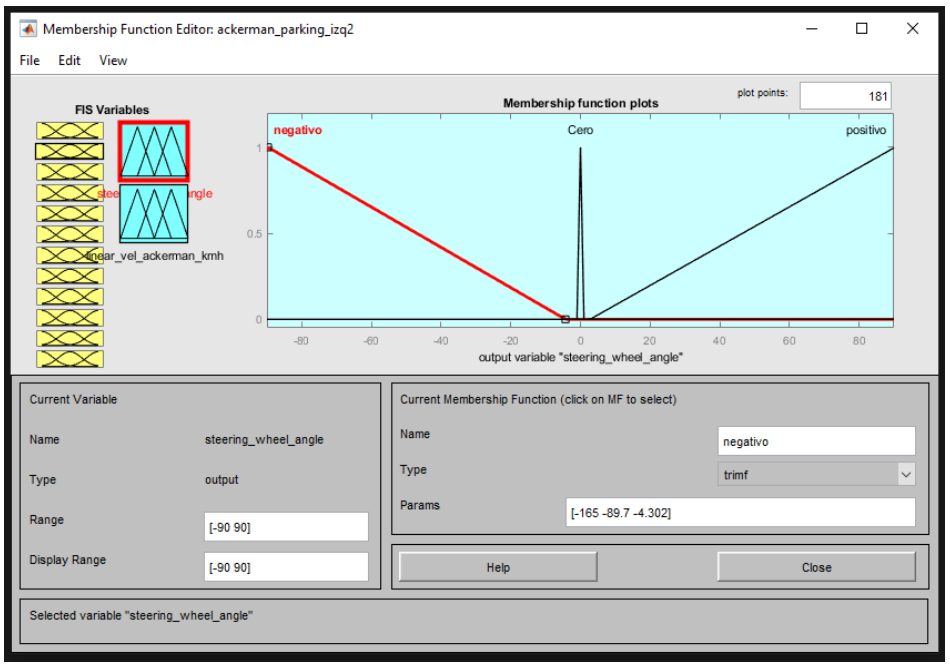
En cuanto a las funciones de activación de estos sensores comenzamos dividiendo su rango en 3 funciones como se puede observar en la siguiente imagen.



En cuanto a las funciones que afectan a la velocidad del vehículo decidimos crear 7, 3 para avanzar hacia delante y otras 3 para avanzar hacia detrás. Junto a esto una regla más para que se quede quieto el vehículo.
Con 3 reglas es suficiente para controlar la velocidad ya que a la hora de aparcar la velocidad no va a tener cambios muy drásticos de un momento a otro, sino constantes.

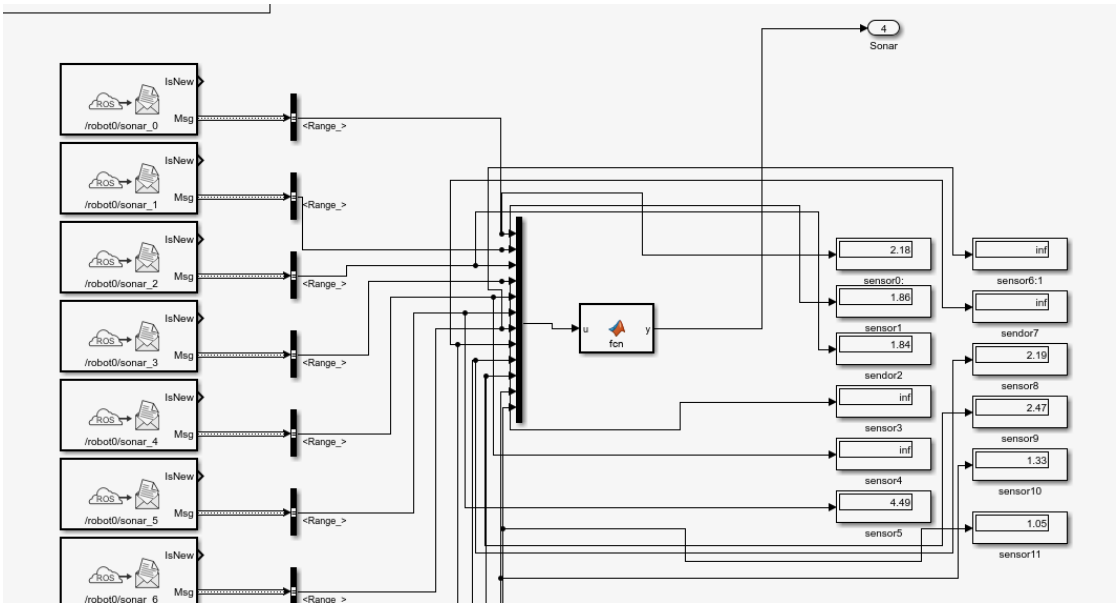


En cuanto al ángulo de giro, hemos decidido implementar 3 reglas muy sencillas. 2 para girar hacia un lado o al otro y una para que no gire y siga en línea recta.



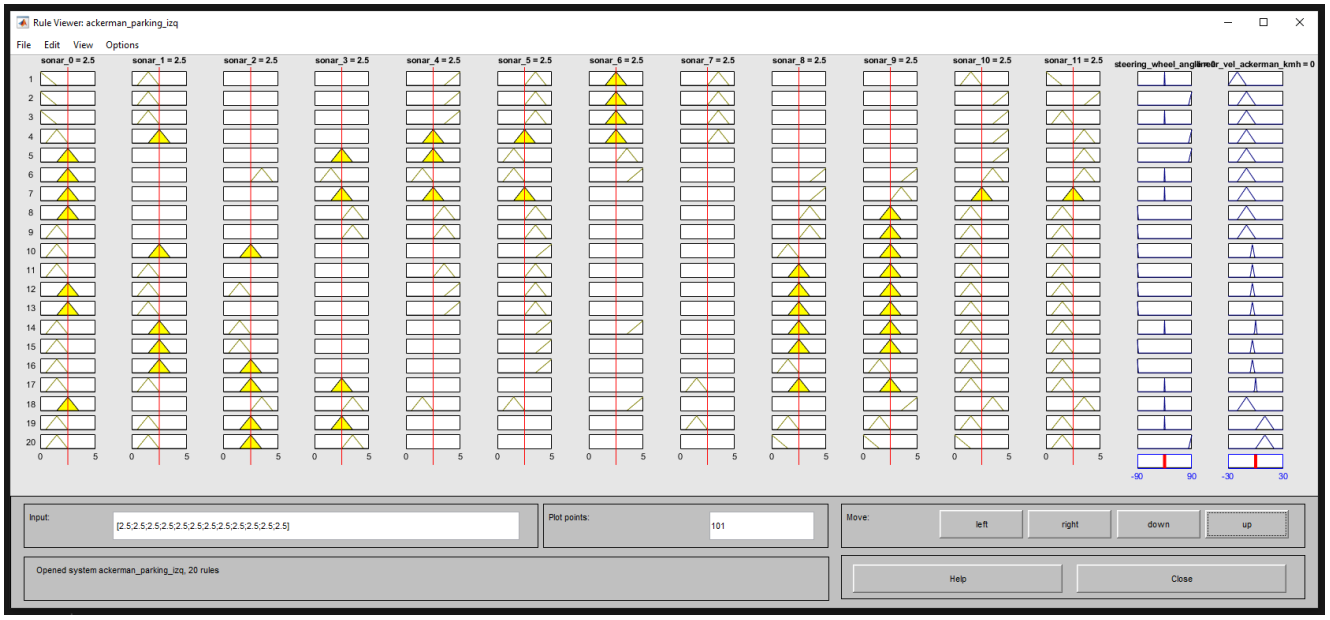
Con todo esto ya definido pudimos pasar a crear las reglas de activación del controlador.
Para esto nos ayudamos desde varios puntos. El primero fue añadir unos displays en cada uno de los

sensores para poder observar durante la ejecución el valor que leen cada uno de los sensores.



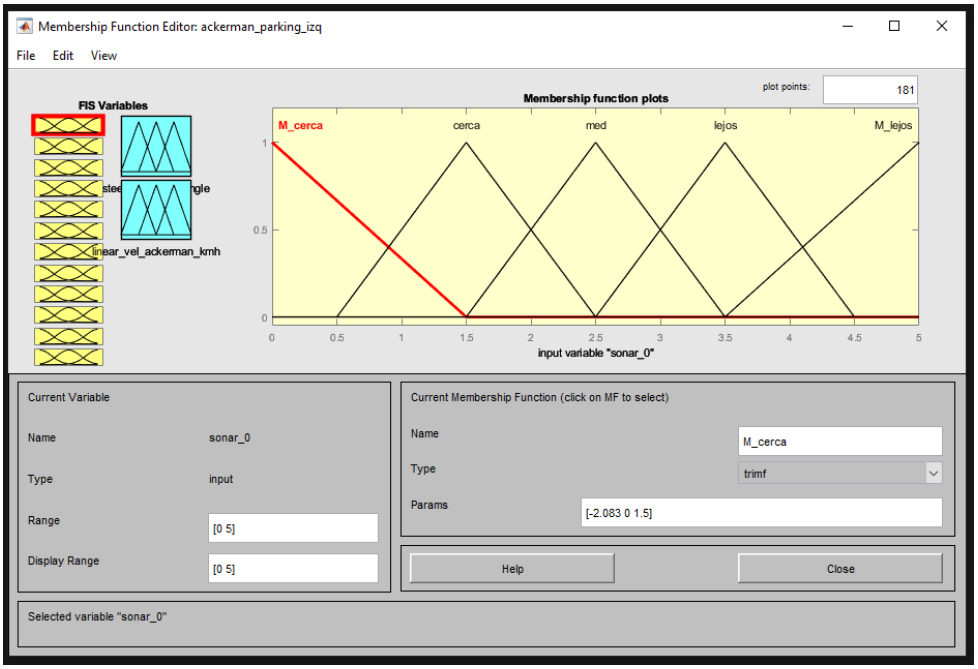
Gracias a esto podemos observar que valores tienen cada uno de los sensores en los momentos en los que se paraba el vehículo ya que no tenía ninguna instrucción que se pudiese ejecutar.

En este momento entra en juego el segundo punto que nos ayudó mucho, En el apartado de “view/Rules” fuimos capaces ver que gráficamente y de forma muy sencilla qué reglas tenemos que poner en esos puntos para que el vehículo continuase su movimiento.

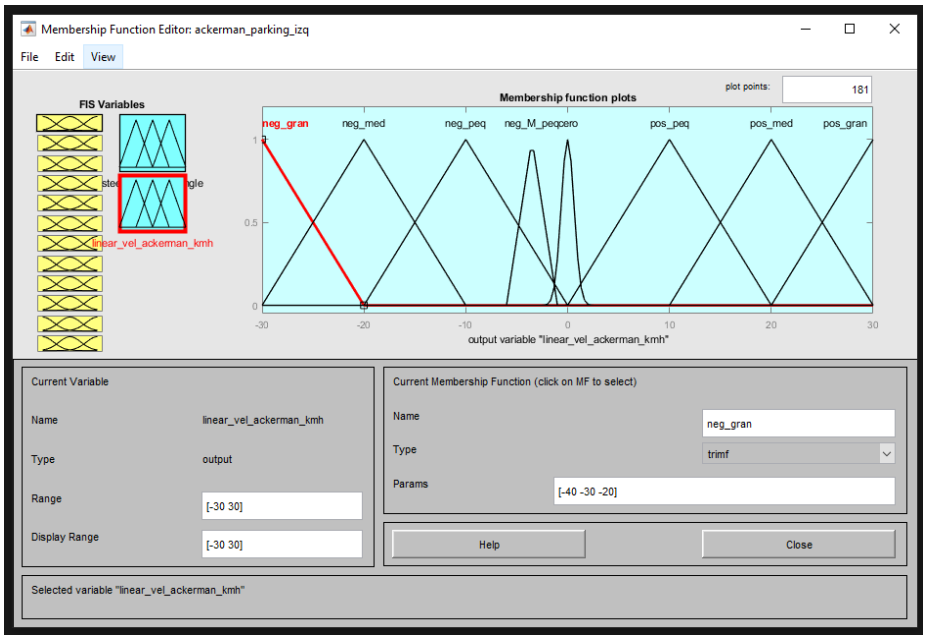


Observamos dónde quedaban las líneas rojas en cada uno de los sensores y ponemos una nueva regla para que en dichos puntos tuvieses alguna regla que hacer.

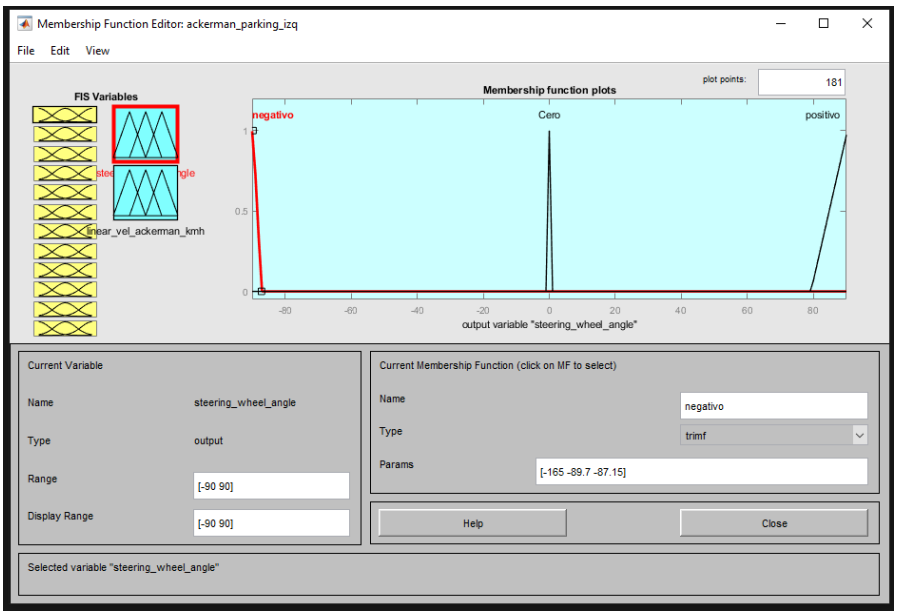
Tras observar los resultados obtenidos, vimos que al utilizar solo 3 funciones de activación en los sensores se nos quedaba corto y decidimos borrar todas las reglas del controlador, añadir 2 funciones más a cada sensor para obtener un total de 5 y volver a crear las reglas del controlador



Junto a este cambio también añadimos una nueva función para la velocidad para poder corregir el ángulo en el que termina el vehículo usando muy poca velocidad.



Y otro cambio que hicimos fue el acentuar los ángulos de giro para que fuesen más bruscos.



Con estos cambios vimos muy rápido la mejora en comparación con la anterior implementación.

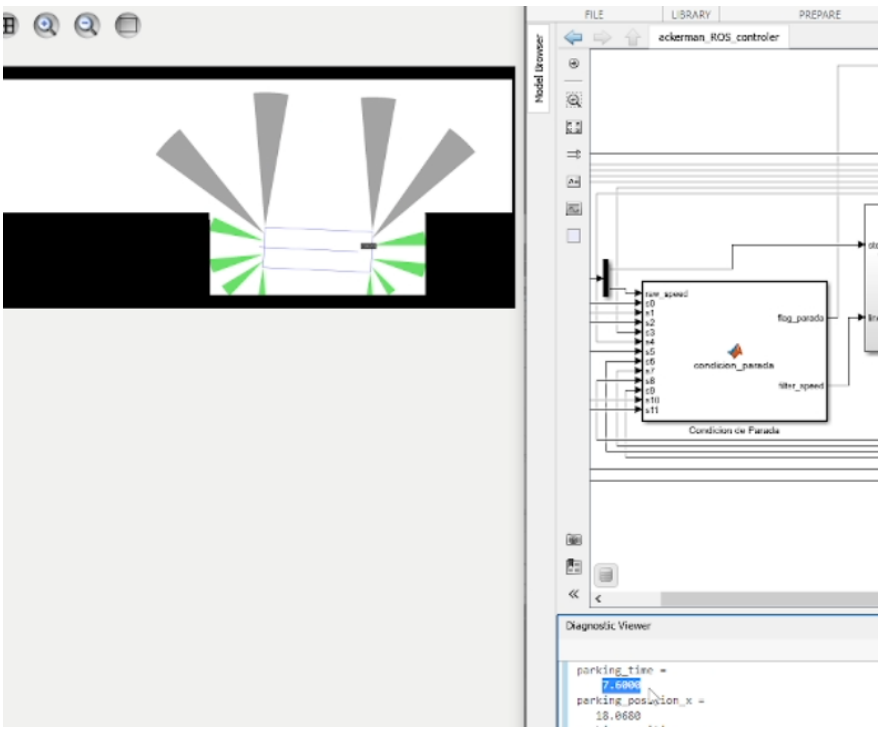
En cuanto a la condición de parada de la simulación hemos visto que en el momento que nosotros consideramos que la maniobra puede darse por finalizada, los cuatro sensores de arriba no están en contacto con ninguna superficie y el resto de la maniobra siempre tiene al menos uno de los sensores detectando un borde.
Por esto mismo hemos puesto que la condición de parada sea cuando estos cuatro sensores no nos arrojen información.

```
function [flag_parada, filter_speed] = condicion_parada(raw_speed, s0,s1,s2,s3,s4,s5,s6,s7,s8,s9,s10,s11)

% Condicion de parada:
% - debeis decidir cuando se finaliza la maniobra de aparcamiento en base
%  a la información disponible:
%       - velocidad_lineal
%       - ángulo de volante
%       - sensores
% Una vez que decidais cual es el criterio de parada debeis:
%    - Parar el vehículo con v_lineal = 0.
%    - Activar el flag_parada a 1 para indicar que ha finalizado la maniobra.

if (s4 == 5 && s5 == 5 && s6 == 5 && s7 == 5) % Ejemplo de condición de parada (abs(raw_speed) < 0.09)

    %Es necesario parar el vehículo con v_lineal = 0 y activar el
    %flag_parada a 1 para indicar que ha finalizado la maniobra.

    %filter_speed = raw_speed;
    %flag_parada=0;
    filter_speed = 0.0;
    flag_parada=1;

else
    filter_speed = raw_speed;
    flag_parada=0;
end
```

Resultados

Como resultado de aplicar el controlador borroso creado obtenemos que el aparcamiento tarda en completarse entre 6 a 9 segundos. Esto varía ya que, a la hora de ejecutar el mismo programa con el mismo punto de comienzo, no siempre lee los mismos valores ya que no se hace una lectura instantánea y continua de todos los sensores. Existe un pequeño retraso que ocasiona lecturas distintas

de valores.



Como se puede observar en la Imagen superior, la maniobra de aparcamiento en un lanzamiento aleatorio tarda en realizar el aparcamiento 7,6 segundos.

Consideramos que el resultado es bueno en su gran mayoría y que se podría llegar a optimizar y mejorar en sus últimos movimientos.

3. Control Neuronal

En este apartado volveremos a crear un controlador que sea capaz de realizar la maniobra de aparcamiento, pero esta vez usaremos una red neuronal para implementarlo. Partiremos del mismo robot y entorno para hacerlo.

Para poder entrenar el controlador neuronal es fundamental disponer de un conjunto de datos. Para este apartado los datos pueden incluir las distancias detectadas por los sensores y las velocidades lineal y angular.

Nuestro primer enfoque consistió en intentar usar los datos generados por el controlador borroso. Sin embargo, fuimos incapaces de entrenar correctamente la red neuronal con estos datos: el vehículo se queda atascado en su posición inicial, alternando constantemente entre ir hacia delante y hacia atrás. Probamos con numerosas variaciones de la red neuronal, cambiando el número de neuronas, número de sensores de entrada y funciones de entrenamiento, pero el problema seguía ocurriendo. Sospechamos que puede deberse a que en la fase final de la maniobra del controlador borroso el vehículo cambia de marcha para ir hacia delante, lo cual podría confundir a la red neuronal.
Es por ello que llegamos a la conclusión que deberíamos entrenar la red neuronal con una trayectoria en la que el robot avance siempre hacia la derecha.

Para generar los datos de entrenamientos optamos por usar el control manual. Para que la trayectoria nos fuese útil tenía que cumplir dos requisitos:

- Como bien se comentó anteriormente, el vehículo debe moverse siempre a la derecha, para evitar confundir a la red neuronal.

- La trayectoria debe mantenerse siempre razonablemente alejada de las paredes. Al principio tratamos de usar una trayectoria muy ajustada que se acercaba mucho a las paredes, pero la trayectoria de la red neuronal siempre de desvía ligeramente, por lo que siempre acababa chocándose. Al mantener la trayectoria alejada de las paredes, hacemos que esas ligeras desviaciones no supongan en un accidente fatal.

Una vez que tuvimos registrada una trayectoria correcta, comenzamos el proceso de entrenamiento. Tuvimos que realizar muchas iteraciones, cambiando los parámetros de la red numerosas veces, hasta que obtuvimos un resultado satisfactorio. Algunas consideraciones y observaciones de este proceso:

- Rellenamos la matriz de datos de entrenamiento introduciendo 3 veces los datos de la trayectoria, para aumentar artificialmente la cantidad de datos de entrenamiento.

- En un principio probamos varios tipos de redes, como las redes '*fitnet'* o las redes '*feedforward'*. Nos decantamos por las redes '*feedforward'*, ya que proporcionaban unos resultados mejores.

- Cambiar el método de entrenamiento de la red alimentada hacia delante por el *algoritmo de propagación hacia atrás BFGS cuasi-Newton* también mejoró notablemente los resultados de los entrenamientos.

- Establecer la complejidad de la red (mediante su número de neuronas) también es un paso importante. Al principio probamos con un número de neuronas relativamente alto, de entre 12 y 15, ya que esa es más o menos la cantidad de entradas de la red. Sin embargo, los resultados del aprendizaje no eran buenos. Después probamos a reducir el número de neuronas, pero si el número era demasiado bajo (3 ó 4, por ejemplo) la red era incapaz de aprender y seguía siempre una trayectoria prácticamente recta, sin realizar ningún giro brusco. Al final nos decantamos por un valor intermedio de 7 neuronas en la capa

oculta, que proporcionaba los mejores resultados. También probamos a introducir varias capas ocultas, pero esto no mejoraba los resultados por lo que decidimos continuar con una única capa oculta.

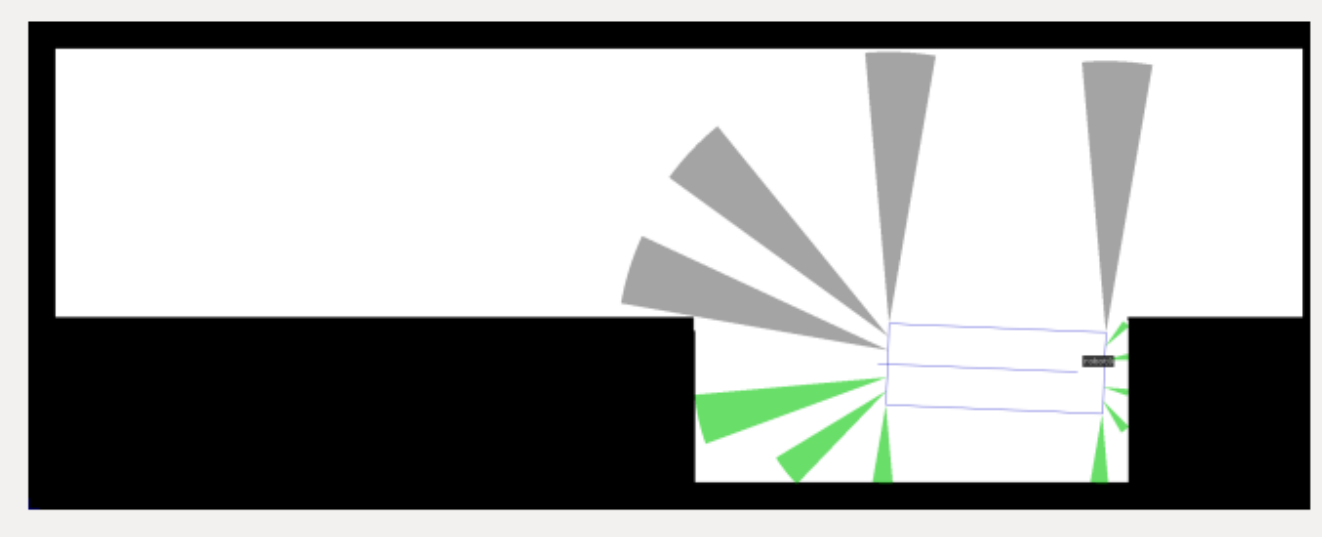
- Incluso con todos los pasos anteriores, el proceso de entrenamiento era bastante irregular: entrenar dos veces la misma red con los mismos datos producía dos trayectorias completamente distintas. Es más, algunas de estas trayectorias ni siquiera se parecían a la trayectoria original de los datos de entrenamiento. Esto es debido a que los pesos iniciales son aleatorios cada vez que se entrena la red. El hecho de que cada entrenamiento daba un resultado completamente distinto nos hizo pensar que el entrenamiento no se estaba completando correctamente. También pensamos eso ya que los entrenamientos siempre terminaban porque llegaban al límite de epochs y no porque lograra un aprendizaje óptimo. Para mejorar esto, aumentamos el límite de epochs de 1000 a 5000. Esto mejoró los resultados e hizo que los resultados de los entrenamientos fueran más uniformes.

- Como última observación, al introducir la red neuronal en el controlador, debemos convertir todos los datos de entrada al tipo double:

Para este apartado también modificamos la condición de parada. Para que el vehículo

Resultados

El controlador neuronal es capaz de aparcar, como se muestra en la imagen, en unos 24 segundos. Este aparcamiento se realizó en el entorno final.



Debido a que no se hace una lectura instantánea y continua de todos los sensores (como ya se comentó en el apartado del controlador borroso), es posible encontrar resultados distintos en ejecuciones distintas, pero todos son muy parecidos.