Manual de usuario de CyBeRSim

José Manuel Cuadra Troncoso
Departamento de Inteligencia Artificial, UNED

Version 1.0.0

Resumen CyBeRSim es un simulador 2D de robots de código abierto con características especiales para el estudio de controles basados en redes neuronales, aunque no restringido a este tipo de controles. CyBeRSim posee una arquitectura muy flexible de manera que resulta cómodo de desarrollar y extender, el investigador o estudiante puede desarrollar su modelo de robot, de sensores, de algoritmo de control, red neuronal, etc. El uso del programa se realiza mediante GUIs desde las que se pueden editar todos aspectos de la simulación. Cuenta con un sistema muy completo de monitorización de fácil uso y proporciona herramientas para la publicación de las investigaciones.

Desarrolladores

José Manuel Cuadra Troncoso desarrollador inicial y director del proyecto.

María Dolores Gómez Tamayo ampliación del módulo de redes neuronales e implemetación de opciones de edición.

Índice general

Capítulo 1

Introducción

El estudio de la Inteligencia Artificial, y de la Robótica Autónoma en particular, atrae a investigadores y estudiantes provinientes de muy distintos campos de la Ciencia como pueden ser: la Informática, la Ingeniería, la Matemática, la Medicina, la Sicología, etc. Dada la diversidad de campos de procedencia así son de diversas lo son las habilidades adquiridas. En el campo del estudio de controles de robots autónomos suele ser necesaria cierta habilidad para programar ordenadores y ésta no se adquiere sino en contadas carreras, debido a lo cual muchas de las personas interesadas se ven con grandes, o incluso insalvables, dificultades para llevar a cabo su experimentación. Esta experimentación suele llevarse a cabo en muchas ocasiones en simuladores por problemas de costes, disponibilidad y desgaste de los robots reales.

El objetivo de suavizar en lo posible las diferencias entre los iniciados y los no iniciados en las técnicas de programación se enfrenta también al problema de la existencia de diferentes sistemas operativos: mientras el usuario se halla acostumbrado a entornos MSWindows, el programador de software libre suele preferir Linux. La mención de software libre nos lleva a otro aspecto a tener en cuenta: los costes, un simulador comercial de grandes prestaciones puede tener un precio elevado.

Por último en nuestro país hay otro aspecto a considerar: los simuladores, tanto comerciales como libres, no suelen crearse en países de habla hispana y, aunque se puedan realizar traducciones, los usuarios y programadores deben comunicarse con los creadores o los mantenedores de dichos simuladores en inglés, lo cual puede dificultar dicha comunicación.

CyBerSim surgió como una herramienta para desarrollar los estudios de Doctorado del autor. Primero como herramienta de diseño de redes neuronales sencillas para el simulador [11] de Khepera [6] de Olivier Michel. Más tarde se construyó el simulador a partir de cero y se introdujeron nuevos tipos de neuronas. Pero en un punto de su desarrollo fue necesario ampliar su arquitectura y se propuso la posibilidad de abordar las cuestiones comentadas en el párrafo anterior.

Actualmente CyBerSim continúa su desarrollo como herramienta de investigación en este departamento y se decidió extraer una versión estable del programa destinada a los estudiantes que cursan estudios de grado y posgrado. Esta versión educativa es la que se presenta aquí. Las principales características de esta versión son las siguientes:

- En primera instancia es un simulador 2D, aunque se piensa pasar a 3D posteriormente, los mundos son creados y editados por un programa de terceras partes.
- Sólo se puede simular un robot a la vez, aunque en breve será multirobot.
- El uso del lenguaje XML para los archivos de definición de los distintos elementos que componen el programa, salvo los mundos, permite una gran flexibilidad a la hora de construir dichos elementos.
- El simulador no está limitado a ningún tipo de robot-2D, ni a ningún tipo de sensores. En próxima versiones será posible modelar robots y sensores desde una GUI, por ahora se pueden editar manualmente los archivos correspondientes, ver [3].

- Aunque soporta cualquier tipo de controles está originalmente pensado para los basados en redes neuronales, disponiendo de varias herramientas específicas para el diseño de redes.
- Dispone de completas GUIs para la edición y control de los elementos que componen la simulación, haciendo más intuitiva la labor del usuario.
- Implementación de dinámica sencilla.
- Grandes capacidades de monitorización gráfica y grabación a ficheros no sólo del robot y sensores, sino de los elementos que componen el control, lo que favorece el trabajo del usuario
- Multiplataforma, la misma versión para MSWindows y Linux.
- Programación en C++, usando el framework de programación Qt [18].
- Documentación en castellano, [3, 2] y este manual, extensa tanto para el usuario como para el programador, se va traduciendo al inglés.
- Sólida arquitectura para extender el programa y reusar elementos.
- Código fuente bajo licencia pública.

Acompañan al programa algunos ejemplos de vehículos de Braitenberg [1] comentados en la sec. 5 para el robot Khepera.

* * * * *

Convenciones

Las cadenas delimitadas por paréntesis angulados <> deben sustituirse por la expresión que corresponda en cada caso, p. ej., una trayectoria de directorios, un fichero, etc.

Los elementos de menú y de las barras de herramientas suelen aparecer en negrita.

Capítulo 2

Instalación

2.1. Cómo obtener CyBeRSim

CyBeRSim se puede obtener tanto en código fuente multiplataforma o ejecutables de MSWindows.

2.2. Instalación de ejecutables MSWindows

Descomprimir el archivo .zip así se instalan el programa, las librerías necesarias y la documentación.

2.3. Instalación del código fuente

El código fuente se divide en tres partes: el programa cybersim-edu y las librerías floatspinplugin y qewextensibledialogs. La instalación, compilación del código fuente y ejecucción del programa han sido comprobados en varias versiones de las distribuciones de Linux: Fedora, Suse, Debian y Ubuntu, así como en MSWindows-XP.

2.3.1. Requisitos

Los requisitos para la instalación, compilación y ejecucción no dependen del sistema operativo en el que se vaya a compilar.

2.3.1.1. Requisitos necesarios

- Es necesario tener instalado un compilador de C++ que se soportado por Qt [19]. En Linux ha sido comprobada la compilación con varias versiones de las series 3 y 4 de GCC [5] y en MSWindows con las últimas versiones de MinGW [4], la versión de GCC para MSWindows. GCC se incluye en las distribuciones habituales de Linux y suele instalarse por defecto.

- Es necesaria la librería Qt [18] versión Qt-3 [17] y sus archivos de cabecera, aunque es preferible la subversión 3.3. Las distribuciones habituales de Linux suelen incluir paquetes con dicha librería y de la siguiente versión, la 4, aunque no las instalan por defecto, y suelen configurar por defecto el sistema para usar la versión 4, lo que deberá cambiarse. Los archivos de cabecera no se suelen instalar por defecto, en ese caso hay que instalar los paquetes de desarrollo de Qt (devel). En MSWindows se puede descargar el paquete fuente de, preferiblemente, la última subversión de Qt-3 win [8]. La versión 3 de Qt de código abierto no está soportada por TrollTech sino por KDE [7].

2.3.1.2. Requisitos opcionales

- Xfig [21] es un programa para CAD sencillo, los mundos en CyBeRSim-edu se crean y editan en Xfig y el programa usa el formato Xfig 3.2 para la exportación de gráficos. En MSWindows se denomina WinFIG [20], aunque tiene ciertas limitaciones.
- Qt Designer es el diseñador de GUI de Qt. Es necesario para desarrolladores no para usuarios finales.
- LaTeX (TeX) [9] es un sistema de procesamiento de textos, especialmente científicos y técnicos, de alta calidad. Se incluye en las distribuciones de Linux. En MSWindows de denomina MiKTeX (TeX) [12].
- Un cómodo editor de LaTeX es LyX [10].

2.3.2. Instalación y compilación

La instalación y compilación de CyBeRSim-edu se lleva a cabo en varios pasos:

1. **Variable de entorno `QTDIR`.**
   Como paso previo es necesario definir la variable de entorno `QTDIR para que` contenga la ruta a las Qt-3. Esta operación depende de sistema operativo:
   - **Linux.** La ruta a las Qt-3 suele ser `/usr/lib/qt*` ó `/usr/share/qt*`. Para comprobar si la variable `QTDIR` está definida se usa el comando de consola (el caracter `%` representa al símbolo del sistema, no se teclea):
     `% echo $QTDIR`
     Si no obtenemos salida deberemos darle valor a la variable de entorno:
     `% export QTDIR=<ruta-a-las-Qt3>`
     La compilación se debe realizar en la ventana donde se ejecutó el comando `export`. O incluir el comando anterior en el fichero `.bashrc` o en `.bash_profile`, que se encuentran (ocultos) en el directorio principal del usuario, y la variable de entorno se econtrará siempre disponible. Dependiendo de la distribución de Linux las variables de entorno de usuario pueden pasar o no al entorno de root, cuando se usan los comandos `su` o `sudo`, si hay problemas consultar la documentación de la distribución sobre que comando utilizar `su` o `sudo`.
   - **MSWindows-XP.** La ruta a las Qt-3 suele ser `C:\(Archivos de programa)\Qt\qt-3`. Para comprobar si la variable `QTDIR` está definida se usa el comando de consola (el caracter `%` representa al símbolo del sistema, no se teclea):
     `% echo $QTDIR`
     Si no obtenemos salida deberemos darle valor a la variable de entorno:
     MiPC ▷ **Propiedades** ▷ **Opciones avanzadas** ▷ **Variables de entorno** y establecemos `QTDIR=<ruta-a-las-Qt3>`.
2. **Floatspinplugin.**
   Descomprimir el archivo .tar (Linux) o .zip (MSWindows) y ejecutar los siguientes comandos en consola:
   a) `% qmake`
   b) `% make`
   c) `% make install` , en Linux este comando como root, con `su` o `sudo`.
3. **QewExtensibleDialogs.**
   Descomprimir el archivo .tar (Linux) o .zip (MSWindows) y ejecutar el siguiente comando en consola:

   a) `% ./qew.build` , y seguir las instrucciones del script, en Linux será necesaria la clave de root.

4. **CyBeRSim-edu.**
   Descomprimir el archivo .tar (Linux) o .zip (MSWindows) y ejecutar los siguientes comandos en consola:

   a) `% qmake`

   b) `% make`

La instalación del programa es local y la de las librerías global.

CyBeRSim está disponible también en inglés, para usar esta versión sólo hay que eliminar el archivo `neuraldis.es.qm` (traducción al español) del directorio `translations` del programa tras la compilación, para volver a la versión española hay que restituir dicho archivo desde el archivo comprimido de la distribución. La traducción al español está casi completa.

Capítulo 3

Uso

3.1. Iniciando CyBeRSim

Para iniciar CyBeRSim nos situamos en el directorio de instalación de la distribución y tecleamos desde consola el comando:

`% ./cybersim-edu` , (en MSWindows sin `./`)

Se puede enlanzar `cybersim-edu (cybersim-edu.bat` en MSWindows) en el escritorio, en el directorio `.../src/imagenes` están los iconos necesarios.

La primera vez que se utiliza CyBeRSim aparece un cuadro de diálogo para la configuración del programa, tiene dos pestañas:

- **General**
  - **Directorio de la distribución**: directorio donde se ha instalado la distribución (por defecto el directorio de inicio del usuario), se puede editar manualmente o elegirlo mediante el botón situado a la derecha, mediante el que se accede al oportuno cuadro de diálogo.
  - **Visor de ayuda** y **Comando de visor**: trabajan conjuntamente, en el primero elegimos el programa con el que visionaremos el manual en formato HTML y en el segundo podemos editar el comando que lo arranca si el que aparece por defecto en cada caso no nos conviene. Si el comando no está en los directorios especificados en el path del sistema se debe especificar la ruta absoluta (completa) al comando, si se intenta dejar en blanco aparece el comando por defecto. También se elegir mediante el botón situado a la derecha.
    El visor integrado en la aplicación es un sencillo visor de limitadas capacidades: no establece conexiones a Internet ni muestra imágenes, está pensado para usuarios que por alguna razón no dispongan de navegador.
- **Directorios de trabajo**. Los directorios de trabajo para guardar redes neuronales, mundos, simulaciones, robots y controles se pueden elegir dentro de esta pestaña. Por defecto son los contenidos en el directorio `.../examples` del programa.

Una vez establecida dicha configuración aparece la ventana del programa propiamente dicha.

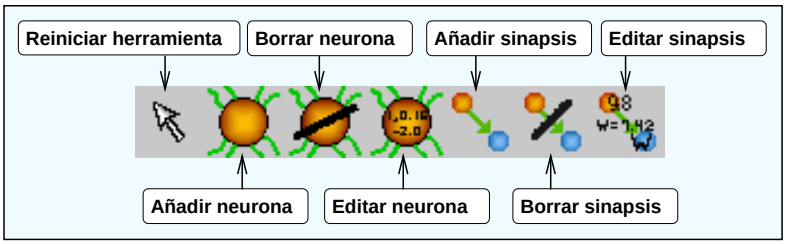
3.2. Descripción de los elementos del menú y barras de herramientas.

Hay elementos de los menús deshabilitados ya que su función está en desarrollo.

3.2.1. Las barras de herramientas

3.2.1.1. La barra del diseñador

Ver explicación de cada herramienta en 3.2.5.



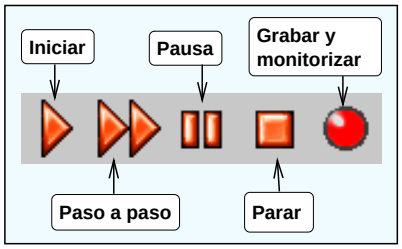
Barra de herramientas del diseñador de redes

Selección/deselección de neuronas Hacer click sobre la neurona para seleccionarla (su icono se iluminará). haciendo click el fondo se deslecciona. Todas las herramientas, salvo **Añadir Neurona**, deseleccionan las neuronas seleccionadas (iluminadas en la interfaz) para una operación si se hace click en un punto que no pertenezca al cuadrado que circunscribe a alguna neurona, en cualquier caso haciendo click cerca del borde de la interfaz o cambiando de herramienta se deselecciona lo que haya seleccionado.

Selección/deselección de sinapsis Para seleccionar o crear una sinapsis debemos hacer click sobre la neurona de la que sale o saldrá la sinapsis y Ctrl-click sobre la neurona a la que llega o llegará. Si se hace click sobre otra neurona cambia la neurona de la que sale la sinapsis pero no cambia la de llegada y viceversa con Ctrl-click, haciendo click sobre el fondo se deseleccionan las dos.

3.2.1.2. La barra del simulador

Ver explicación de cada herramienta en 3.2.6.



Barra de herramientas del simulador

3.2.2. El menú archivo

Los distintos elementos del menú **Archivo** realizan las operaciones habituales de abrir, cerrar, guardar ficheros de red neuronal, salir del programa, etc. La opción de submenú Imprimir no ha sido implementada. A parte de las opciones comentadas encontramos

- **Exportar red** nos permite exportar la red neuronal actual a formato Xfig 3.2 y posteriormente editarla con este programa. También es posible exportarla a Pdf pero es neceserio disponer de la utilidad `fig2dev (transfig)` que se incluye en las distribuciones de Linux o se puede descargar desde las páginas de WinFIG.

- **Exportar trayectoria** nos permite exportar un archivo de datos de trayectoria de un robot a formato Xfig 3.2. Se selecciona el fichero de datos a convertir y luego el nombre para el archivo `.fig` (hay que cambiar la extensión). La trayectoria exportada puede ser incluida dentro del mundo donde se creó con la operación `merge` de Xfig para su posterior publicación.
- **Test XML** comprueba la estructura de un fichero `.xml`, sólo para desarrolladores.

3.2.3. El menú editar

- En el menú **Editar** están implementados los elementos típicos: **Cortar**, **Copiar**, **Pegar**, **Deshacer** y **Rehacer**, para operaciones sobre la red neuronal y además **Editar Preferencias**, que muestra el cuadro de diálogo descrito en la sección 3.1 de este manual y **Guardar Preferencias**.

3.2.4. El menú vista

Los elementos del menú vista permiten mostrar u ocultar herramientas e información.

- La **Barra de tareas**: operaciones habituales con ficheros de red.
- La **Barra de estado**: mensajes en la parte inferior de la ventana.
- La **Barra de diseñador**: herramientas de edición de redes neuronales. (Ver la sección 3.2.1.1).
- **Mostrar sensores**: muestra u oculta el campo de los sensores que tengan implementada dicha posibilidad.
- **Mostrar leyenda de pesos**: muestra u oculta una pequeña ventana con la información relativa al valor de peso correspondiente al color con que se dibuja la sinapsis.
- **Mostrar fichero de texto puro**: muestra el fichero de red neuronal en formato texto. (Sólo es necesario para desarrolladores)
- **Mostrar vista de red**: permite seleccionar un área de la red neuronal para verla en una nueva ventana.

3.2.5. El menú diseñador

El menú del **Diseñador** contiene las herramientas para crear y editar redes neuronales. Las más usuales se hallan también en la **Barra del diseñador**, siendo el manejo del programa más completo, cómodo e intuitivo desde dicha barra. Se describirán ambas formas de utilización conjuntamente, aunque se recomienda el uso exclusivo de la **Barra del diseñador**. Esta barra incluye un icono en forma de flecha para deseleccionar herramientas, si una herramienta seleccionada no funcionase pulsar sobre dicha flecha y volver a seleccionar la herramienta.

3.2.5.1. Construir una red neuronal

Tanto la opción de este menú 3.2.5.3 como la herramienta **Añadir Neurona** de la **Barra del diseñador** usadas en una interfaz vacía también realizan esta operación.

Si tenemos cargada una red nos mostrará un mensaje de advertencia, ya que se borrará la red previa, cuidado. A continuación aparece un cuadro de diálogo para que introduzcamos el número de neuronas de cada capa. La configuración que aparece por defecto es la que requiere Khepera, con los 8 sensores de proximidad y los 8 de luz y posibilidades de control de la velocidad adelante y atrás por parte del usuario.

La **descripción de las distintas capas** es la siguiente:

- **Entrada**: Las neuronas de entrada disponen de un campo donde se elige la identificación del generador de entradas (sensor) conectado a la neurona. Los identificadores son:
  - 1 a 8 para los 8 sensores de proximidad (infrarrojos) empezando en el trasero del lado izquierdo y siguiendo, en el sentido de las agujas del reloj, hasta el trasero del lado derecho.
  - 9 a 16 para los 8 sensores de luz de la misma manera que antes.
  - 17, 18, 19 para los sensores de colisión izquierdo, trasero y derecho (no aparecen por defecto).
- **Oculta**: Las que aparecen por defecto sirven para mandar comandos de velocidad al robot. Una vez contruida la red eatas neuronas deben ser editadas y establecidas sus subcapas, ver lista más abajo, y posteriormente conectadas las neuronas de salida con sinapsis de peso = 1.0. El fichero `.../examples/nets/estandar.net` contiene la red por defecto, se puede abrir y usar como base, así nos ahorramos lo anterior. Se pueden eliminar las neuronas de movimiento hacia atrás tipos 254 (`estandar.net` 23), 255 (24) si no se piensan enviar órdenes de movimiento hacia atrás. A partir de aquí hay que, posiblemente, hay que añadir más ocultas, conectar las de entradas, etc. para diseñar nuestra red para el robot.
  - oculta 17 subcapa: 252 (velocidad rueda izquierda adelante) a salida 21,
  - oculta 18 subcapa: 253 (velocidad rueda izquierdo atrás) a salida 22,
  - oculta 19 subcapa: 254 (velocidad rueda derecho atrás) a salida 23,
  - oculta 20 subcapa: 255 (velocidad rueda derecho adelante) a salida 24.
- **Salida**: las neuronas de salida mandan su valor de salida a los motores de manera proporcional al parámetro de velocidad máxima del robot. De izquierda a derecha (neuronas 21, 22, 23, 24 en `estandar.net`) motor izquierdo adelante, izquierdo atrás, derecho atrás, derecho adelante (derecho, izquierdo desde el punto de "vista" del robot que es el mismo que el del usuario). El número de neuronas de salida debe ser 4 para un correcto funcionamiento de robots con dos ruedas motrices como Khepera.

3.2.5.2. Borrar red neuronal

Limpia la interfaz, si tenemos cargado un red nos mostrará un mensaje de advertencia ya que se borrará la red previa, cuidado. No hay una herramienta en la **Barra del diseñador** para esta función.

3.2.5.3. Añadir neurona

Equivale a la herramienta de la **Barra del diseñador** ▷ **Añadir neurona**.

Usadas en una interfaz vacía crean una red neuronal como en 3.2.5.1. En una interfaz no vacía si la usamos desde el menú aparece un cuadro en el que se pregunta por la capa y el número de posición de la neurona a añadir. **No usar este elemento de menú, le falta la elección de tipo, sino la herramienta**. Usada como herramienta añade una neurona entre las neuronas y en la capa donde hagamos click con el ratón, podemos hacer click antes de la primera o después de la última neurona de una capa. Nos pregunta primero por la tipo de neurona a añadir (doble click elige). A continuación aparece el cuadro de edición comentado en 3.2.5.5. Las tipos de neurona disponibles son:

- **Neurona** para dispositivos de umbral.
- **Neurona de entrada** normalmente conectadas a sensores.
- **Dispositivo ISO** antes de usarlos leer [14], su investigación fue el motivo de la segunda etapa del desarrollo de CyBeRSim, ver .

- **Dispositivo ICO** antes de usarlos leer [15].
- **Neurona Comando Motor.**

3.2.5.4. Borrar neurona

Equivale a la herramienta de la **Barra del diseñador ▷ Borrar neurona**.
Si la usamos desde el menú aparece un cuadro en el que se pregunta por la capa y el número de posición de la neurona a borrar, usada como herramienta borra la neurona sobre la que hagamos click con el ratón. También elimina todas las sinapsis que llegan o salen de dicha neurona.

3.2.5.5. Editar neurona

Equivale a la herramienta de la **Barra del diseñador ▷ Editar neurona**.
Si la usamos desde el menú aparece un cuadro en el que se pregunta por la capa y el número de posición de la neurona a editar, usada como herramienta selecciona la neurona sobre la que hagamos click con el ratón. El cuadro de diálogo de edición tiene una parte común a todas las tipos de neuronas, no comentamos los dispositivos ISO e ICO, la descripción es:

- **Subcapa**: por ahora aparte de los tipos 252, 253, 254, 255 comentados en 3.2.5.1**,** sólo se han usado para situar las neuronas ocultas en varios niveles en la interfaz gráfica para facilitar la visión de las conexiones laterales. Así de arriba a abajo: tipo de 0 a 74 ▷nivel 1, tipo de 75 a 149 ▷nivel 2, tipo de 150 a 199 ▷nivel 3, tipo de 200 a 255 ▷nivel 4.
- **Salida**: no hace nada.
- **Umbral**: establece el umbral de la nuerona sólo tienen sentido umbrales entre 0 y 1, umbrales superiores a 1 desactivan la neurona.
- **Valor externo inicial**: no hace nada.
- **Función de activación**: establece la función de activación de la neurona. El tipo por defecto es el lineal (suma de entradas con saturación a 1). Las definiciones de las funciones de activación son:
  siendo w_i los pesos de las sinapsis de entrada, u_i las entradas a la neurona, θ el umbral, v la salida y $s = \sum w_i u_i$.
  - **Lineal:**

$$v = \begin{cases} 0 & \text{si} \quad s \leq \theta \\ s & \text{si} \quad \theta < s \leq 1 \\ 1 & \text{si} \quad 1 < s \end{cases}$$

  - **Perceptrón:**

$$v = \begin{cases} 0 & \text{si} \quad s \leq \theta \\ 1 & \text{si} \quad \theta < s \end{cases}$$

  - **Sigmoide:**

$$v = \begin{cases} 0 & \text{si} \quad s \leq \theta \\ \frac{1}{1+e^{-5s+5}} & \text{si} \quad \theta < s \end{cases}$$

- **Monitorizable y grabable:** establece si la neurona aparece o no en el cuadro de monitorización y grabación de controles neuronales, ver 4.1.1.2 y 4.2.2.
- **Identificación:** establece el sensor, ver 3.2.5.1, al que se conecta una neurona de entrada.

3.2.5.6. Añadir sinapsis

Equivale a la herramienta de la **Barra del diseñador ▷ Añadir sinapsis**.

Si la usamos desde el menú aparece un cuadro en el que se pregunta por la capa y el número de posición de la neurona de la que sale la sinapsis y de la neurona a la que llega. **No usar este elemento de menú, le falta la elección de tipo, sino la herramienta**. Para usarla como herramienta debemos seleccionar las neuronas de salida y entrada, ver 3.2.1.1. A continuación aparece el cuadro de edición 3.2.5.8. La sinapsis aparece como una línea con una flecha en su centro apuntando hacia la neurona de llegada, la posición de la flecha puede ayudarnos a distinguir hacia que neurona se dirige la conexión en caso de solapamientos. Se puede conectar una neurona a sí misma, la conexión se muestra como un arco. Las tipos de sinapsis disponibles son:

- **Sinapsis fija**, sin aprendizaje.
- **Sinapsis hebbiana**, su peso crece si las unidades que conectan se activan a la vez.
- **Entrada a dispositivo ISO,** antes de usarlos leer.
- **Enlace de aprendizaje**, antes de usarlos leer .

3.2.5.7. Borrar sinapsis

Equivale a la herramienta de la **Barra del diseñador ▷ Borrar sinapsis**.
Si la usamos desde el menú aparece un cuadro en el que se pregunta por la capa y el número de posición de la neurona de la que sale la sinapsis y una lista para seleccionar la neurona a la que llega. Para usarla como herramienta debemos seleccionar la sinapsis, ver 3.2.1.1.

3.2.5.8. Editar sinapsis

Equivale a la herramienta de la **Barra del diseñador ▷ Editar sinapsis**.
Si la usamos desde el menú aparece un cuadro en el que se pregunta por la capa y el número de posición de la neurona de la que sale la sinapsis y una lista para seleccionar la neurona a la que llega. Para usarla como herramienta debemos seleccionar la sinapsis, ver 3.2.1.1. El cuadro de diálogo de edición tiene una parte común a todas las tipos de sinapsis, no comentamos las relacionadas con los dispositivos ISO e ICO, la descripción es:

- **Monitorizable y grabable:** establece si la sinapsis aparece o no en el cuadro de monitorización y grabación de controles neuronales, ver 4.1.1.2 y 4.2.2.
- **Editar peso:** establece el valor del peso de la sinapsis. El valor del peso se elige de la lista y puede ser editado si no hay otra sinapsis que lo use, (si necesitamos editarlo lo mejor es elegir alguno que todavía no esté usado, si los hay, y editarlo).
- **Constante de aprendizaje:** establece el valor de la constante de aprendizaje para sinapsis hebbianas.

3.2.5.9. Añadir subred

Equivale a la herramienta de la **Barra del diseñador ▷ Añadir subred**.
Si la usamos desde el menú aparece un cuadro en el que se selecciona el fichero de la subred, en otro cuadro se indica la posición de la subred a añadir.

3.2.5.10. Borrar subred

Equivale a la herramienta de la **Barra del diseñador ▷ Borrar subred**.
Si la usamos desde el menú aparece un cuadro en el que se solicita el número de subred a borrar, usada como herramienta borra la subred sobre la que hagamos click con el ratón. También elimina todas las sinapsis que llegan o salen de dicha subred.

3.2.5.11. Editar subred

Equivale a la herramienta de la **Barra del diseñador** ▷ **Editar subred**.
Si la usamos desde el menú aparece un cuadro en el que se pregunta por el número de subred a editar, usada como herramienta selecciona la subred sobre la que hagamos click con el ratón. Abre una nueva ventana en la que se puede ver la subred a editar.

3.2.6. El menú simulador

3.2.6.1. Descripción de la estructura de una simulación

Antes de describir los elementos de este menú vamos a comentar brevemente la estructura de las simulaciones disponibles en CyBeRSim. Los elementos que componen una simulación son: el robot con sus sensores y motores, el control que procesa la información de recogida por los sensores y la tranforma en órdenes motoras y el mundo o escenario donde se desarrolla la simulación.

- El robot se compone de cuerpo, sensores y motores. Los sensores implementados se enumeran en 3.2.5.1, el tipo de movimiento implementado corresponde a dos ruedas motrices independientes fijas con respecto a su eje vertical. Se dispone de los cuerpos de Khepera y Pioneer 3-at con escalas distintas. Asociado al robot encontramos dispositivos de monitorización en pantalla, 4.1.1.1, y grabación en fichero, 4.2.1, de trayectorias y velocidades.
- Los controles incluidos como ejemplos (`.../examples/controls`) se basan en redes neuronales reactivas, salvo un control nulo que se incluye como ejemplo para programadores. Asociados a los controles encontramos dispositivos de monitorización en pantalla, 4.1.1.2, y grabación en fichero, 4.2.2, de actividades y pesos de la red neuronal.
- El tipo de mundos manejados en esta versión son ficheros de formato Xfig 3.2, su edición se realiza con este programa, ver 3.2.7.

La configuraciones de las simulaciones se guardan en ficheros `.sim` que incluyen referencias a ficheros de robots `.robot` y estos a sensores `.sen`, imágenes `.xpm` y `.xbm` y ficheros de grabación-monitorización `.robmon`, también incluyen referencias a ficheros de control `.ctrl` y éstos a ficheros de red neuronal `.net` y a ficheros de monitores y grabadores de red neuronal `.rmon`. Todos estos ficheros tienen formato XML y pueden ser editados a mano si fuera necesario.

Las dimensiones de la ventana principal de la simulación y la del mundo no están limitadas por las de la pantalla del ordenador, si se abren varias cajas monitores analógicos puede ser conveniente utilizar dos monitores conectados al ordenador.

El menú del **Simulador** contiene las herramientas para crear, editar y ejecutar simulaciones. Las más usuales se hallan también en la **Barra del simulador**, siendo el manejo del programa más cómodo e intuitivo desde dicha barra. Se describirán ambas formas de utilización conjuntamente.

3.2.6.2. Nueva simulación

Cuando se crea una nueva simulación se suceden una serie de cuadros de diálogo que nos preguntan sobre:

- Tipo de simulación: solamente está disponible la simulación con un robot (doble click elige).
- Supertipo de control: basados en red o no.
- Tipo de control: depende del supertipo elegido:
  - Control nulo: no basado en red, solamente mantiene constante la velocidad del robot.
  - Control de red neuronal básico: para simulaciones de redes neuronales reactivas con posibilidad de aprendizaje hebbiano
  - Control de red neuronal con dispositivos ISO e ICO: ver referencias en 3.2.5.3.

- Si se ha elegido un control neuronal se nos pedirá abrir un fichero de red neuronal (`.net`).
- Se nos pedirá abrir un fichero de robot (`.robot`).
- Se nos pedirá un nombre para el fichero de control que estamos creando (`.ctrl`).
- Se nos pedirá un nombre para el fichero simulación de que estamos creando (`.sim`).
- Para finalizar se abrirá el cuadro de edición de la simulación, ver 3.2.6.4.

El proceso puede ser cancelado en cualquier momento.

3.2.6.3. Cargar simulación

Si no hay un mundo cargado se nos pedirá abrir uno (`.fig`). Si durante la carga de la simulación no se encontrara alguno de los ficheros necesarios se nos mostrará un cuadro para que lo busquemos, en ese momento el proceso de carga puede ser cancelado.

3.2.6.4. Editar simulación

La estructura del cuadro de diálogo de edición de la simulación depende del tipo de control cargado, describiremos el caso de controles neuronales que es el implementado en los ejemplos. El cuadro de diálogo se organiza en varias páginas que se eligen mediante doble click en la vista de árbol de la izquierda de la ventana:

- **General:** parámetros generales de la simulación.
  - **Periodo de avance:** debe ajustarse al ciclo de muestreo de sensores del robot, 20 ms. en Khepera y 100 ms. en Pioneer 3-at.
  - **Velocidad de simulación:** puede elegirse tiempo real, definido por **Periodo de avance**, máxima del sistema y cámara lenta, múltiplos de **Periodo de avance**.
  - **Duración:** se establece la duración de la simulación y si tras una parada los relojes se reinician o no, **Reiniciar/Continuar**.
  - **Actualización de pantalla:** por motivos de eficiencia puede actualizarse la pantalla cada cierto tiempo o nunca, por defecto se actualiza siempre.
- **Robot:** Diversas variables cinemáticas y grabación-monitorización de posiciones y velocidades
  - **Cinemática**:
    A este cuadro de diálogo también se accede pulsando el botón derecho del ratón sobre el robot en la ventana del mundo.
    - **Ruido:** porcentaje de ruido uniforme que afecta a las medidas de sensores y a órdenes de velocidad.
    - **Modo de sensores:** en el modelo Khepera I los sensores de proximidad tiene un alcance de 5 cm. y en Khepera II de 10 cm. Esta proporción también se aplica a los sensores de luz.
    - **Inercia:** grado de inercia aplicado a los movimientos.
  - **Cinemática_<nombre de robot>:**
    Para establecer la posición, orientación y velocidad de robot. A este cuadro de diálogo también se accede pulsando el botón derecho del ratón sobre el robot en la ventana del mundo. Se puede guardar esta información en el fichero de simulación si marcamos la casilla correspondiente y posteriormente guardamos la simulación.
  - **Grabación y monitorización**:
    Ver 4.2.1 y 4.1.1.1.

- **Control**:
  Las páginas correspondiente al control solamente aparecen si son necesarias, p. ej. el control nulo, ver 3.2.6.2, no las necesita.

  - **Aprendizaje:**
    - **Constante de aprendizaje:** marcar **Sobreescribir valores individuales**, por ahora relacionado con dispositivos ISO e ICO, ver referencias en 3.2.5.3.
    - **Pesos iniciales:** marcar **Sobreescribir valores individuales** para inicializar los pesos de las sinapsis no fijas, por ahora relacionado con dispositivos ISO e ICO, ver referencias en 3.2.5.3.
    - **Propagación:** por la red puede ser secuencial, cada capa transmite a la siguiente en un paso de simulación, o instantánea, todas las capas en un paso.
    - **Estabilización de pesos:** relacionado con dispositivos ISO e ICO, ver referencias en 3.2.5.3.
  - **Grabación y monitorización**:
    Ver 4.2.2.

3.2.6.5. Guardar simulación

Si ficheros relacionados con el de la simulación, ver 3.2.6.1, han cambiado durante la edición de la simulación, se nos pedirá si queremos guardar sus modificaciones o no. Al guardar la simulación también se guardan las posiciones de las ventanas asociadas que estén abiertas.

3.2.6.6. Cerrar simulación

Si ha habido cambios durante la edición de la simulación se nos pedirá si queremos guardar las modificaciones o no.

3.2.6.7. Iniciar simulación

Equivale a la herramienta de la **Barra del simulador** ▷ **Iniciar simulación**.
Inicia el proceso de control y todas las operaciones de grabación y monitorización si dicha opción se activó, ver 3.2.6.11.

3.2.6.8. Simulación paso a paso

Equivale a la herramienta de la **Barra del simulador** ▷ **Simulación paso a paso**
La duración paso de simulación se establece en **Periodo de avance**, ver 3.2.6.4.

3.2.6.9. Pausa en simulación

Equivale a la herramienta de la **Barra del simulador** ▷ **Pausa en simulación**.
Pausa momentánea en la simulación.

3.2.6.10. Parar simulación

Equivale a la herramienta de la **Barra del simulador** ▷ **Parar simulación**.
El proceso de control se para junto con el robot y las operaciones de grabación y monitorización. Aparece un cuadro indicando los tiempos de ejecución de cada parte de la simulación, lo usan los desarrolladores para comprobar la eficiencia de algoritmos, este informe se puede añadir a un fichero asociado a la simulación denominado `<nombre_de_fichero_de_simulación>.log` que se guarda en el directorio donde se halle el fichero de simulación.

3.2.6.11. Grabar simulación (y monitorizar)

Equivale a la herramienta de la **Barra del simulador** ▷ **Grabar simulación**.

Si el botón está pulsado se monitorizarán y grabarán aquellos proceso que estén habilitados en la simulación, ver 4.1.1. Es necesario pulsar de nuevo (botón arriba) para finalizar el proceso de grabación y cerrar los archivo de datos, ya que estos se añaden a los fichero hasta se realiza esta operación o la simulación se cierra.

3.2.7. El menú mundo

Los mundos en CyBeRSim-edu se crean y editan con Xfig o Winfig, ver 2.3.1.2, que deben estar en el path del sistema. La opciones de este menú **Crear** y **Editar** invocan a Xfig o Winfig. Solamente un subconjunto del formato Xfig 3.2 es soportado, concretamente elipses, líneas poligonales cerradas, los splines cerrados se intrepretan como líneas poligonales cerradas. La forma de insertar luces en los mundos depende del programa utilizado.

- **Xfig:** Para poder insertar luces en los mundo hay que decirle a Xfig que las tenemos en una librería Xfig para lo cual se debe copiar crear un directorio (en Linux como root), no importa el nombre, dentro del directorio `Libraries` de la distribución de Xfig y copiar allí el fichero `.../scr/world/imagenes/luz.fig`, entonces podremos encontrar nuestra librería en Xfig, ver manual de Xfig, con el nombre de directorio creado. Hay otras formas más "ortodoxas", ver manual de Xfig.
- **WinFig:** La versión gratuita de WinFig no incluye el manejo de librerías. Por lo cual debemos realizar a mano el siguinete proceso:
  Abrir el fichero `.../src/world/imagenes/luz.txt` y el fichero de mundo (`.fig`) elegido con un editor de textos. Copiar el contenido de `luz.txt` en el fichero de mundo justo antes de la última línea (que contiene un '-6'), creando una nueva línea. Repetir el proceso si se quieren insertar más luces. Guardar el fichero de mundo y abrirlo con WinFig. Las luces aparecen superpuestas en las esquina superior izquierda del mundo, trasladarlas a donde se desee.

3.2.8. El menú ventana

Este menú tiene las opciones de colocación en **Mosaico** y **Cascada**, no muy útiles ya que no sirven para la disposición de las ventanas asociadas a la simulación. A continuación aparecen los nombres de las ventanas abiertas para poder traerlas a primer plano.

3.2.9. El menú ayuda

El menú de **Ayuda** nos permite acceder a este manual en formato HTLM como se comentó en 3.1.

Capítulo 4

Monitores y grabadores

4.1. Monitores

Dentro de los tipos de monitores disponibles tenemos los analógicos y los digitales.

4.1.1. Monitores analógicos

Son ventanas rectagulares en las que se pueden representar hasta 6 gráficas de señales dependientes del tiempo. Normalmente tienen fondo negro pero en algunos casos se puede cambiar a blanco Los monitores se pueden agrupar en cajas de hasta 12 (4x3) monitores. Al cuadro de edición de un monitor se puede acceder haciendo doble click sobre él o desde el cuadro de edición de la simulación, ver 4.1.1.1 y 4.1.1.2. En contenido de un monitor se borra si otra ventana lo oculta.

Los items del cuadro de edición de un monitor pueden aparecer con otras denominaciones cuando aparecen dentro del cuadro de edición de la simulación, los items son los siguientes:

- **Título**
- **Tiempo de actualización:** tiempo en segundos que se representa en el eje x. Cuando ese tiempo, o un múltiplo suyo, se sobrepasa se comienza a pintar desde el lado izquierdo del monitor.
- **Número de etiquetas de tiempo:** número de divisiones en el eje x.
- **Valor máximo:** del eje y.
- **Valor mínimo:** del eje y.
- **Número de etiquetas:** del eje y.

4.1.1.1. Robot

El robot tiene asociada una caja con dos monitores en la que se representan sus velocidades lineal (cm/s) y radial (grados/s). Ambos monitores se pueden editar desde el cuadro de edición de la simulación en la página **Robot ▷ Grabación y monitorización**. Aquí las etiquetas son más específicas que en el cuadro de edición propio del monitor y aparecen los items:

- **On:** la habilitación o deshabilitación de la monitorización para que esta se realize o no, cuando pulsamos el botón para grabar/monitorizar la simulación, ver 3.2.6.11.
- **Vista de monitor: Osciloscopio**, fondo negro, o **Papel**, fondo blanco.

4.1.1.2. Red neuronal

A la edición de los monitores de red se accede desde el cuadro de edición de la simulación en la página **Monitorización** del control. En ella encontramos varias widgets:

- **On:** la habilitación o deshabilitación de la grabación para que esta se realize o no si pulsamos el botón para grabar/monitorizar la simulación, ver 3.2.6.11.
- **Cargar de fichero:** los parámetros de los monitores, ficheros de tipo `.robmon`. Este botón es compartido con la página de **Grabación**, ver 4.2.2.
- **Guardar en fichero:** los parámetros de los monitores. Este botón es compartido con la página de **Grabación**, ver 4.2.2.
- **Dispositivos y sus conexiones:** esta área de la ventana se divide en tres partes. Las de los lados son dos vistas de árbol, en ellas sólo aparecen los dispositivos y conexiones declarados como grabables y monitorizables, ver 3.2.5.5 y 3.2.5.8. En la izquierda están los dispositivos y conexiones que no se desea monitorizar y en la de la derecha los que sí y cómo se organizan en monitores. Cada dispositivo tiene su propia caja de monitores, hasta dos monitores para el dispositivo y hasta diez para sus conexiones de salida. En la zona central encontramos los botones de operación:
  - **Monitores añadir:** debe estar seleccionado un dispositivo del área derecha para añadirle un monitor.
  - **Flechas:** para pasar de una ventana a otra dispositivos y conexiones. Cuando se pasa un dispositivo a la zona de monitorización se abre el cuadro de edición correspondiente, ver **General editar**. Para pasar una conexión a la zona de monitorización tiene que estar seleccionado un monitor del dispositivo correspondiente.
  - **General editar:** al pulsar este botón se invoca al método de edición del elemento, o sus monitores, seleccionado en el árbol de monitorizados, también se accede a la edición haciendo doble click sobre el elemento. En el caso de dispositivos aparece un cuadro de diálogo para dos monitores para editar los parámetros comentados en 4.1.1 y además con una lista, **Señales**, donde elegimos los valores a monitorizar del dispositivo en cada monitor. Dependiendo del tipo de dispositivo, hay disponibles hasta tres valores por ahora: entrada, salida y primera diferencia de la salida, pero se puede aumentar su número programáticamente.
    Al pulsar el botón sobre un monitor se abre un cuadro de diálogo para editar los parámetros del monitor comentados en 4.1.1.

4.1.2. Monitores digitales

Se usan para visualizar los valores de los sensores y la velocidad de cada rueda del robot. Son circulares.

4.2. Grabadores

Los grabadores escriben en un fichero los valores de las señales que se desee junto con la marca de tiempo de la lectura de los valores. La escritura se realiza en un fichero de texto plano, cada registro ocupa una línea y cada dato ocupa una columna, las columnas están separadas por tabuladores, la primera columna es la marca de tiempo.

El cuadro de edición de un grabador se compone de:

- **On:** la habilitación o deshabilitación de la grabación para que esta se realize o no si pulsamos el botón para grabar/monitorizar la simulación, ver 3.2.6.11.

- **Modos de escritura**:
  - **Auto: Añadir** o **Sobreescribir** dependiendo de si el modo de reloj de la simulación está en **Continuar** o **Reiniciar**, ver **Duración** en 3.2.6.4.
  - **Añadir:** Los datos se añaden al fichero, sin borrar los previos, cada vez que se inicia la simulación.
  - **Sobreescribir:** Los datos sobreescriben el fichero cada vez que se inicia la simulación.
- **Archivo:** nombre del fichero de datos, ver 4.2.1 y 4.2.2.
- **Escribir cada:** periodo en segundos de grabación de datos, el valor **Siempre** equivale a cada paso de simulación.

Los datos de cada grabación van precedidos de una cabecera con hora, fecha e información sobre lo grabado. Si se van a procesar los ficheros de datos con otros programas puede convenir hacer una copia y eliminar dicha cabeceras sin dejar huecos entre los datos.

4.2.1. Robot

La única elección que el usuario puede hacer es la del fichero donde se guardarán los datos, se realiza en el cuadro de edición de la simulación en la página **Grabación y monitorización** del robot. En la cabecera de grabación aparecen datos sobre las dimensiones del mundo donde corrió el robot, entre otros. En la zona de datos se graban: las marcas de tiempo, las coordenadas x e y, la velocidad lineal y la radial. La grabación de datos de realiza a cada paso de simulación por lo que, por espacio de disco, no se debería grabar al robot durante periodos largos de tiempo, además en la representación gráfica de una trayectoria extensa es difícil apreciar nada, si el robot no se mueve por un área también extensa.

4.2.2. Red neuronal

A la edición de los grabadores de red se accede desde el cuadro de edición de la simulación en la página **Grabación** del control. Esta página tiene un funcionamiento similar a la de **Monitorización**, ver 4.1.1.2, salvo que aquí no existe un concepto como el de monitor y cajas de monitores, ya que todo se graba en un único fichero por dispositivo. Los campos de edición son los comentados en 4.2 salvo por **Archivo**, ya que cada dispositivo tiene su propio archivo de grabación, y a la edición del nombre de ese archivo se accede seleccionando el dispositivo en el área de grabados y haciendo doble click o pulsando el botón **General editar**.

Capítulo 5

Tutorial

5.1. El primer robot.

Para iniciar este tutorial comenzaremos creando el vehículo de Braitenberg (que llamaremos BCK y está en el archivo `.../examples/nets/BCK.net`) descrito en el paquete comercial de Khepera (el robot real), citado por C . Touzet [16]. Se ha cambiado el orden de asignación de sensores a las neuronas de entrada para evitar en lo posible cruces entre sinapsis. Los valores de los pesos dados en el artículo citado deben ser divididos por 10. En la figura tenemos la red del robot BCK, es un ejemplo del vehículo número 3(b) pg. 11 y figura 4 de la obra de Braitenberg [1], con conexiones cruzadas inhibitorias. Se puede cargar en CyBeRSim o construirlo como se comenta a continuación.

1. Tras arrancar CyBeRSim hacemos click sobre el botón de la herramienta descrita en 3.2.5.3 (situando por unos momento el puntero del ratón sobre un icono de las barras de herramientas aparece una ventana con su nombre, o se puede usar el botón de la interrogación que aparece en dichas barras) y aparecerá el cuadro de diálogo citado en 3.2.5.1 usaremos 8 neurona de entrada, 2 ocultas y las 4 de salida. Aparecerá la figura del robot BCK sin conexiones.

2. A continuación editamos las neuronas ocultas dandole tipo 252 a la número 9 y tipo 255 a la 10.

3. A las neuronas de entrada se les puede dar un umbral bajo (<=0.2 p. ej.) para hacerlas más realistas.

4. A continuación elegiremos la herramienta 3.2.5.6, el tipo de sinpsis debe ser fijo (fixed).

   a) hacemos click sobre la neurona 2 y Ctrl-click sobre la 14, en el cuadro de diálogo posterior elegiremos el segundo peso (el primero es mejor conservarlo para el peso 1.00, muy habitual) y editaremos su valor para que sea -1.7;

   b) hacemos click sobre la neurona 3 (la 14 sigue seleccionada como neurona de llegada), elegimos el tercer peso lo editamos para que sea -1.1 ;

   c) hacemos click sobre la 5 y le ponemos de peso -0.7 (está en la tabla que aparece en el diálogo).

5. Hacemos click sobre el fondo para deseleccionar ambas neuronas

6. Repetimos el punto 3 conectando las neuronas de entrada 5 (-0.7), 6 (-1.1) y 7 (-1.7) con la de salida 11, entre paréntesis se dan los pesos.

7. Tras repetir 4 conectamos la neurona oculta 9 a la de salida 11 y la 10 a la 14 con pesos de valor 1.0, como se explicó en 3 y 4.

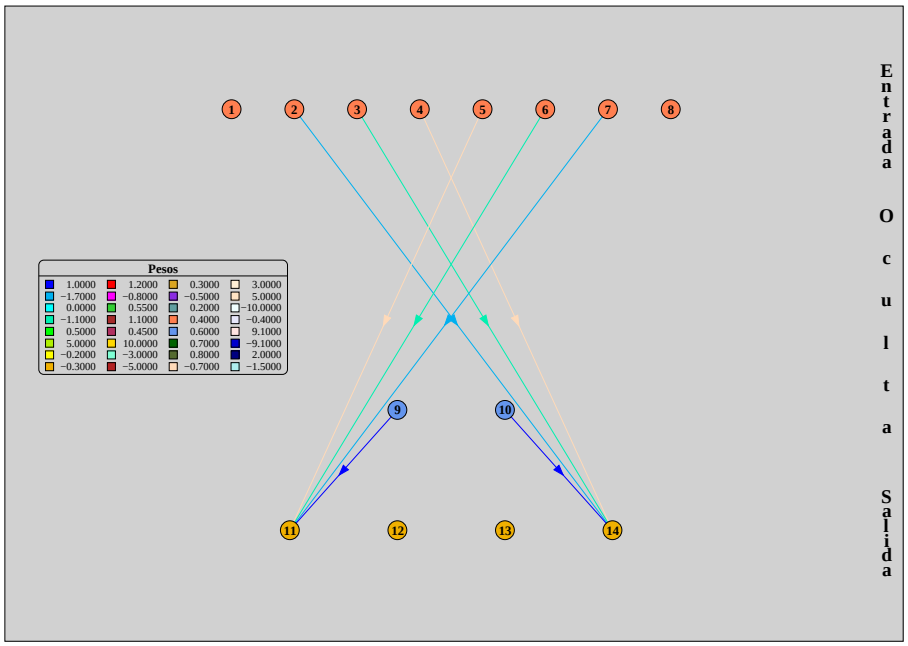


Figura 5.1: Red del robot BCK

8. Guardamos el fichero.

9. Para probar la red debemos crear una simulación con control basado en red al que asociemos dicha red, ver 3.2.6.2 y la lista de archivos más abajo, o más fácil, usar la simulación de ejemplo `.../examples/simulations/bck-org.sim`. Los distintos archivo que componen esta simulación son:

   - `.../examples/controls/BCK-01-control.ctrl`,
   - `.../examples/nets/BCK.net`, (incluido en `.crtl`)
   - `.../examples/robot/twowheelsnoholo.robot`,
   - `.../examples/robot/robot_monitor03.robmon`,
   - `.../examples/robot/sensors/khep_proximity.sen`, (incluido en `.robot`)
   - `.../examples/robot/sensors/khep_bump.sen`. (incluido en `.robot`)

El robot exhibe un comportamiento bastante aceptable de evitación de obstáculos pero se queda parado fácilmente ante ellos debido a que los estímulos de un lado y otro tienen similar intensidad, mínimos locales. Para evitar en lo posible estas situaciones, se pueden cambiar los pesos de un lado para romper la símetría.

5.2. Un robot más complejo

Este robot se ha diseñado para que superara los inconvenientes del anterior y además buscara fuentes de luz. Describiré ambas redes por separado y luego como se ensamblan para conseguir las conductas deseadas. En su contra hay que señalar que los movimientos de este robot son un tanto bruscos cuando evita obstáculos y no creo que el robot real pudiera responder adecuadamente a los cambios de velocidad, habría que comprobarlo. Este no emplea las neuronas ocultas de velocidad "crucero" hacia atrás, pero podría cambiarse su diseño.

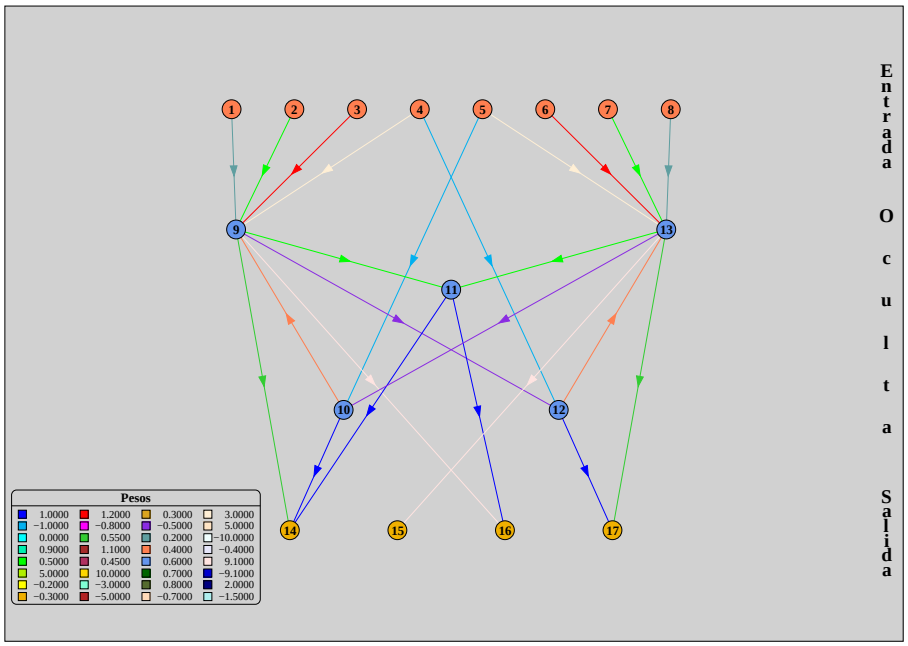


Figura 5.2: Red del robot evitador de obstáculos.

5.2.1. El evitador de obstáculos

Su fichero es `.../examples/nets/evitador.net`. Este robot usa principalmente conexiones directas y cruzadas principalmente excitatorias, los ajustes de los pesos están afinados a mano para conseguir la conducta deseada: muy pocos o ningún choque (bump), paso por pasos estrechos, no quedar atrapado en rincones en los mundos, la supervisión se realiza a ojo. Dada la simetría bilateral casi absoluta de la red, salvo por la neurona 11, describiré el lado izquierdo solamente.

En este robot preferí dirigir las conexiones procendentes de cada una de las neuronas de entrada (de la 1 a la 4 en el lado izquierdo) a una neurona oculta la 9 (evitación izquierda) con pesos crecientes hacia los sensores centrales (ver leyenda de la figura), en vez de hacia la de salida motor izquierdo adelante (aquí la 14, en BCK la 11), para una mayor modularidad en el diseño. Pienso que los sistemas sensori-motores deben organizarse con una estructura modular, como ocurre en los seres vivos. Además así se consigue una mayor posiblidad de interacción dentro de la misma red y con la de búsqueda de luz. Esta neurona 9 tiene un peso de 0.6 para que sólo se active si el obstáculo está cerca Khepera I tiene 5.5 cm de diámetro (6 cm Khepera II) y los sensores de infrarrojos tienen un rango de detección de entre 2 (salida de neurona = 1) y 5 cm (salida de neurona = 0), aunque con un rango de error considerable, ver [16], en la nueva versión del robot real se ha aumentado el alcance hasta 10 cm, pero estos ejemplos están pensados para el modo de sensores **Khepera I**, ver 3.2.6.4. De la neurona 9 parte una sinapsis inhibitoria hacia la neurona 12 (velocidad de "crucero" del lado opuesto) , dos excitatorias, una hacia las neuronas 14 (motor izquierdo adelante) y la otra hacia la 16 (motor opuesto atrás) con un peso alto para forzar un brusco giro a la derecha para evitar el obstáculo que aparece por la izquierda. La neurona 11 tiene un umbral muy alto (0.9) para que sólo pueda ser activada por el efecto conjunto de las neuronas 9 y 13 (caso de "acorralamiento") forzando un brusco giro a la izquierda (excita a las neuronas 14 y 16).

Completan el diseño una conexión cruzada inhibitoria de la neurona 4 (sensor central izquierdo) a la de velocidad "crucero" del lado opuesto 12 para favorecer el paso por sitios estrechos y otra excitatoria de la 10 (velocidad "crucero" izquierda) a la 9 (evitación izquierda) para aumentar la "sensibilidad" de ésta con la velocidad.

Para ver los pesos con detalle abrid el fichero con CyBeRSim y editad los elementos que os

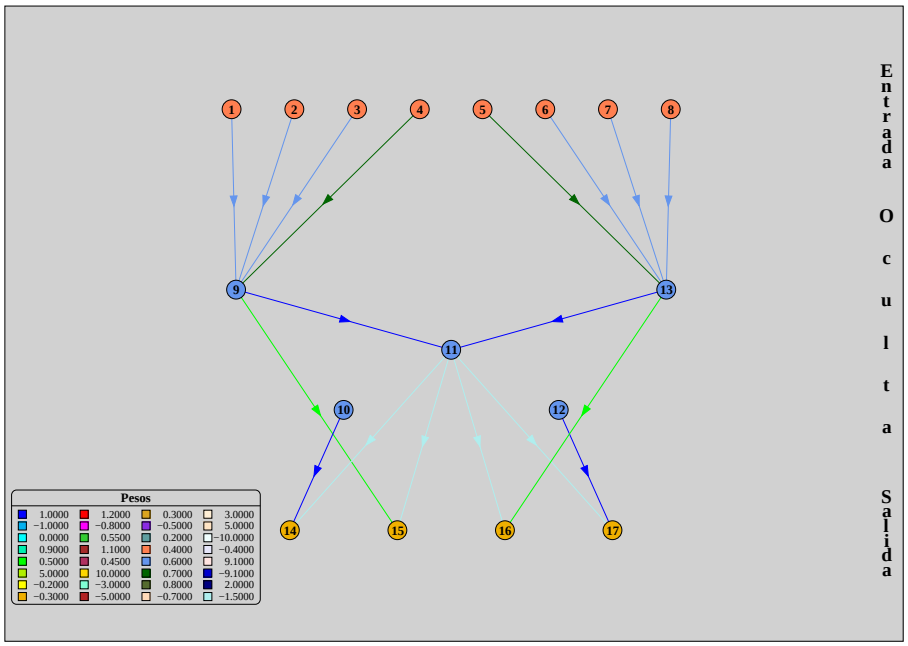


Figura 5.3: Red del robot buscaluz.

interesen, las neuronas cuyos umbrales no se han citado lo tienen igual a 0, a las neuronas de entrada se les puede dar un umbral bajo (<=0.2 p. ej.) para hacerlas más realistas. Este robot muestra un compotamiento muy bueno con una velocidad >= 30 cm/s , aunque al aumentar la velocidad aumenta ligeramente la probabilidad de choque o atrapamiento, a velocidades menores se queda parado en bastantes ocasiones.

5.2.2. El buscador de luz

Se encuentra en el fichero `.../examples/nets/buscaluz.net`. Su objetivo es detectar una fuente de luz y pararse frente a ella a poca distancia. Esta red presenta simetría bilateral absoluta, todas las conexiones son directas, las neuronas de la 1 a la 8 reciben información de los sensores de luz, que tienen un alcance de (25/50 cm).

Describo el lado izquierdo, las neuronas de entrada envían su salida a la neurona oculta 9 (luz lado izquierdo) el peso de la central es un poco superior a las de las demás, para encarar mejor el objetivo. La neurona 9, que tiene un peso bajo (0.2 para evitar que reaccione al ruido de los sensores pero detecte la luz a distancia) excita a su vez a la 15 (motor izquierdo atrás) forzando un giro hacia ese lado. La neurona 10 es la de velocidad de "crucero" izquierda, con umbral= 0. La neurona central, la 11, tiene la misión de para al robot cuando éste se encuentre cerca y frente a la luz, su elevado umbral (0.95) hace que sólo pueda ser excitada por el efecto conjunto de la 9 (luz lado izquierdo) y la 13 (luz lado derecho), consiguiendo así el objetivo deseado ya que la neurona 11 inhibe, con un peso moderadamente alto, al todas las de la capa de salida (14, 15, 16, 17).

Para probar el funcionamiento de este robot hay que ponerlo en un mundo sin obstáculos, dado que no tiene sistema de evitación.

5.2.3. Ensamblando las dos redes: el moscardón de Braitenberg

Le dí ese nombre porque su comportamiento me recordaba al del insecto que va buscando una flor que libar (luz) por entre ramas y hojas y teniendo que salir, de forma un tanto violenta, de rincones o huecos en los que se siente atrapado. He de comentar que no diseñé cada red por separado totalmente, o sea, evitador y buscaluz son simplemente las redes del moscardón separadas

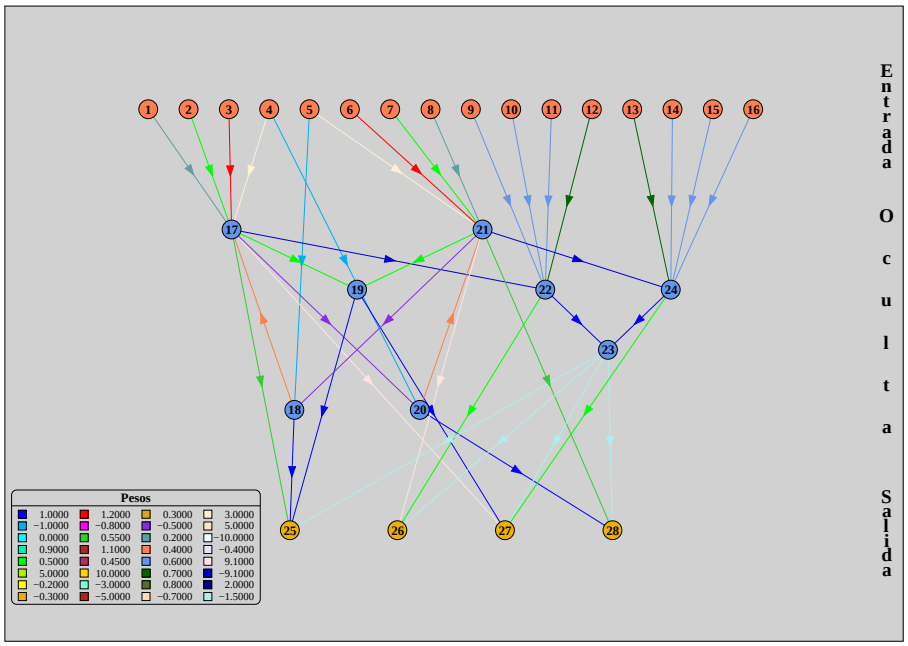


Figura 5.4: Red del robot moscardón de Britenberg.

una vez finalizado el diseño.

La primera cuestión que tuve que abordar al ensamblar las dos redes fue la de cómo establecer un orden de prioridades para las conductas que definen, una de las cuales dirige al robot a su meta. En nuestro caso el sistema de evitación debe inhibir el de seguimiento de luz en caso choque inminente, por este motivo se condujeron casi todas las salidas de las neuronas de entrada de cada lado a una única neurona oculta en cada red sensorial, ver figura. Así una fuerte conexión inhibitoria de la neurona que ejerce la función evitación lado izquierdo (número 17) sobre la que detecta luz lado izquierdo (22) establecerá dicha prioridad, de la misma forma en el lado derecho (21 inhibe a 24). En el caso de una luz colocada cerca de un obstáculo (en el simulador la luces, por sí mismas, no se reconocen como obstáculos y los obstáculos son transparentes a la luz) el comportamiento definido por la red de seguimiento de luz, parada, evita el obstáculo, al a vez que inhibe la reacción del sistema de evitación que impulsaría al robot hacia atrás.

La segunda fue la afinación de la red, una vez terminada es fácil de explicar pero hubo muchísimas pruebas previas con distintos conexionados y con distintos pesos.

Para probar esta red cargar la simulación `.../examples/simulations/moscardon.sim`.

Los distintos archivo que componen esta simulación son:

- `.../examples/controls/moscardon.ctrl`,
- `.../examples/nets/moscardon.net`, (incluido en `.crtl`)
- `.../examples/robot/twowheelsnoholo-full.robot`,
- `.../examples/robot/robot_monitor03.robmon`,
- `.../examples/robot/sensors/khep_proximity.sen`, (incluido en `.robot`)
- `.../examples/robot/sensors/khep_bump.sen`. (incluido en `.robot`)

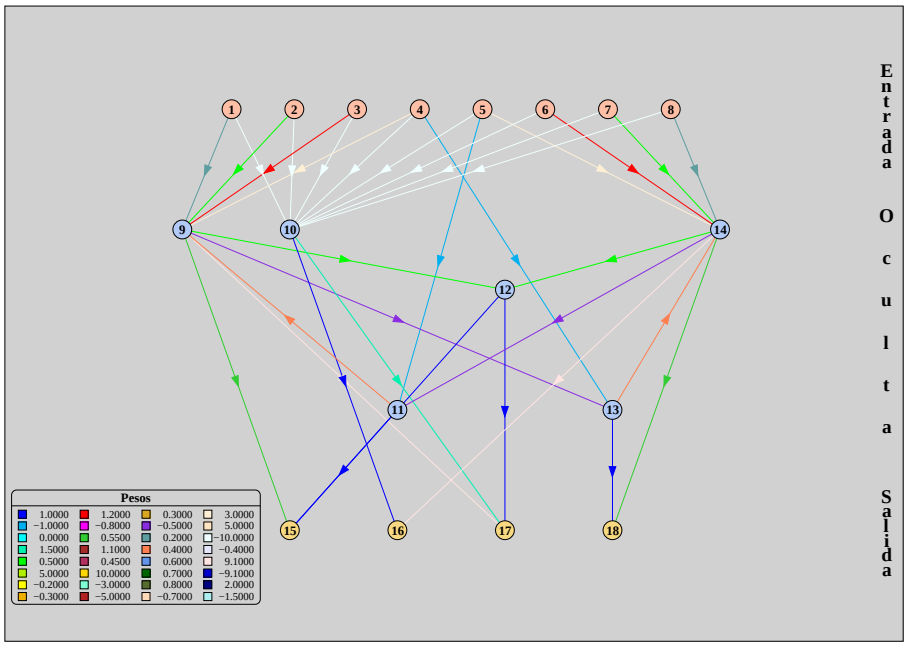


Figura 5.5: Red del robot seguidor de paredes.

5.3. Un nuevo comportamiento: seguimiento de paredes

El robot seguidor de paredes está basado en el evitador de obstáculos, sección 5.2.1, al que se le ha añadido un nuevo reflejo. Denominaremos evitación de la privación sensorial a nuevo reflejo, ya que su misión es evitar que el robot llegue a una zona en la que no perciba nada a través de sus sensores de proximidad, mediante un movimiento de retroceso al entrar en una de estas zonas.

Este reflejo se consigue al añadir una neurona oculta que siempre está activada actuando sobre las neuronas motoras de marcha atrás derecha e izquierda, con una salida que supera la velocidad crucero del robot. Sin embargo esta neurona se ve inhibida por las correspondientes a los sensores de proximidad, salvo cuando la entrada de los sensores es casi nula, lo que correspondería a estar a punto de entrar en una zona sin percepción sensorial y, por lo tanto, sin posibilidad de obtener información del entorno. Imaginemos que estamos en una sala, grande y completamente desconocida, totalmente a oscuras, tenderíamos a caminar pegados a la pared y no a adentrarnos a ciegas en la sala. Una diferencia en la salida a los motores hace que el movimiento de retracción junto con el mismo efecto sesgo introducido en la neurona anti-acorralamientos del evitador, ver 5.2.1, hacen que el seguimiento de las paredes sea a izquierdas.

El robot consigue seguir la pared (a izquierdas) aunque con problemas, ya que entra en círculos viciosos, debidos a que en los recodos a veces enfrenta la pared de manera que el sistema evitador de obstáculos dirige el movimiento a derechas, y en las zonas rectas sigue una trayectoria en zigzag debido a las reacciones enfrentadas del sistema evitador de obstáculo y el de evitación de privación sensorial.

5.4. ISO-learning e ICO-learning

Estos ejemplos se basan en una teoría mucho más compleja, los interesados pueden leer. Básicamente es un robot que aprende a no chocar con obstáculos. Las simulaciones son

- `.../examples/simulations/khepera-iso.sim`,
- `.../examples/simulations/khepera-ico.sim`

- `.../examples/simulations/pioneer-iso.sim.`

En esta última simulación el robot empleado es un Pioneer 3 AT [13].

Capítulo 6

Programación en CyBeRSim

Para la programar CyBeRSim hay que descargarse el código fuente y seguir las instrucciones de compilación a instalación, ver 2.3, se dispone de varios manuales: el manual de arquitectura del software y el manual de referencia.

Capítulo 7

Copyright

Bibliografía

[1] Valentino Braitenberg. *Vehicles: experiments in synthetic psychology*. MIT, 1986.

[2] José Manuel Cuadra Troncoso. Documento de arquitectura del software de NeuralDis, septiembre 2005.

[3] José Manuel Cuadra Troncoso. *Manual de Referencia de NeuralDis*. noviembre 2007.

[4] GNU. GNU C,C++ MSWindows compiler. `http://www.mingw.org`.

[5] GNU. GNU C,C++,Java Linux compiler. `http://gcc.gnu.org/`.

[6] K-Team Corporation. Khepera minirobot, 2005. `http://www.k-team.com/robots/khepera`.

[7] KDE. Qt3-win. `http://www.kde.org/`.

[8] Kde-cygwin. Qt3-win32. `http://qtwin.sourceforge.net/qt3-win32/`.

[9] LaTeX. LaTeX. `http://www.latex-project.org/`.

[10] LyX. LyX. `http://www.lyx.org/`.

[11] Olivier Michel. Khepera simulator, 1995. `http://diwww.epfl.ch/lami/team/michel/khep-sim`.

[12] MiKTeX. MiKTeX. `http://www.miktex.org/`.

[13] MobileRobots. Pioneer 3 AT. `http://www.activrobots.com/ROBOTS/p2at.html/`.

[14] B. Porr y F. Wörgötter. Isotropic sequence order learning. *Neural Computation*, (15):831–864, 2003.

[15] Bernd Porr y Florentin Wörgötter. Strongly improved stability and faster convergence of temporal sequence learning by utilising input correlations only. *Neural Computation*, 18(6):1380–1412, 2006.

[16] Claude Touzet. Neural reinforcement learning for behaviour synthesis. *Robotics and autonomous systems*, 22(3):251–281, diciembre 1997.

[17] TrollTech. Qt. `http://www.trolltech.com/products/qt/qt3/`.

[18] TrollTech. Qt, 2007. `http://www.trolltech.com`.

[19] TrollTech. Qt, 2007. `http://www.trolltech.com/products/qt/qt3/platforms/index`.

[20] WinFIG. WinFIG. `http://www.schmidt-web-berlin.de/winfig/`.

[21] Xfig. Xfig. `http://www.xfig.org/`.