Texto

Descripción generada automáticamente

PRÁCTICA FINAL

SISTEMAS DE CONTROL INTELIGENTE

Diseño de Controladores Borrosos y Neuronales para la maniobra de aparcamiento de un vehículo

Grado en Ingeniería Informática

David Barreiro Zambrano

Natalia Montoya Gómez

*ÍNDICE*

[*Instalación y configuración del entorno* 3](#_Toc156100359)

[Desarrollo de la práctica 8](#_Toc156100360)

[PARTE I. Diseño manual de un control borroso de tipo MAMDANI 8](#_Toc156100361)

[PARTE II. Diseño automático de un controlador neuronal 14](#_Toc156100362)

[Conclusiones 15](#_Toc156100363)

# *Instalación y configuración del entorno*

A continuación, se muestra el procedimiento que se ha tomado para poner a punto la máquina, en cuanto a instalación y configuración de la misma, para un correcto funcionamiento del entorno.

En primer lugar, se descarga el fichero *.*ova, ubicado en el espacio del aula virtual de la asignatura. Esta máquina viene preparada por los profesores de la asignatura con los programas necesarios para el desarrollo de esta práctica final:



Una vez descargado el fichero que contiene la imagen de la máquina virtual, desde VirtualBox, se importa de tal manera que solo es necesario determinar la política de dirección de MAC, la cual es: *Generar una nueva dirección MAC para todos los adaptadores de red*, tal y como aparece en la siguiente figura:

Captura de pantalla de computadora

Descripción generada automáticamente

Presionamos el botón de *Terminar* y la máquina empieza a importarse a nuestro equipo:

Captura de pantalla de un celular

Descripción generada automáticamente

Una vez instalada la máquina, el nombre de usuario y contraseña es *alumno*. Con lo cual, podremos entrar a la máquina, pero antes de ello, es necesario configurar algunos detalles más.

En la ventana de *VirtualBox*, se clica en *Configuración* y se cambia, en caso de que tuviese otra configuración, a la que aparece en la figura siguiente, *Adaptador puente*, lo cual permite que haya una conexión con el host principal del equipo, Internet y otras máquinas virtuales. Con esto se consigue que la información viaje desde la máquina virtual hacia el host y viceversa, para ejecutar *Matlab* mientras en la máquina se ejecuta *ROS*.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Aprovechamos que estamos con esta ventana abierta, para generar distintas direcciones MAC por si hubiese problemas con alguna conexión. Por lo que, clicamos varias veces sobre las flechas azules que aparecen al lado de la dirección MAC.

Estando dentro de la máquina, se abre una terminal, desde donde lanzaremos los siguientes comandos y podremos acceder a ciertas carpetas:

Texto

Descripción generada automáticamente

Añadimos las siguientes líneas de código al fichero .*bashrc*:



Como últimos pasos, y para tener el entorno preparado, se mueven los archivos descargados desde el aula virtual en la carpeta de descargables de la asignatura, donde se almacenan los archivos necesarios para STDR, entre los que se encontrarán el mapa, el fichero *.launch* y el *.xml*.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

El fichero *.launch*, el cual contiene la posición donde estará posicionado el robot en el mapa, está ubicado en la carpeta *stdr\_launchers/launch*. En la cual, para cambiar al entorno definitivo de la práctica, se necesitan cambiar los parámetros que se señalan en amarillo en la figura siguiente:

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Los ficheros que configuran el mapa en ROS, son el *.png* y el *.yaml* mostrados en la siguiente figura, los cuales están ubicados en la carpeta de *stdr\_resources/maps*.

Imagen que contiene Interfaz de usuario gráfica

Descripción generada automáticamente

El fichero que contiene las especificaciones del robot, tales como las posiciones de los sónares, y otras características, están en el fichero *.xml*, ubicado en la carpeta *stdr\_resources/resources/robots*.

Escala de tiempo

Descripción generada automáticamente con confianza media

Una vez ubicados todos los ficheros en sus respectivas carpetas, se lanza la siguiente instrucción para poder abrir el programa, con el robot ubicado en la posición en la que ha sido especificado, y el mapa correspondiente.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Al ejecutar la instrucción anterior, la interfaz del simulador de STDR con ROS se abre, y desde la terminal, podemos observar que se encuentra en espera de algún mandato para poder empezar a mover el robot.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente

Por último, y con Matlab abierto en nuestro host, podemos hacer una comprobación de que máquina y host están conectados, dando al botón de *Test* con el cual se abrirá una ventana emergente en la que comunicará si ha habido éxito en la conexión o no.

Interfaz de usuario gráfica, Texto, Aplicación, Correo electrónico

Descripción generada automáticamente

Es probable que el *Firewall* esté activado, por lo que deberá dar permisos a su versión de Matlab en redes públicas y privadas.

Con este último paso, termina la instalación y configuración del sistema.

A continuación, se describe el desarrollo de la práctica.

# *Desarrollo de la práctica*

## PARTE I. Diseño manual de un control borroso de tipo MAMDANI

Se comienza con el diseño del controlador borroso, para ello, abrimos la consola de Matlab y ejecutamos la instrucción *fuzzy* tal y como se especifica en el enunciado de la práctica. Con esto, podemos crear un controlador borroso básico inicial. Este comando permite abrir la ventana emergente de Fuzzy Logic Design, desde donde se controlan las entradas y salidas, funciones de pertenencia y reglas de un sistema de control borroso.

En nuestro caso, a modo de idea para englobar el problema que se plantea, hemos tomado la ayuda del Control Manual que se proporcionaba en los ficheros de ayuda descargables. De esta manera, hemos ido haciéndonos una idea de los sensores que eran o no necesarios para este modelo de problema.

En la siguiente figura se visualiza el robot con todos los sensores numerados:

4

Diagrama

Descripción generada automáticamente con confianza baja

8

9

10

11

7

6

5

0

1

2

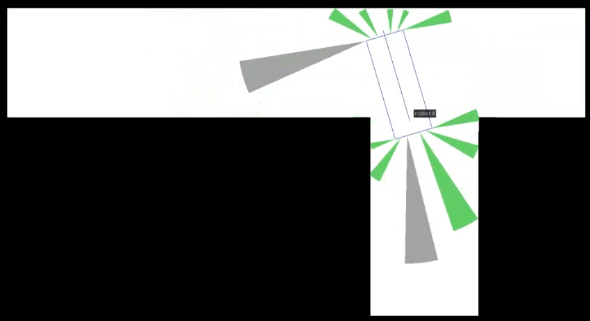
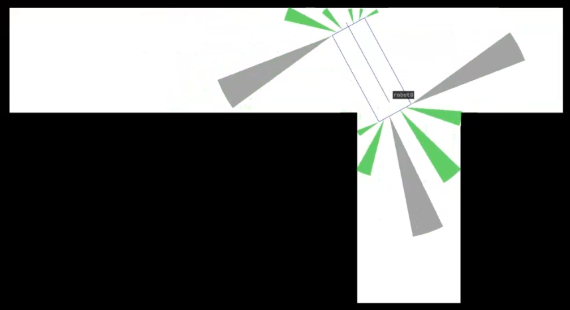
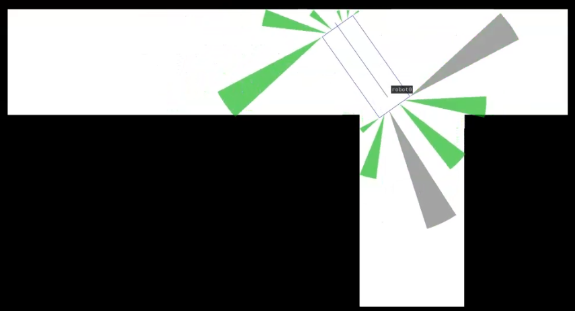
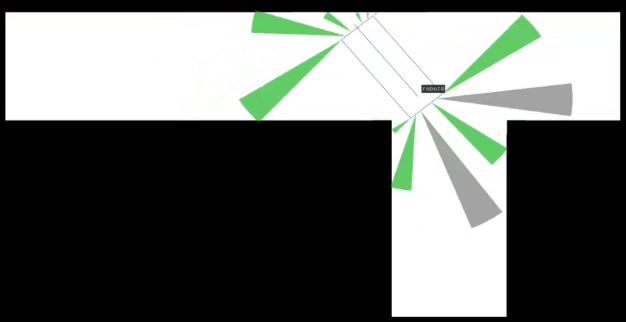
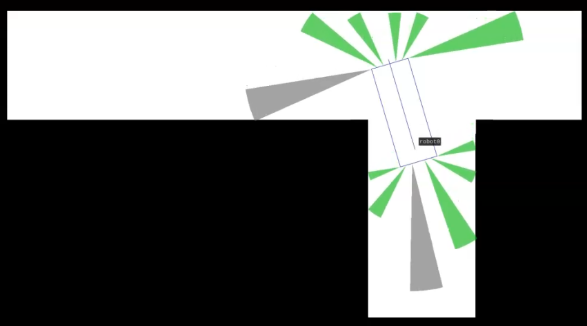
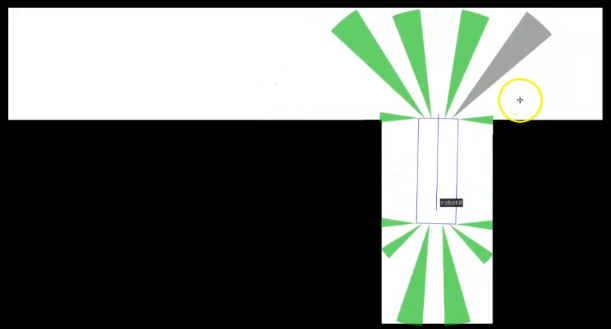
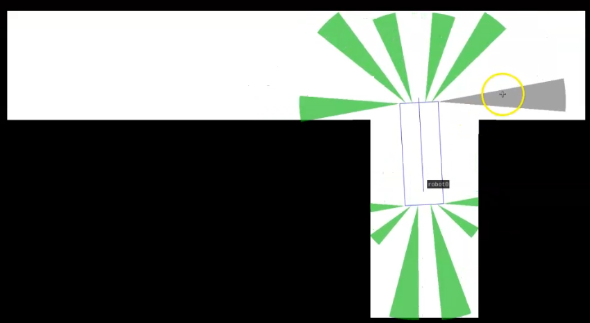
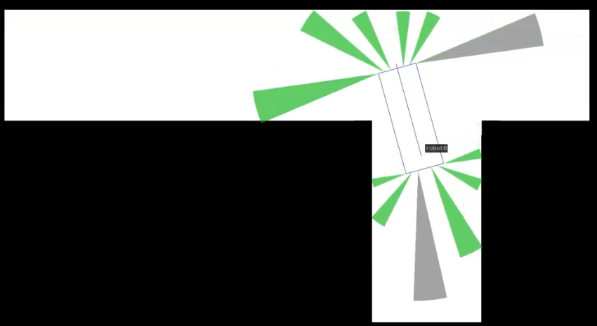
3

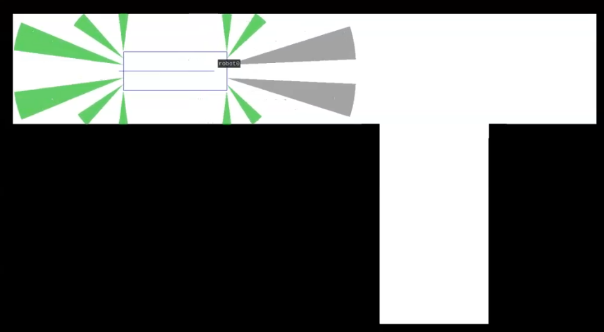
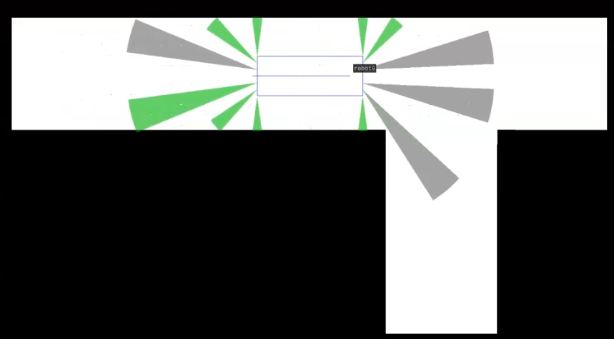
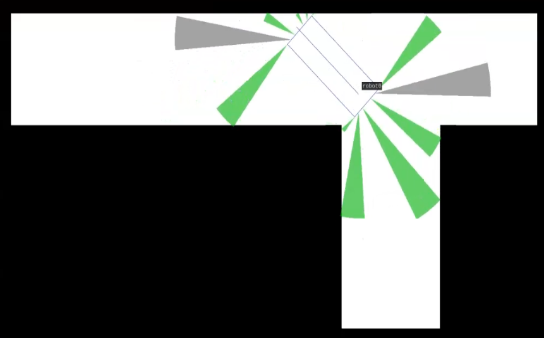
Tras varias maniobras de aparcamiento con el controlador manual ‘ControlManual\_Ackerman\_12\_sensores\_V\_steering’ hemos llegado a la conclusión que solo es necesario usar 6 sensores para que la maniobra solicitada se ejecute de forma correcta.

Notamos que cuando ciertos sensores tienen como entrada su máxima distancia (cercano a 5 o superior) nos sirven para referenciar el giro del robot, junto con los que tienen entradas más pequeñas, para el aparcamiento.

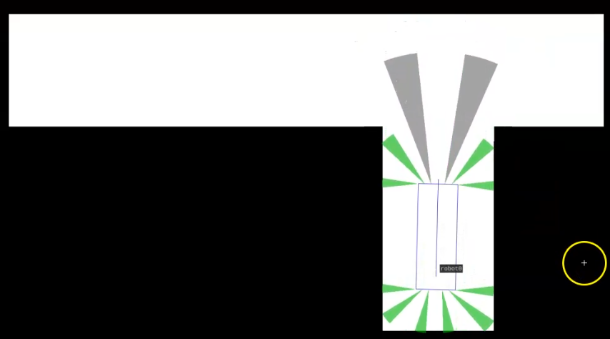
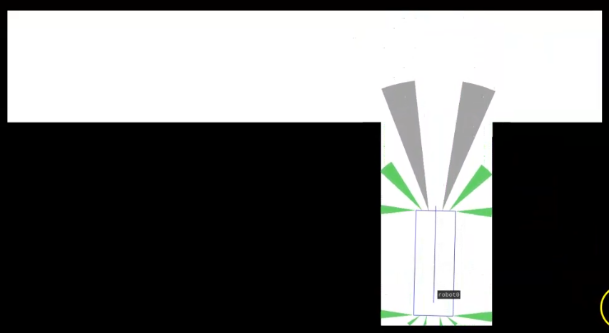
Asi como para saber si el robot debe seguir moviéndose sin hacer ningún giro. Po ejemplo cuando empieza a moverse y se va acercando a la zona de aparcamiento o bien cuando ya esta colocado dentro de la misma con la orientación correcta.

A continuación, demostraremos gráficamente el estudio sobre el que se ha basado el analisis de la maniobra y la decisión del uso de los sensores, tomando los mínimos e imprescindibles que aporten información. Con puntos amarillos denotamos los sensores que tienen una distancia menor a 5 y con puntos rojos los sensores que tienen una distancia superior o próxima a 5.

Dibujo en blanco y negro

Descripción generada automáticamente con confianza baja 







De los mostrados en la imagen, como hemos comentado, solo usamos 6 de ellos, ya que son los que entendemos necesarios para la maniobra. Estos son el 5, 6, 7, 8, 10 y 11:

7

6

5

Imagen que contiene Diagrama

Descripción generada automáticamente

11

10

8

El robot está orientado hacia atrás, por lo que el sensor 8 y 10 son los sensores que tiene en la parte trasera. De esta forma, la plaza de aparcamiento se encuentra a su izquierda, con lo que hemos considerado que solo serán necesarios los sensores que se mostraban en la imagen anterior.

Estos sónares se ven reflejados como entradas en el sistema de control borroso que se ha creado para resolver este problema, los cuales se pueden visualizar en la siguiente figura. El sistema tiene como salidas, la velocidad y el ángulo.

Diagrama

Descripción generada automáticamente

Se puede observar que las funciones de pertenencia de los sónares son las mismas para todos ellos, puesto que hemos considerado dos estados, la distancia grande y la distancia pequeña. Cuando se trata de una distancia grande, se entiende que el sónar no ve ningún obstáculo en una distancia cercana a 5m. Sin embargo, se considera que una distancia es pequeña cuando empieza a ser menor de 4.5m.

Esta figura es aplicable para todos los sónares, pues tienen los mismos parámetros:

Interfaz de usuario gráfica

Descripción generada automáticamente

Cuando se trata de las funciones del ángulo de giro del volante y de la velocidad del robot, las funciones de pertenencia serán distintas.

En cuanto a la velocidad, tenemos 5 funciones.

DriveG: El robot va hacia delante con velocidad superior a 2.5.

DriveP: El robot va hacia delante con una velocidad entre 0 y 5.

Cero: El robot toma una velocidad entre -2.5 y 2.5.

ReverseP: El robot va marcha atrás con velocidad entre 0 y -5.

ReverseG: El robot va marcha atrás con velocidad superior a -2.5.

Pero lo más predominante será el uso de la función de reversa, ya que nuestro robot solo maniobra marcha atrás durante toda la maniobra, pues según como está posicionado, la forma más rápida de realizar el aparcamiento es de esta manera y girando el ángulo perfecto como para no tener que moverse hacia delante durante dicha maniobra.

Interfaz de usuario gráfica

Descripción generada automáticamente

Además, hemos visto que los mas eficiente es que lleve una velocidad pequeña hacia atrás (ReverseP) para que pueda reconocer y reaccionar a tiempo a las entradas de los sensores.

En cuanto al ángulo, tenemos 4 funciones:

GiroIzq: El robot hace un giro entre superior a 0 hasta -90.

GiroCero: El robot hace un giro entre -5 y 5.

GiroDer80: El robot hace un giro entre 0 y 90 con su valor máximo en 80.

GiroDer90: El robot hace un giro entre 80 y 90.

Interfaz de usuario gráfica, Gráfico de líneas

Descripción generada automáticamente

La función más usada es la de giro de 90 grados, pues en el momento en el que el sónar 10 tiene una distancia grande, debería hacer un giro radical de 90 grados mientras se va moviendo Exceptuando, cuando el sonar 5 y el 8 tienen una distancia grande que es el momento de corregir el robot con un angulo de 80 grados para que el robot se coloque correctamente en la plaza de aparcamiento.

El giro Cero es utilizado cuando el robot no tiene que girar, simplemente avanzar en una dirección recta. Es decir, cuando inicia la maniobra o cuando el robot esta ubicado en la zona de aparcamiento con la orientación adecuada.

Imagen que contiene Tabla

Descripción generada automáticamenteUna vez definidos el rango, los parámetros de las variables de entradas y salidas, y habiendo creado las funciones de pertenencia, se diseñan las reglas del controlador borroso. Para establecer las reglas, solo hemos definido las reglas inferidas del análisis inicial.

1ª REGLA: Todos los sónares tienen distancia pequeña, exceptuando el sónar 8 que tiene una distancia grande, significa que el robot debe ir avanzando en reversa sin realizar ningún giro.

Logotipo, nombre de la empresa

Descripción generada automáticamente



2ª REGLA: Todos los sónares tienen distancia pequeña exceptuando los sónares 8 y 10, quiere decir que el robot debería empezar a girar, con un ángulo de 90, pues la plaza de aparcamiento está próxima.

Imagen que contiene Diagrama

Descripción generada automáticamente



3ª REGLA: Todas las distancias son pequeñas, excepto 7 y 10, significa que tiene posibilidad de giro aún, debe seguir girando mientras sigue avanzando.

Imagen que contiene Forma

Descripción generada automáticamente



4ª REGLA: Debe seguir girando un poco más, todos los sónares tienen distancia pequeña excepto el 7, lo que significa que falta muy poco espacio de giro.

Forma

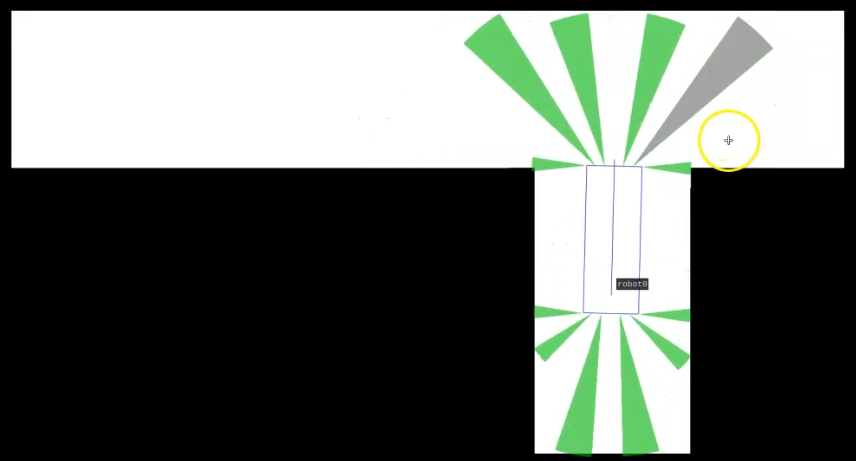
Descripción generada automáticamente con confianza media

A continuación, se agruparán las siguientes reglas: 6ª, 7ª, 8ª ,9ª,10ª,12ª, puesto que el robot en estas reglas necesita seguir girando. Hasta el momento en que el sónar 5 tiene una distancia grande junto al sonar 8 (regla 11ª), es el momento de aplicar un giro más suave cercano a 80.

Imagen que contiene Icono

Descripción generada automáticamente



Finalmente, el robot vuelve a tener los 6 sensores con una distancia pequeña, por ello es momento de aplicar la 5ª regla. Es decir, aplicar un GiroCero, muy pequeño o cercano a cero.

El robot seguirá avanzando hasta que el flag de parada se ponga a 1 con los criterios que explicaremos más adelante.

Diagrama

Descripción generada automáticamenteFinalmente, el sistema de Simulink queda de la siguiente manera:

Tomamos 6 sensores del ROS Robot (s05,s06,s07,s08,s10,11). Añadimos a Controlador el controlador borroso.

Texto

Descripción generada automáticamenteSe puede observar una condición de parada que conecta con el bloque de ROS Robot, que a su vez conecta con un bloque de evaluación de maniobra especificado por los profesores de la asignatura, el cual no debe modificarse, y, por último, un multiplexor que acaba en el controlador visto anteriormente.

La condición de parada está pensada para que cuando el sónar 8 tenga un campo de visión menor a 0.5 y el giro del volante sea menor a 8, sabremos entonces que el robot está cercano a la pared de la plaza de aparcamiento y lleva un giro muy pequeño, por lo que en ese momento se efectuaría la parada.

Texto

Descripción generada automáticamente

Por último, en la presentación de la práctica, se puede ver el funcionamiento de este sistema borroso de tipo Mamdani, con los entornos definitivo y original.

## PARTE II. Diseño automático de un controlador neuronal

Esta segunda parte de la práctica final se trata de solventar el mismo problema anterior, pero utilizando una red neuronal, la cual es necesaria de entrenar para su correcto funcionamiento.

Para el entrenamiento de la red neuronal hemos optado por la opcion de usar nuestro controlador borroso, ya que ejecuta correctamente la maniobra para los dos entornos proporcionados.

Para ello, creamos el script en Matlab ‘Entrenamiento’, que se apoya en el

1. script de Matlab maniobra\_park\_ackerman\_datos\_entrenamiento\_alumnos.m (incluido en los archivos de la práctica)

Para la ejecución del control manual, tal y como se especifica en el enunciado, ha sido necesario hacer, en la máquina virtual, un *ifconfig,* de tal manera que conozcamos la IP del ordenador, para poder conectarlo a Matlab y lo necesario.

Texto

Descripción generada automáticamente

De esta manera, la variable ROS\_MASTER\_IP tiene que ser igual a la IP señalada en la imagen anterior en amarillo.

Tabla

Descripción generada automáticamente

Antes de comenzar con la generación de datos y el entrenamiento de la red neuronal, se establece la conexión con el entorno ROS. Se limpian las variables existentes, y se apaga cualquier instancia ROS previa.

A continuación, se inicia la conexión ROS con la dirección IP '192.168.1.33. Se declaran variables globales para almacenar datos relevantes de los sónares y el control del robot.

Después de realizar la conexión, se inicializan los suscriptores y publicadores ROS, así como se generan mensajes para el control de velocidad del robot. Se espera hasta recibir un mensaje relacionado con el robot para asegurar una inicialización adecuada.

Texto

Descripción generada automáticamente

Se crea un bucle que realiza maniobras de aparcamiento simuladas. Utilizando N como numero de iteraciones sobre los entornos.

Interfaz de usuario gráfica, Texto

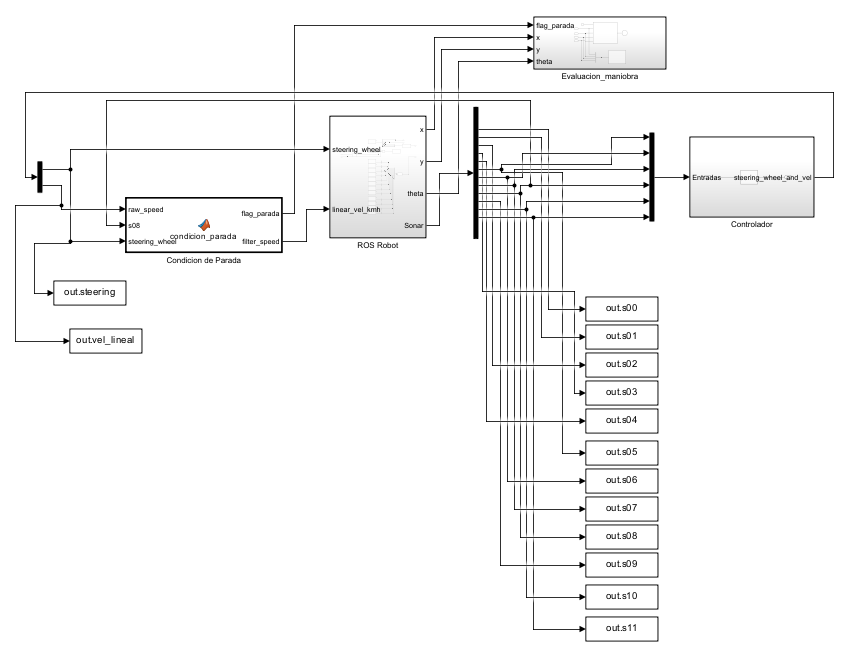
Descripción generada automáticamente con confianza media

Y declaramos las matrices que irán almacenando los datos en cada una de ellas.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media

Empezamos por lanzar nuestro controlador borroso implementado en Simulink y con las modificaciones necesarias para capturar los valores de salida necesarios, como podemos ver en la siguiente imagen.



Los datos de velocidad lineal, ángulo de dirección y lecturas de sensores son recopilados y almacenados en matrices.

Interfaz de usuario gráfica, Texto

Descripción generada automáticamente con confianza media

Luego haremos una pausa de 10 segundos para reiniciar el robot con el mapa deseado y preparar la siguiente iteracion en ROS.

Finalmente se acotan las matrices de datos, puesto que se producen pequeñas variaciones en el tiempo de la maniobra al capturarlas y esto nos podria dar una o dos filas mas. Generando un problema en el momento de concatenarlas con las matrices creadas anteriormente.

Seguidamente, concatenamos los matrices con las de las iteraciones anteriores y empezamos con la siguiente iteracion.

Los datos recopilados se utilizan para entrenar una red neuronal de alimentación directa con 8 neuronas en la capa oculta. La red se configura, entrena y luego se genera un bloque de Simulink para implementar el control neuronal. Está entrenada con 10 maniobras, 5 en cada entorno, original y definitivo, de manera alterna.

Texto

Descripción generada automáticamente

Este código en MATLAB se utiliza para conectar con un entorno ROS, inicializar un simulador de robot Ackerman, generar datos de entrenamiento a través de simulaciones de aparcamiento, y finalmente entrenar una red neuronal con esos datos.

En la siguiente figura podemos ver la información relacionada con Neuronal Network y Network Diagram:

Interfaz de usuario gráfica

Descripción generada automáticamente

En la siguiente imagen se muestra la información de Performance, Training State, Error Histogram y Regression:

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamenteInterfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

Finalmente, sustituimos nuestro controlador borroso por la red neuronal generada. Para ellos usamos como entrada todos los sensores y el bloque “*Data Type Conversion*” entre la salida del multiplexor de los ultrasonidos y la entrada de la red.

De este modo nos queda el siguiente sistema:

Diagrama

Descripción generada automáticamente

# *Conclusiones*

Como conclusiones se puede concluir que, el controlador borroso ofrece mejores resultados ante circuitos que se conocen perfectamente las funciones de pertenencia, por lo que supera al controlador neuronal.

Por otra parte, la eficacia del controlador neuronal depende considerablemente de la calidad de los datos de entrenamiento.

Otra conclusión que hemos sacado es que a la hora de entrenar el sistema, funcionaba mejor cuando intercalábamos entre un mapa y otro, el original y el definitivo.