UNIVERSIDAD POLITÉCNICA DE CARTAGENA

ESCUELA TÉCNICA SUPERIOR DE INGENIEROS DE TELECOMUNICACIÓN

DEPARTAMENTO DE TECNOLOGÍAS DE LA INFORMACIÓN Y LAS COMUNICACIONES

PROYECTO FIN DE CARRERA

INGENIERÍA DE TELECOMUNICACIÓN

Estrategias para la Planificación de Trayectorias con Arquitecturas Neuronales de Aprendizaje

TITULACIÓN: Ingeniero de Telecomunicación
DIRECTOR: Juan Luis Pedreño Molina
AUTOR: Adolfo A. García-Figueras y López

CURSO ACADÉMICO
2004/2005

Índice

En agradecimiento a mi director del proyecto y a mi familia por el tiempo que les he robado para la realización de este trabajo

Objetivos

Uno de los campos de mayor actualidad en la aplicación de los modelos basados en redes neuronales artificiales es el del diseño de arquitecturas adaptativas capaces de dotar de inteligencia a sistemas que operan en entornos dinámicos o de riesgo. Tal es el caso de la robótica industrial, donde cada vez más se precisa de sistemas inteligentes de control de comportamiento ante circunstancias inesperadas o de cambio de objetivo. En este proyecto se aborda esta problemática en aplicaciones de alcance y generación de planificación de trayectorias en robots industriales, tales como ABB-1400 o CRS-A255. De esta forma operaciones del tipo ensamblaje, soldadura, *pick-and-place*, etc. son abordadas y sus respuestas analizadas mediante la implementación de varias alternativas de alcance basadas en la interconexión de diferentes modelos de redes neuronales.

En este tipo de plataformas industriales se establece un compromiso entre la precisión de la tarea, que necesitaría de un sistema de control en lazo cerrado, y la velocidad de respuesta, que obliga al diseño de arquitecturas en lazo abierto. La solución que se aporta en este proyecto está basada en el diseño de sistemas híbridos adaptativos, basados en aprendizaje, interconectando sistemas lineales en lazo abierto (VITE, AVITE..) [1], sistemas no lineales en lazo cerrado (RBF, HRBF) [2,3] y redes autoorganizativas (ART, KOHONEN) [4] para clasificación, aprovechando las posibilidades que aporta cada uno de los modelos considerados: velocidad, robustez, flexibilidad, precisión, autoaprendizaje, etc. Una aportación importante de este proyecto es la de crear arquitecturas de control senso-motor, basadas en el comportamiento del sistema biológico humano. Estos modelos serán entrenados y aplicados a ambas plataformas robóticas.

Con el objetivo de verificar la validez de las estructuras diseñadas, aplicaciones no sólo de alcance de objetivos, sino también de seguimiento y persecución, han sido

desarrolladas y probadas, junto con un análisis comparativo del comportamiento de las arquitecturas propuestas, así como de los diferentes parámetros de los modelos de redes neuronales utilizados. Como aplicación inmediata de los modelos propuestos se aporta una solución al problema de las arquitecturas cerradas que normalmente presentan los robots industriales, cuyos controladores precisan de un elevado tiempo para la ejecución de comandos de movimiento, lo que impide que se puedan utilizar con redes adaptativas en entornos dinámicos. La interconexión de modelos en lazo abierto y lazo cerrado, junto con las técnicas de aprendizaje, permiten reducir el tiempo de proceso y posibilitar aplicaciones de alcance de objetivos en movimiento, considerando, también, la redundancia de movimientos existentes en determinadas plataformas robóticas a la hora de diseñar las estrategias de alcance.

La implementación de los modelos propuestos se ha realizado, en una primera fase, sobre el simulador virtual (*Virtual Robot Simulator*) que permite generar escenarios de movimiento con los diseños CAD de robots industriales reales. En esta plataforma se ha realizado la programación de las arquitecturas, el entrenamiento de los pesos de la red neuronal y la ejecución y testeo de las operaciones de generación de trayectorias para el alcance tanto de objetivos fijos como en movimiento. Posteriormente, estos resultados han sido implementados en un robot industrial real, con un cabezal estereoscópico para la realimentación visual de la posición del efector final y del objetivo. La verificación in-situ de los parámetros de velocidad y precisión ha permitido validar y comparar los diseños de las estrategias de alcance presentadas en este proyecto.

Por tanto, el *objetivo general* de este proyecto es el diseño de arquitecturas de aprendizaje basadas en la interconexión de diferentes modelos de redes neuronales para resolver el problema de la generación de trayectorias en robots industriales con arquitecturas cerradas. Posteriormente, el análisis de los resultados obtenidos va a permitir el ajuste de los parámetros de las arquitecturas diseñadas y la comparación entre ellas. Este objetivo general se ha subdividido en una serie de *objetivos específicos*, cuyas descripciones son las siguientes:

- Diseño de una arquitectura de alcance en lazo cerrado basada en modelos adaptativos no-lineales generados a partir de redes del tipo HRBF (*Hyper*

Radial Basis Functions), correspondiendo con la primera estrategia de planificación de trayectorias.

- Diseño de una arquitectura de alcance híbrida formada por la interconexión de redes autoorganizativas del tipo ART (*Adaptive Resonante Theory*) o KOHONEN y redes lineales trabajando en lazo abierto del tipo AVITE (*Adaptive Vector Integration to End-Point*), correspondiendo con la segunda estrategia de planificación de trayectorias.
- Programación en Visual C++ de ambas arquitecturas, haciendo uso de una plataforma de simulación virtual con los robots industriales ABB-1400 y CRS-A255. Uno de estos robots se utilizará posteriormente para la validación real de los modelos.
- Entrenamiento de ambas arquitecturas analizando la influencia de los diferentes parámetros de aprendizaje en el resultado final.
- Aplicación de ambas estrategias de alcance en los robots considerados, enfocando la experimentación en los aspectos:
    - Alcance de Objetos fijos
    - Alcance de objetos en movimiento
    - Alcance de objetos en presencia de perturbaciones en su posición
- Análisis de resultados
- Implementación de ambas arquitecturas en un sistema real con robots industriales ubicados en el laboratorio NEUROCOR (Neurotecnología, Control y Robótica) de la UPCT.

1. Introducción a la Planificación de Trayectorias

Uno de los procesos más estudiados en la literatura de control senso-motor es el de alcance de objetivos en el espacio. Muchos autores han abordado el estudio de este problema proponiendo modelos de redes relativamente simples que abarcan desde la percepción visual hasta el accionamiento de las articulaciones del brazo. Estos modelos describen en gran detalle el problema del alcance mediante una colección de redes neuronales cada una de las cuales realiza una subtarea, motivada por datos anatómicos, fisiológicos o psicológicos. En esta línea varios investigadores del Departamento de Sistemas Neuronales y Cognitivos (CNS) de la Universidad de Boston propusieron algunos tipos de redes diferentes, englobadas dentro de lo que se conoce como Teoría de la Dinámica Neuronal, que son capaces de llevar a cabo tareas tales como:

- Representación neuronal de un objetivo percibido visualmente dentro de un sistema de coordenadas centrado en la cabeza y su posterior transformación a un sistema de referencia fijo en el cuerpo.
- Formación de una trayectoria espacial desde la posición actual de la mano hasta el objetivo.
- Transformación de esta trayectoria espacial en órdenes a los actuadores de brazo (cinemática inversa).
- Generación de comandos a las articulaciones de los brazos para mantener su posición a pesar de los cambios en la tensión de los músculos (dinámica inversa).

Estos modelos han sido formulados prestando gran atención en la consistencia interna dentro las redes neuronales y entre ellas, es decir, que estén interconectadas para formar un sistema senso-motor integrado y que las representaciones usadas en las diferentes interfaces sean iguales o estén relacionadas. Además. Tienen la ventaja de que la solución propuesta para el alcance guiado visualmente es compatible con otras tareas motoras como por ejemplo la escritura. Otra de las bases de estos modelos es que

han surgido del análisis de los sistemas biológicos de control senso-motor, por lo que se denominan modelos neurobiológicos.

En la elaboración de estos modelos se ha puesto gran atención en el papel de los procesos de aprendizaje. Las propiedades mecánicas de los sistemas biológicos cambian con el tiempo, como por ejemplo las longitudes de los brazos y la fuerza de los músculos que cambian con la edad. Esto conduce a que determinados parámetros dentro del sistema de control senso-motor (por ejemplo, la activación de las conexiones neuronales) deben ser adaptativos y no preestablecidos al nacer. Además, la ejecución de una tarea en condiciones de restricción temporal, como por ejemplo el bloqueo de una determinada articulación, no requiere un nuevo aprendizaje para reestructurar el sistema de control motor, sino una adaptación. Las redes neuronales que se van a describir en lo sucesivo utilizan la información que se genera en un ciclo de acción-percepción para encontrar los parámetros que permiten al sistema adaptarse a nuevas situaciones.

Estos modelos llevan a cabo el alcance automáticamente, es decir, sin necesidad de conmutar a modos especiales de ejecución o sin requerir un nuevo aprendizaje para las diferentes condiciones de operación. Por tanto, los modelos de redes neuronales que se presentan utilizan únicamente la información que se genera durante los ciclos continuos de acción-percepción con el fin de aprender las transformaciones necesarias en un sistema extremadamente flexible capaz de ejecutar exitosamente una gran variedad de tareas bajo diferentes condiciones. Es por ello que su implementación sobre diferentes plataformas va a permitir la verificación de los módulos de aprendizaje propuestos.

Muchas de las tareas de movimientos en robótica, sobre todo de alcance y agarre, están definidos en sistemas de coordenadas que son diferentes del espacio de trabajo del actuador al que se le envían los comandos motores. Es por ello, que la planificación y el aprendizaje de trayectorias en el espacio de las tareas requiere una transformación apropiada de coordenadas desde el espacio de tareas al espacio del actuador con anterioridad a que los comandos motores sean calculados y enviados.

La transformación desde la planificación cinemática, que vienen dadas en coordenadas externas (coordenadas espaciales XYZ) en coordenadas internas del robot

(coordenadas motoras) es lo que se conoce como el problema del cálculo de la cinemática inversa del robot. Este problema se genera, fundamentalmente por el hecho de que las transformaciones inversas son, normalmente, muy difíciles de definir numéricamente puesto que parten de información poco precisa, con ruido o redundante. La siguiente figura muestra un esquema de un robot de 6 grados de libertad con la posición cartesiana del efector final o manipulador, junto con lo que se denomina el espacio de trabajo del robot, es decir, aquéllas posiciones 3D que pueden ser alcanzadas por el efector final para el diseño mecánico de ese robot concreto

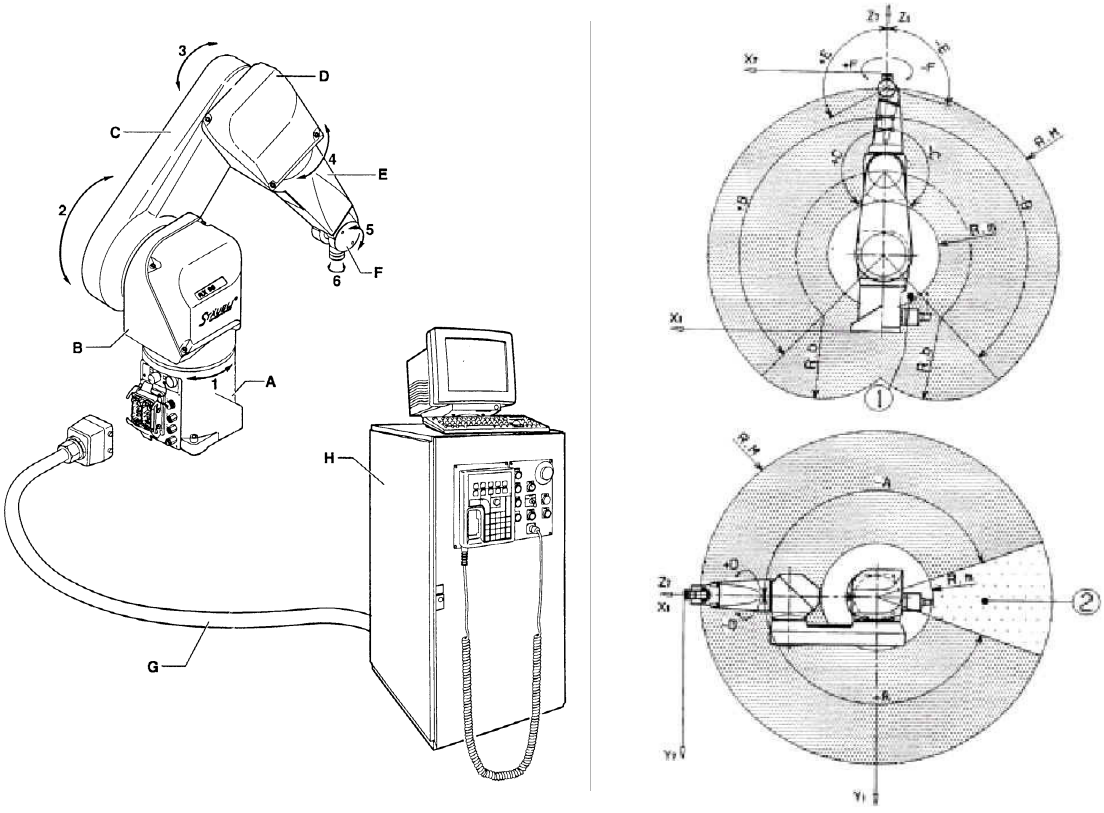

Figura 1. *Configuración de un robot con 6 grados de libertad y el espacio de trabajo 3D asociado al efector final.*

Por este motivo, el enfoque y la aproximación considerados en este trabajo trata de resolver este problema, usando redes neuronales artificiales para el aprendizaje de la cinemática inversa en robots redundantes para la planificación de trayectorias, es decir, aquéllos que pueden alcanzar el mismo objetivo (en coordenadas espaciales) mediante la configuración de diferentes posturas definidas en coordenadas motoras. Esta misma

filosofía, permitirá resolver problemas inherentes a sistemas físicos, como pueden ser el envejecimiento, bloqueo de articulación, ruido en las medidas de posición por deterioro del hardware electrónico de control, etc.

Las trayectorias en el espacio de tareas deben ser ejecutadas por medio de apropiados movimientos del manipulador o efector final de un brazo robot. Una posibilidad para su realización es el aprendizaje de un mapa de posiciones en el espacio de tareas, de forma que cada punto en el espacio 3D pueda tener su correspondiente configuración de las articulaciones del robot, cuya proyección sobre el espacio de tareas satisface dicha solución. Otra posibilidad es utilizar lo que se conoce como mapeo direccional desde la dirección de movimiento deseada en el espacio de tareas y las direcciones de movimiento en el espacio de los comandos motores del robot. En este proyecto se han considerado ambas alternativas, con el objeto de testear y evaluar la efectividad, robustez y flexibilidad de ambos modelos, cuyos mapas serán aprendidos mediante el uso de modelos de redes neuronales artificiales de diferente naturaleza. Ello va a permitir una compensación autónoma para las diferentes perturbaciones (ruido en las medidas, cambios en la posición del objetivo…) o restricciones impuestas al entorno de trabajo (bloqueos de articulación, relajación postural…). El aprendizaje de mapas asociativos para el control de movimientos de un robot manipulador está estrechamente unido a los controladores robóticos que utilizan lo que se denomina la matriz Jacobiana, que establece la relación matemática para el cálculo de la cinemática directa, es decir, a partir de las posiciones de cada una de las articulaciones que forman un brazo robot, estimar la posición espacial del efector final.

Así, la relación incremental entre el vector de coordenadas espaciales (Δx) y el vector de posiciones articulares del robot ($\Delta\theta$), se define por medio de la ecuación:

$$\Delta x = J(\theta)\Delta\theta \tag{1}$$

siendo Δx el vector de velocidad espacial del efector final del brazo robot (pinza, mano, herramienta…), $\Delta\theta$ el vector de velocidad de articulación y $J(\theta)$ es la matriz Jacobiana del robot, cuyos elementos dependen únicamente de la configuración de las articulaciones del robot en cada instante. Así, para obtener el vector de velocidad de las

articulaciones que permite ejecutar el movimiento del robot hacia una velocidad espacial deseada se puede calcular mediante la expresión:

$$\Delta\theta = J^{-1}(\theta)\Delta x \tag{2}$$

donde $J^{-1}(\theta)$ se conoce como la inversa de la matriz Jacobiana o el Jacobiano inverso. Para el caso en que lo que se tiene es un robot redundante, es decir que $\dim(\Delta x)<\dim(\Delta\theta)$, existen diferentes soluciones para le Jacobiano inverso. En este caso, $J^{-1}(\theta)$ corresponde con la pseudoinversa de la matriz Jacobiana, o inversa generalizada. Una solución para la inversa generalizada (conocido como Moore-Penrose) es aquélla que obtiene la mínima norma del vector de articulaciones que puede producir el vector de velocidad espacial deseado, es decir un movimiento rectilíneo.

En este proyecto, el cálculo de $J^{-1}(\theta)$ se obtiene mediante la aplicación de modelos de redes neuronales artificiales. El aprendizaje de la cinemática inversa de un robot manipulador es una alternativa muy eficaz cuando se da alguna de las siguientes circunstancias:

- El modelo cinemática del robot no es preciso o complicado.
- El modelo de cinemática no está disponible para el operador de la plataforma robot.
- La información espacial es proporcionada por cámaras de visión mal calibradas.
- La complejidad computacional de la solución es muy elevada y costosa.

Los métodos de aprendizaje son inherentes a los procesos de autocalibración, evitando los problemas debidos a las singularidades cinemáticas. La Figura 2 representa el esquema general de aprendizaje para la adquisición de los mapas asociativos que permiten la trasformación entre los valores incrementales de las coordenadas espaciales deseadas en las coordenadas de movimiento (rotación y traslación) de las articulaciones del robot manipulador.

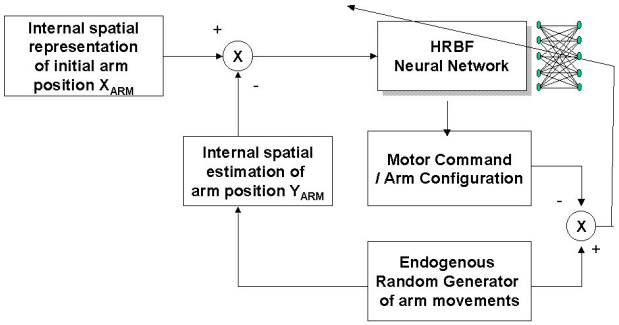

Figura 2. *Esquema de aprendizaje de trayectoria utilizando redes neuronales artificiales.*

Por otro lado, la Figura 3 representa el esquema del proceso adaptativo para el desarrollo o ejecución de la tarea basada en el aprendizaje descrito anteriormente. Durante esta etapa, el control del error y la información de la posición final deseada para el efector final viene dado por un sistema de visión artificial (en el caso de una plataforma real) o por la información propioceptiva del robot (en el caso de una plataforma de simulación). Esta información calcula de forma continua el vector diferencia en coordenadas del espacio de tareas entre la actual posición del efector final y la posición del objetivo (que puede ser fijo o móvil).

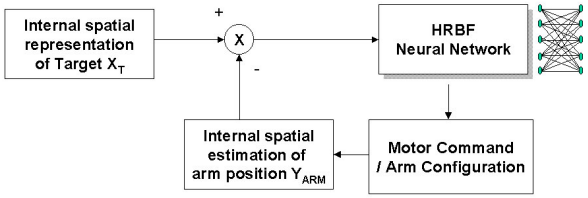

Figure 3.- *Esquema del desarrollo del sistema de alcance basado en redes neuronales.*

Este vector diferencia y la información propioceptiva sobre la actual configuración de las articulaciones del robot son las entradas de la red neuronal para calcular las rotaciones de articulación que permitirán al efector final alcanzar el objetivo deseado.

En el paradigma de aprendizaje desarrollado en este proyecto, los comandos de movimiento son generados en el espacio de los comandos motores (utilizando

entrenamientos aleatorios) de manera que el sistema aprenda un mapa entre el espacio de tareas de esos movimientos y los comandos de movimiento que realizaron dichas tareas. Posteriormente, este mapa inverso puede utilizarse para calcular y enviar comandos motores con el objetivo de alcanzar la tarea deseada (ver Figura 3). El proceso de aprendizaje se establece por medio de ciclos de acción-percepción en los cuales diferentes configuraciones del robot son alcanzadas a la vez que la información visual asociada con el desplazamiento del efector final es correlada con los incrementos de articulación que generaron tales desplazamientos. De esta forma, el aprendizaje de la cinemática inversa del robot no producirá posturas imposibles a diferencia del control clásico de robots basado en los modelos cinemáticas definidos por $J(\theta)$ en los que una postura irrealizable puede ser generada como resultado de la inversión de una matriz mal condicionada.

El mayor y principal obstáculo que se presenta en el proceso de aprendizaje de la cinemática inversa de un robot se basa en el problema de que la cinemática inversa de un robot redundante puede tener infinitas soluciones, es decir, diferentes posturas del mismo sistema pueden alcanzar el mismo objetivo. Por este motivo, el algoritmo de aprendizaje tiene que adquirir una solución particular y válida para la cinemática inversa. Sin embargo, diversos autores han demostrado que es posible obtener una solución válida para un sistema redundante si se utiliza un proceso de aprendizaje espacialmente localizado. Este tipo de aprendizaje, conocido como aprendizaje direccional, significa que *el sistema aprende una aproximación del Jacobiano inverso para cada postura del robot, de forma que esta aproximación pueda después utilizarse para otras posturas que guarden una relación de proximidad o vecindad con la postura anterior*.

Es importante, también resaltar que el mapeo direccional que ha sido utilizado para el aprendizaje de las diferentes alternativas planteadas en este proyecto, es localmente lineal, incluso para robots redundantes. Esto significa que si se considera una pequeña región en el espacio de las articulaciones, el conjunto de vectores de velocidad que producen un deseado vector espacial mantienen una relación que puede ser aproximada linealmente.

En este proyecto se analizan e implementen dos alternativas muy diferentes de alcance, cada una con diferentes aportaciones en cuanto a precisión, robustez, simplicidad, rapidez, etc. en las que esta técnica de aprendizaje direccional se ha utilizado. En concreto se trata de modelos basados en funciones de base radial (RBF) y redes autoorganizadas (SOM). En ambas, se utilizarán diferentes parámetros para diferentes regiones del espacio de trabajo y una posterior interpolación, más o menos suave en función del modelo diseñado, entre estos diferentes parámetros.

Finalmente, un aspecto importante en referencia a los sistemas que utilizan mapas direccionales es que pueden alcanzar los objetivos deseados de forma satisfactoria, incluso cuando estos mapas de aprendizaje contienen elevados niveles de error. Así, cualquier error residual que pudiera existir en el proceso de aprendizaje, como puede ser el suponer linealidad en áreas de trabajo demasiado grandes, no impedirá que el sistema alcance el objetivo deseado, aunque ello suponga un mayor número de acciones en el sistema (SOM) o un aumento de la curvatura de alcance (RBF).

2. Control Adaptativo

Las limitaciones de las herramientas de control clásico para operar en sistemas no lineales y el cada vez mayor interés en desarrollar robots cuyas formas, modelos de aprendizaje y comportamiento se asemejen a los humanos, son razones que explican el empleo de arquitecturas neuronales en el control de sistemas robots formados por estructuras antropomorfas de brazos, manos, cabezales, etc. En el ámbito del desarrollo de tareas de alcance, son las características de los sistemas concretos a los que se aplican y, en mayor medida, el modo de plantear el problema los elementos que determinan el modelo de control a utilizar. En lo que a la manera de enfocar el problema se refiere, las vías de investigación actuales son las siguientes:

1. Considerar el problema del alcance o planificación de trayectorias como el de obtener la relación entre el espacio visual (espacio de las tareas) y el espacio motor (espacio de las articulaciones), de modo que una vez establecida esa relación, ante un determinado patrón visual, el sistema sea capaz de alcanzar el objetivo deseado de forma que éste pueda ser, posteriormente agarrado o manipulado por cualquier instrumento de agarre como pueda ser una pinza, una mano robot o un instrumento de soldadura, de pintura…

2. Diseñar herramientas neuronales que permitan un control óptimo de la tarea de llevar el efector final del sistema robot a la posición deseada a través de la realimentación de señales sensoriales (normalmente visuales).

Desde el primer punto de vista, si se dispone de suficiente información como para establecer un sistema supervisado, es decir aquél que permite reajustar los valores de la cinemática inversa mediante la realimentación del error cometido, los modelos neuronales basados en redes RBF se plantean como una alternativa adecuada para solucionar el problema. Otra forma de solucionar este problema es el de la utilización de arquitecturas que permitan la implementación de modelos de aprendizaje no supervisado, como pueden ser las redes de Kohonen o las redes ART (*Adaptive Resonance Theory*) en combinación con modelos de redes neuronales de tipo lineal, como pueden ser las redes de tipo AVITE (*Adaptive Vector Integration to End-Point*).

El uso de redes neuronales como controladores en lazo cerrado es una consecuencia lógica del análisis del comportamiento humano, en el que el cerebro se constituye como un elemento capaz de controlar varios miles de actuadores (fibras musculares) en paralelo, en circunstancias de alta no-linealidad y en presencia de ruido. Así, en las siguientes figuras se muestran dos esquemas generales de lo que puede ser un lazo de control (o control en lazo cerrado) clásico y un sistema de control adaptativo basado en estructuras de redes neuronales.

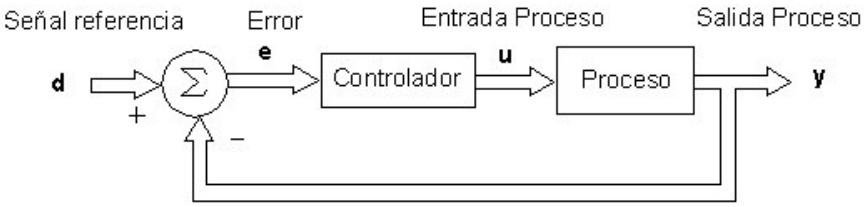

Figura 4.- *Esquema de lazo de control clásico.*

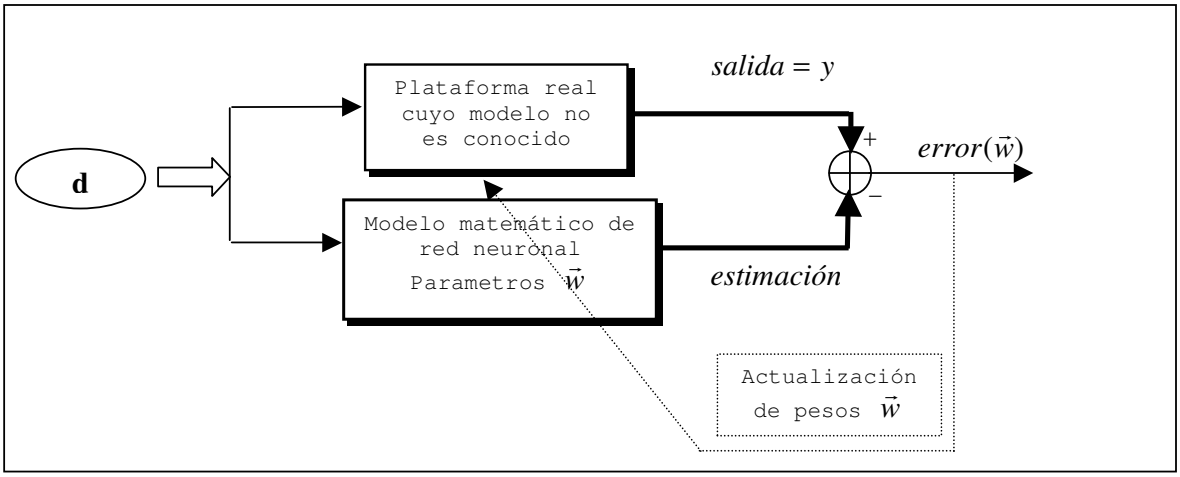

Figura 5.- *Esquema de control adaptativo con redes neuronales*

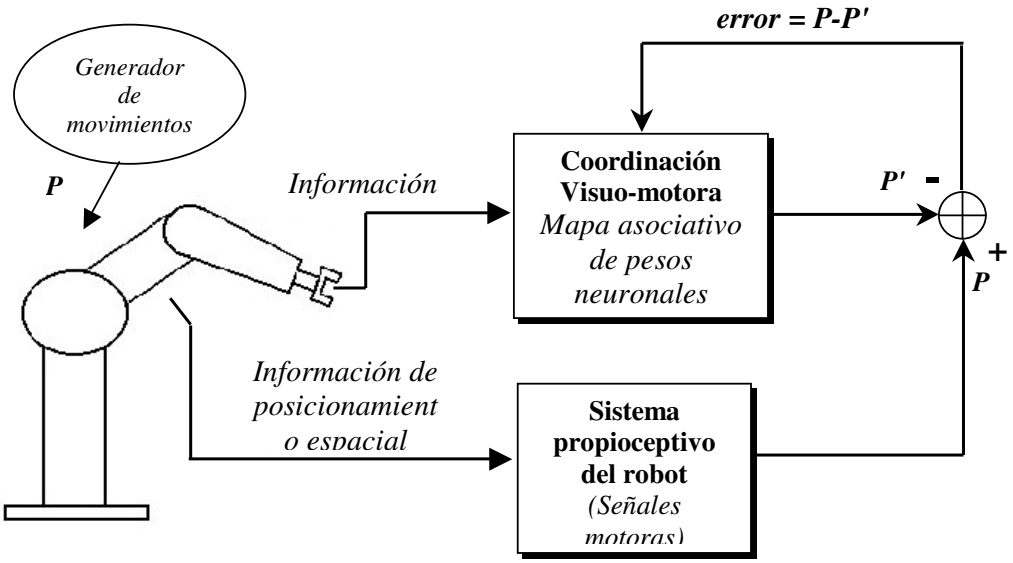

Figura 6.- *Esquema de control adaptativo sobre un robot manipulador*

Tal y como muestra la figura 6, en cada iteración, se incrementa la probabilidad de producir la salida óptima para cada valor de la entrada, a través del ajuste de los pesos de la red neuronal. Esta red neuronal se diseña de manera que su modelo matemático pueda ser un buen aproximador del proceso que se quiere aprender. Así, podrá ser lineal o no lineal, dependiendo de la relación propuesta entre la entrada y la salida del proceso. Esto supone un compromiso, puesto que para los *modelos lineales*, se obtiene una solución rápida para los pesos de la red pero con un error considerable. Sin embargo, para los *modelos no lineales* ocurre lo contrario, es decir, se puede obtener una mejor aproximación a cambio de necesitar mayor número de iteraciones para el aprendizaje de los pesos de la red. Finalmente, el correcto cálculo de los pesos, cuya ecuación de actualización dependerá del modelo de aprendizaje seleccionado dependerá de lo que se denominan algoritmos de aprendizaje, que son procesos iterativos de modificación de los pesos de manera que en cada iteración el error final obtenido entre la señal real y la estimada sea cada vez menor. En ese caso se dice que el algoritmo evoluciona en la dirección contraria del gradiente del error.

3 Modelos adaptativos no lineales

3.1 RBF (Radial Basis Functions)

Un problema de común aparición en el área de la matemática aplicada a la tecnología es el de estimar una función a partir, únicamente, de una muestra de pares de valores entrada-salida que se corresponden con dicha función. Es lo que se denomina *problema de regresión*. Como ya se ha apuntado, existe un modelo de red neuronal denominado RBF que es un excelente aproximador de funciones no lineales y que está basado en una relación entre la entrada y la salida del proceso a través de funciones de base radial, es decir, aquéllas que tiene cierta simetría con respecto a alguno de los ejes, tales como las funciones Parabólicas o Gaussianas. Estas funciones son conocidas como funciones de activación de la red neuronal. Las redes neuronales con funciones de activación radiales constituyen aproximadores universales mediante los que se puede resolver este problema de regresión.

Desde el punto de vista de las redes RBF, el problema de regresión se encuadra dentro de la categoría de *aprendizaje supervisado*, puesto que la función es *aprendida* a partir de una serie de ejemplos conocidos, mediante los que se pueden establecer funciones del error cometido en las estimaciones realizadas mediante la red, tal y como se ha mostrado y descrito en la Figura 6. De este modo, el conjunto de aprendizaje está constituido por una colección de pares de valores (vectores) de la variable independiente (entrada) y de la variable dependiente (salida). En este apartado se explicarán los modelos RBF como base para lo que posteriormente será el modelo definitivo que se ha empleado en este proyecto para resolver el problema de generación de trayectorias en lazo cerrado: el modelo HRBF. Así por ejemplo, considerando la función:

$$y = f(x) \tag{3}$$

La variable independiente es x y la dependiente es y. El valor de y depende, a través de la función buscada f, de las componentes de x. Es necesario tener en cuenta que, en el caso más general, ambas variables son vectoriales. El conjunto de entrenamiento se representa como:

$$T = \{x_i, \hat{y}_i\}_{i=1}^{p} \tag{4}$$

La razón para notar la variable y con el símbolo ^ es que se asume la presencia de ruido en la toma de muestras que da lugar al conjunto de entrenamiento. Es decir, el valor exacto de la salida y_i para una entrada x_i es desconocido. Esta es una de las ventajas de la utilización de las redes neuronales: incorporan la aparición de ruido al proceso de entrenamiento, al igual que lo hace un brazo humano cuando sufre de alguna anomalía en su funcionamiento que no impide aprender esa nueva configuración para realizar tareas de alcance de objetos.

Dado que sólo conocemos T, el modelo a emplear ha de ser necesariamente un modelo de *regresión no paramétrica*. Esto es debido a que no tenemos ningún conocimiento *a priori* acerca de la forma de la función a estimar.

Estos modelos se distinguen de los modelos *paramétricos*, en que la forma de la función es conocida, y sólo se trata de estimar el valor de ciertos parámetros. Un ejemplo de esto es el ajuste lineal de una serie de pares de valores:

$$f(x) = ax + b \tag{5}$$

Obsérvese que en el caso de regresión paramétrica, la dependencia entre las variables es conocida, no así el valor de los parámetros (*a* y *b* en la situación de relación lineal).

MODELOS LINEALES

Un modelo lineal para la estimación de una función *y(x)* tiene la forma:

$$f(x) = \sum_{j=1}^{m} w_j h_j(x) \tag{6}$$

El modelo *f* es, por tanto, una combinación lineal de *m* funciones básicas *h(x)*. La notación *w* para los coeficientes de la combinación lineal y *h* para las funciones básicas refleja el hecho de que en el ámbito de las redes neuronales tengamos pesos que caracterizan a las sinapsis entre neuronas (*weights*) y capas ocultas de neuronas (*hidden units*) entre la entrada y la salida del modelo.

Aunque existen muchas definiciones o criterios para clasificar la linealidad o no de una red neuronal, en general se suele definir un *modelo lineal* como aquél en el que la relación entre la entra y salida de la red es no lineal. Para el caso anterior, una relación lineal con *x* para la función *h(x)* daría lugar a un modelo lineal. En los *modelos no lineales*, también los parámetros de las funciones básicas pueden ir alterándose durante el entrenamiento o la relación entre la entrada y la salida es no lineal. Clásicos ejemplos de funciones básicas son:

- Ondas sinusoidales:

$$h_j(x) = \sin\left(\frac{2\Pi j(x - \theta_j)}{m}\right) \tag{7}$$

- Funciones logísticas

$$h_j = \frac{1}{1+\exp(b^T x - b_0)} \tag{8}$$

- Funciones radiales, objeto de nuestro estudio.

$$h(x) = \exp\left(-\frac{(x-c)^2}{r^2}\right) \tag{9}$$

De entre todas las funciones básicas que se puedan generar, existe un tipo de funciones que se denominan de base radial o **Funciones Radiales.**

La forma general es:

$$h(x) = \phi\left((x-c)^T R^{-1}(x-c)\right) \tag{10}$$

donde ϕ es la función básica (gaussiana, muticuadrática,...), c es el centro y R es la métrica. La distancia establecida según la métrica R entre el la entrada x y el centro c viene expresada según el término dado por la expresión (11), donde habitualmente se utiliza la métrica Euclídea, definida por la expresión (12):

$$(x-c)^T R^{-1}(x-c) \tag{11}$$

$$R = r^2 I \tag{12}$$

Los tipos de funciones básicas más comúnmente utilizados son:

-Gaussiana:

$$\phi(z) = e^{-z} \tag{13}$$

-Multicuadrática:

$$\phi(z) = (1+z)^{\frac{1}{2}} \tag{14}$$

-Multicuadrática inversa,

$$\phi(z) = (1+z)^{-\frac{1}{2}} \tag{15}$$

-Función de Cauchy,

$$\phi(z) = (1+z)^{-1} \tag{16}$$

En cualquier caso sólo se modifican los pesos atendiendo a una regla tendente a disminuir la diferencia entre el valor aproximado por el algoritmo y el valor conocido perteneciente al conjunto de entrenamiento.

La característica fundamental de las funciones radiales es que su respuesta aumenta (multicuadrática) o disminuye (Gaussiana) de manera monótona con la distancia a un punto central. El centro (c) y la forma (o anchura) de la función radial (determinada mediante r) son parámetros del modelo, fijos si éste es lineal, como se comentó en el apartado anterior. Considerando la métrica Euclídea, la expresión para una RBF Gaussiana queda expresada de la forma (17), donde 'r' es la desviación típica de la campana de Gauss o como una RBF multicuadrática (18):

$$h(x) = \exp\left(-\frac{(x-c)^2}{r^2}\right) \tag{17}$$

$$h(x) = \frac{\sqrt{r^2 + (x-c)^2}}{r} \tag{18}$$

Las RBF multicuadráticas dan una respuesta global. Por el contrario, las RBF Gaussianas son locales, esto es, son capaces de dar una respuesta significativa sólo en una determinada área de actuación en las proximidades del centro, puesto que debido a la aparición de la exponencial negativa con la distancia al centro de la Gaussiana, estas funciones tienen a anularse para distancias muy alejadas del centro considerado. Las figuras 7 y 8 muestran una secuencia de ajuste de una función senoidal con funciones Gaussianas:

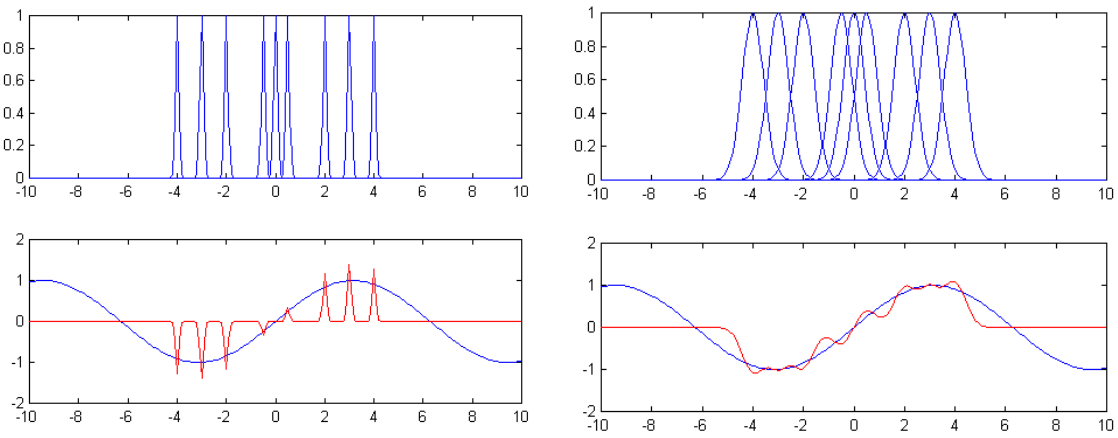

Figura 7.- *Ajuste de una función senoidal mediante M=9 funciones de base radial con centros en {-4,-3,-2,-1,0,1,2,3,4} y diferentes valores de 'r'. El valor de los pesos ha sido calculado mediante ajuste de mínimos cuadrados.*

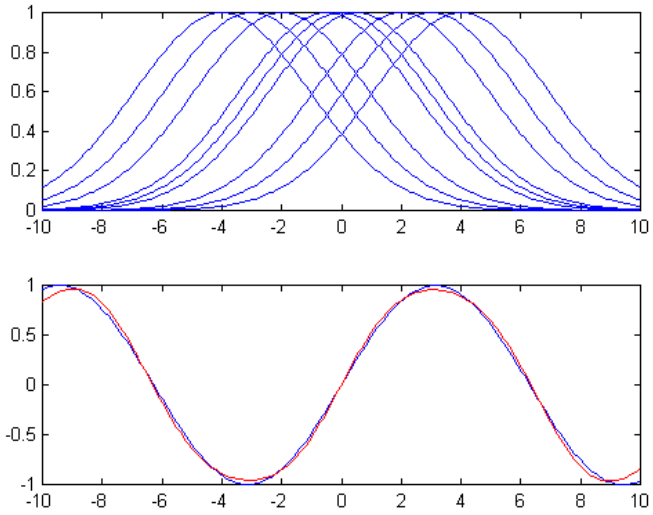

Figura 8.- *Ajuste de una función senoidal mediante M=9 funciones de base radial con centros en {-4,-3,-2,-1,0,1,2,3,4} para un valor óptimo de 'r'.*

REDES NEURONALES BASADAS EN FUNCIONES RADIALES

Como ya se ha comentado, las funciones radiales constituyen una de tantas clases funcionales que podemos utilizar como funciones de activación en una red neuronal. Por tanto, pueden ser empleadas en modelos lineales o no lineales. Sin embargo, tradicionalmente, las redes RBF han estado asociadas a estructuras neuronales de una sola capa oculta, con salida lineal, tal y como muestra la siguiente figura:

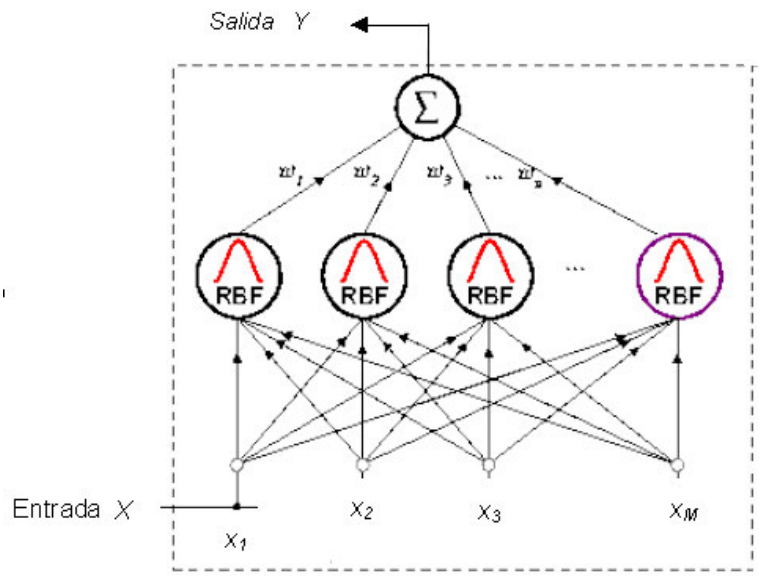

Figura 9.- *Configuración y estructura de una red neuronal RBF. Las incógnitas de la red son los pesos w, mientras que las funciones de base radial h(x) son fijas, con valores de 'c' y 'r' previamente definidos*

MÍNIMOS CUADRADOS

Según este principio, para obtener una buena estimación de los pesos, hay que minimizar el término de error cuadrático entre la salida deseada '*y*' y la realmente obtenida por la red neuronal *f(x)*, respecto a los pesos del modelo, de la forma:

$$S = \sum_{i=1}^{p} \left(\hat{y}_i - f(x) \right)^2 \tag{19}$$

Para ello es necesario derivar la función *S* don respecto a los pesos e igualar a cero para obtener la solución para el vector *w* que hace mínimo el valor de *S*. Este vector de denomina **vector de pesos óptimo.** En el caso de aplicar el principio de mínimos cuadrados a un modelo lineal la función a minimizar es *S*, teniendo en cuenta que en una red RBF la función *f(x)* se expresa de la forma:

$$f(x) = \sum_{j=1}^{m} w_j h_j(x) \tag{20}$$

Para obtener el valor óptimo del j-ésimo peso, comenzamos por diferenciar la

función de coste *S*, resultando

$$\frac{\partial S}{\partial w_j} = 2\sum_{i=1}^{p}\left(f(x_i) - \hat{y}_i\right)\frac{\partial f(x_i)}{\partial w_j} \tag{21}$$

Necesitamos calcular la derivada de $f(x_i)$ respecto a w_j, que a partir de la expresión (20) se obtiene:

$$\frac{\partial f}{\partial w_j}(x_i) = h_j(x_i) \tag{22}$$

Sustituyendo (22) en la ecuación (21) de la derivada de la función de coste, e igualando a cero para obtener el mínimo de la función, se llega a la expresión:

$$\sum_{i=1}^{p} f(x_i)h_j(x_i) = \sum_{i=1}^{p} \hat{y}_i h_j(x_i) \tag{23}$$

que podemos escribir matricialmente como:

$$h_j{}^T f = h_j{}^T \hat{y} \tag{24}$$

cuya representación se corresponde con la ecuación (25):

$$\begin{pmatrix} \mathbf{h_1}^{\mathbf{T}}\mathbf{f} \\ \mathbf{h_2}^{\mathbf{T}}\mathbf{f} \\ \vdots \\ \mathbf{h_m}^{\mathbf{T}}\mathbf{f} \end{pmatrix} = \begin{pmatrix} \mathbf{h_1}^{\mathbf{T}}\hat{\mathbf{y}} \\ \mathbf{h_2}^{\mathbf{T}}\hat{\mathbf{y}} \\ \vdots \\ \mathbf{h_m}^{\mathbf{T}}\hat{\mathbf{y}} \end{pmatrix} \tag{25}$$

Considerando las leyes del producto matricial, podemos expresar esta relación como:

$$\mathbf{H}^{\mathbf{T}}\mathbf{f} = \mathbf{H}^{\mathbf{T}}\hat{\mathbf{y}} \tag{26}$$

H es la denominada matriz de diseño:

$$\mathbf{H} = \begin{pmatrix} h_1(x_1) & h_2(x_1) & \cdots & h_m(x_1) \\ h_1(x_2) & h_2(x_2) & \cdots & h_m(x_2) \\ \vdots & \vdots & \ddots & \vdots \\ h_1(x_p) & h_2(x_p) & \cdots & h_m(x_p) \end{pmatrix} \tag{27}$$

y como

$$\overline{\mathbf{h}}_{\mathbf{i}} = \begin{pmatrix} h_1(x_i) \\ h_2(x_i) \\ \vdots \\ h_m(x_i) \end{pmatrix} \tag{28}$$

entonces podemos escribir:

$$\boldsymbol{f} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_p \end{pmatrix} = \begin{pmatrix} \overline{\boldsymbol{h}}_{\boldsymbol{1}}\hat{\boldsymbol{w}} \\ \overline{\boldsymbol{h}}_{\boldsymbol{2}}\hat{\boldsymbol{w}} \\ \vdots \\ \overline{\boldsymbol{h}}_{\boldsymbol{p}}\hat{\boldsymbol{w}} \end{pmatrix} = \boldsymbol{H}\hat{\boldsymbol{w}} \tag{29}$$

de manera que:

$$\boldsymbol{H}^T\hat{\boldsymbol{y}} = \boldsymbol{H}^T\boldsymbol{f} = \boldsymbol{H}^T\boldsymbol{H}\hat{\boldsymbol{w}} = \left(\boldsymbol{H}^T\boldsymbol{H}\right)\hat{\mathbf{w}} \tag{30}$$

y el vector de pesos óptimo queda constituido como:

$$\hat{\boldsymbol{w}} = \left(\boldsymbol{H}^T\boldsymbol{H}\right)^{-1}\boldsymbol{H}^T\hat{\boldsymbol{y}} \tag{31}$$

con lo que la red queda totalmente definida puesto que todos los parámetros y matrices de la ecuación (31) son conocidos excepto las incógnitas *w* que queremos calcular. La forma de calcular los pesos utilizando la ecuación (31) se denomina *aprendizaje por mínimos cuadrados* y su desarrollo matemático se describirá posteriormente.

3.2 HRBF (Hyperplane Radial Basis Functions)

3.2.1 Características del modelo

El modelo HRBF, al igual que los modelos basados en arquitecturas neuronales RBF, permite resolver el problema de la cinemática inversa en sistemas robóticos estableciendo un mapeo mediante el que se relacionan los espacios entrada (datos proporcionados por cualquier sistema de simulación o sensores externos, como por ejemplo lo de visión, como en el caso que nos ocupa) y salida (ángulos de giro de las articulaciones con los que el brazo robot se posiciona en la localización deseada).

El modelo HRBF se inscribe en el conjunto de aproximadores universales del que forman parte los algoritmos neuronales basados en funciones de activación radial explicados en el apartado anterior. Esto quiere decir que mediante el empleo de cualquiera de estos modelos se puede obtener la relación de transformación entre dos espacios θ (vector de ángulos de la posición del robot) y X (coordenadas 3D de la posición del efector final, en nuestro caso) cualesquiera. Sin embargo, el uso del HRBF presenta una serie de ventajas que se describen a continuación, y que motivan su utilización en tareas de generación de trayectorias de alcance, como la que ocupa este proyecto. Antes de nada es importante resaltar que este modelo constituye un sistema de control en *lazo cerrado* puesto que, como ya se verá, la activación de las funciones básicas o radiales depende de la posición actual θ del robot, por lo que el algoritmo neuronal debe ser calculado en cada posición de robot a medida que se acerca a la posición final deseada.

Por tratarse de un método iterativo, el costo computacional es menor que en los algoritmos clásicos basados en el cálculo de la inversa del Jacobiano, puesto que hay que realizar más operaciones pero más sencillas que para el cálculo del Jacobiano inverso (inversión de matrices…).

En el modelo HRBF, el aprendizaje se centra en los pesos característicos de las funciones básicas presentes en los distintos planos neuronales, aunque, gracias a la adaptabilidad del modelo, se pueden incorporar algoritmos para el aprendizaje de otros

parámetros que pueden ser críticos, como por ejemplo la posición de lo centros '*c*' o la desviación típica de las funciones Gaussianas '*r*' (que no es objetivo de este proyecto). Esta facilidad de adaptación redunda en una pronta convergencia hacia valores mínimos del error. Se trata de un algoritmo altamente no lineal, con lo que la resolución del problema de regresión está garantizada para cualquier tipo de función, empleando (en caso de ausencia importante de ruido) un número reducido de patrones de entrenamiento. Además, en HRBF no es tan crítica la colocación inicial de las funciones Gaussianas en segmentos bien definidos del espacio como en otros modelos neuronales de mapeo.

En el ámbito de las aplicaciones a la resolución de la cinemática inversa en sistema visuo-motores, conviene destacar que mediante el modelo HRBF podemos incorporar un control sobre la relajación postural de las articulaciones, de manera que la configuración final adoptada por el brazo a la hora de efectuar el agarre estará siempre constituida por ángulos o posiciones poco forzadas. Si bien esto no ha sido desarrollado y aplicado en este proyecto, sí que es una línea de investigación futura para aplicaciones con robot reales que carecen de este sistema de control de la postura.

3.2.2 Descripción del algoritmo

El controlador diseñado en base a este modelo neurobiológico permite una primera aproximación al alcance de precisión de un objeto cuya posición es cambiante a lo largo de un proceso. El problema de la redundancia en el posicionamiento del brazo robot se puede resolver mediante un control del *esfuerzo postural* (que no ha sido abordado en este trabajo) de cada articulación o en la forma en cómo este modelo aprende el mapa de correspondencias entre posiciones motoras y espaciales durante la fase inicial de aprendizaje.

El modelo HRBF de Guenther está basado en el concepto de red neuronal con funciones de base radial RBF (*Radial Basis Functions*). Está formada por varias subrredes de tipo RBF con una capa de entrada, una capa oculta y una capa de salida, cada una de ellas. Además tiene la posibilidad de incorporar un término de control de la relajación postural de las articulaciones del brazo robot presente en el proceso de agarre.

De esta forma es posible resolver, al mismo tiempo, el problema de la equivalencia motora en sistemas redundantes y el aprendizaje de la transformación cinemática inversa del sistema de alcance. La particular división del espacio de entrada permite que el sistema de aprendizaje inicial precise un número reducido de entrenamientos aleatorios. La siguiente Figura muestra el esquema general de la red HRBF.

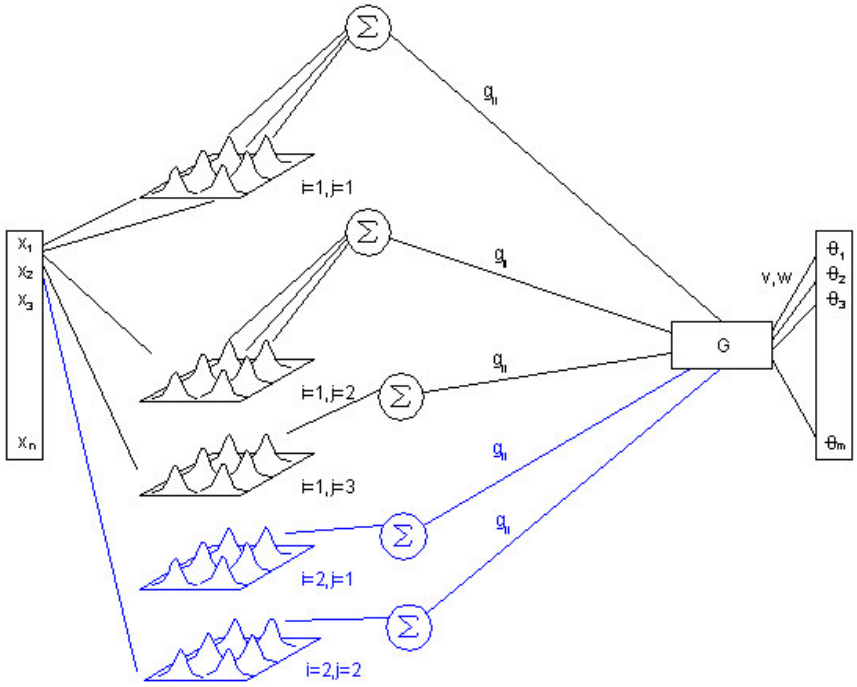

Figura 10. *Representación gráfica del modelo HRBF. La entrada X que se corresponde con la posición 3D deseada para el efector-final del sistema robot activa los diferentes planos de funciones RBF, cuyas salidas son sumadas y evaluadas por la función G para generar los comando motores θ de cada una de las articulaciones del brazo robot. De esta forma se resuelve la cinemática inversa. Como la solución depende de la posición actual del brazo robot, este modelo debe ejecutarse en lazo cerrado.*

Esta red es una versión de otros modelos de alcance para la estimación de la cinemática inversa del sistema cuya aportación es, por una lado suprimir la discretización del espacio de trabajo del robot, por otro, el aumento de la precisión en la transformación no lineal de ambos espacios utilizando funciones de base radial y, por último aportar una solución al problema de la equivalencia motora en sistemas

redundantes basada en el aprendizaje. Este modelo realiza un mapeo direccional entre los incrementos de las posiciones motoras de las articulaciones del robot $\Delta\theta$ y los incrementos de las posiciones espaciales Δx entre la actual posición *XYZ* del efector-final y la posición del objeto que se quiere alcanzar (posición deseada *X'Y'Z'*). El espacio de entrada (el de las componentes de *'x'*) se divide en varias redes o hiperplanos, cada uno representado a su vez por redes del tipo RBF, cuya estructura se ha mostrado en la figura 10. Cada nodo o centro de la capa oculta queda definido por su posición en la red (*'c'*), por los pesos de las conexiones neuronales (*'w'*) y por la desviación típica de las funciones gaussianas (*'r'*) a lo largo de las diferentes direcciones que forma la base del espacio de entrada.

Cuando se aplica una entrada a esta red la activación de los nodos adquiere un nivel que es función de la distancia entre ese patrón de entrada, la subrred a la cual se asigna y la posición de los nodos dentro de esa subrred. Esa distancia se mide como la distancia euclídea en las diferentes direcciones que vienen determinadas por cada una de las componentes del vector de entrada.

La salida obtenida para esa entrada realimenta al sistema de forma que se actualicen los pesos de las conexiones, la desviación típica de los centros y su posición dentro de la subrred que ha sido activada. Para este proceso de actualización se puede emplear cualquier algoritmo de aprendizaje como por ejemplo el algoritmo de mínimos cuadrados que se ha descrito anteriormente. Esta capacidad de aprendizaje de los diferentes parámetros de la red permite optimizar la transformación lineal entre los espacios de entrada y salida. Así, es posible organizar la posición y zona de influencia de los nodos de la red de forma que se obtenga una mayor aproximación en las zonas en las cuales se necesita una mayor precisión (mayor sensibilidad al error) y resolver el problema de la equivalencia motora.

Como ya se ha comentado en la introducción de este trabajo el mapa de correspondencias espaciales Δx y motoras $\Delta\theta$ se establece en función de la siguiente ecuación:

$$\Delta\theta = G(\theta)\cdot\Delta x + R(\theta) \tag{32}$$

La matriz *G* representa la estimación de la inversa del Jacobiano del sistema, mientras que la matriz *R* permite el control de la relajación postural de los elementos de $\Delta\theta$. Para este proyecto la matriz *R=0*, puesto que no se va a considerar la relajación postural, fundamentalmente porque la plataforma robot comercial utilizada para testear este modelo tiene internamente resuelto ese problema. Cada elemento de la matriz *G* representa una red o hiperplano $g_{ij}(\theta)$ definida, a su vez por una serie de pesos *w* y *v,* que representan la magnitud y la aproximación lineal de los datos en la región de influencia de cada una de las redes $g_{ij}(\theta)$, cuya expresión es:

$$g_{ij} = \sum_k \left(\frac{A_{ijk}}{\sum_k A_{ijk}} \right) \cdot \left(v_{ijk} + \sum_l c_{ijkl} \cdot w_{ijkl} \right) \tag{33}$$

siendo *k* el índice de cada función de base definida en cada una de las redes $g_{ij}(\theta)$, c_{ijk} es una medida de la distancia entre el valor de entrada y el centro μ de las *k* funciones de básicas de cada red y A_{ijk} es la activación de las funciones básicas. Una de las ventajas que incorpora esta red es la capacidad de definir diferentes comportamientos de la función básica con respecto a cada una de las dimensiones fijadas el vector de entrada. De esta forma es posible que, por ejemplo en un espacio de entrada bidimensional cada gaussiana (de desviación estándar σ) tenga un comportamiento exponencial diferente según la dirección de cada eje del espacio. El parámetro *l* define la dimensión de todas estas direcciones. Las ecuaciones que definen todos estos parámetros y el proceso de aprendizaje, utilizando el método de mínimos cuadrados, son las siguientes:

$$c_{ijkl} = \frac{x_l - \mu_{ijkl}}{\sigma_{ijkl}} \tag{34}$$

$$A_{ijk} = e^{-\sum c_{ijkl}^2} \tag{35}$$

y las ecuaciones de actualización de los pesos:

$$\Delta v_{ijk} = -\alpha \cdot \left(\frac{\partial H_1}{\partial v_{ijk}} \right); \quad \Delta w_{ijkl} = -\alpha \cdot \left(\frac{\partial H_1}{\partial w_{ijkl}} \right) \tag{36}$$

siendo α el paso de adaptación y H_1 la función de coste que hay que minimizar, cuya expresión es:

$$H_1 = \sum_i \left(\Delta\theta_{Bi} - \Delta\theta_i\right)^2 \tag{37}$$

El parámetro $\Delta\theta_{Bi}$ representa la posición real de las articulaciones obtenidas en una generación aleatoria de movimientos y $\Delta\theta_i$ representa la posición estimada a la salida de esta red neuronal. Sustituyendo las expresiones de (36) en (37) y utilizando la regla de la cadena en la operación derivada, se obtiene:

$$\Delta v_{ijk} = -2\alpha \cdot \left(\Delta\theta_{Bi} - \Delta\theta_i\right) \cdot \frac{\partial \Delta\theta_i}{\partial g_{ij}} \cdot \frac{\partial g_{ij}}{\partial v_{ijk}} = -2\alpha \cdot \left(\Delta\theta_{Bi} - \Delta\theta_i\right) \cdot \Delta x_j \cdot h_{ijk} \tag{38}$$

$$\Delta w_{ijkl} = -2\alpha \cdot \left(\Delta\theta_{Bi} - \Delta\theta_i\right) \cdot \frac{\partial \Delta\theta_i}{\partial g_{ij}} \cdot \frac{\partial g_{ij}}{\partial w_{ijkl}} = -2\alpha \cdot \left(\Delta\theta_{Bi} - \Delta\theta_i\right) \cdot \Delta x_j \cdot \left(h_{ijk} \cdot c_{ijkl}\right) \tag{39}$$

habiéndose definido un parámetro h_{ijk} de activación normalizada como:

$$h_{ijk} = \frac{A_{ijk}}{\sum_k A_{ijk}} \tag{40}$$

El modelo de red neuronal artificial HRBF descrito en este apartado ha sido planteado y analizado por Guenther (1997) para incorporar las ventajas de las funciones de base radial para la aproximación de funciones. En [4] se presenta este modelo y su aplicación genérica a un brazo robot para alcance de objetivos. Los resultados en simulación han permitido comprobar el comportamiento de esta red en aplicaciones al control específico de la relajación postural de un sistema robot redundante, cuyos resultados se muestran en la siguiente figura , si bien, como ya ha sido comentado no es esta adaptación del modelo, un objetivo de este proyecto, proponiéndose como líneas de trabajo futuro para robots redundantes.

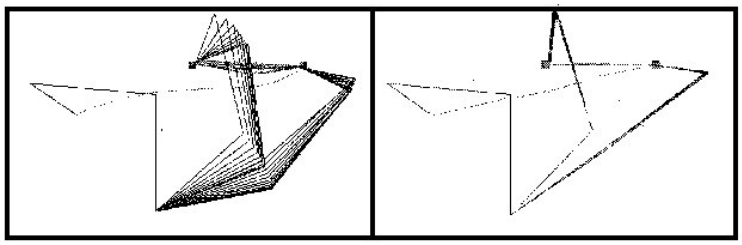

Figura 11. *Comparación del proceso de alcance de tres posiciones diferentes con un brazo robot de 3 grados de libertad utilizando el método de cálculo directo de la pseudoinversa (derecha) y la versión basada en el modelo HRBF (izquierda). Resultados obtenidos en los trabajos de Guenther.*

3.2.3 Fase de aprendizaje

Ante la entrada de un vector de *n* componentes *X* (que en nuestro caso será el vector 3D que mide la diferencia de posición entre el efector-final del robot y la del objeto), calculamos la activación en cada función básica según (38) y (39). A partir de la configuración de pesos presente en ese momento, se calcula la matriz *G*, mediante la que se puede calcular el error cometido en la obtención de la salida correspondiente a *X* como diferencia entre los valores de θ calculados mediante el modelo HRBF y vector θ^t conocido porque es la salida real que se sabe para la entrada correspondiente del conjunto de entrenamiento.

Este tipo de aprendizaje se conoce como aprendizaje supervisado pues la relación entre Δx y $\Delta\theta$ es conocida y debe ser proporcionada por cualquier tipo de sensor externo o interno. En nuestro caso, el vector $\Delta\theta$ lo proporciona directamente la lectura de las posiciones de los motores de cada articulación del brazo robot, mientras que el vector Δx lo proporciona un sistema de visión externa (en una plataforma real) o el propio software de simulación (en la plataforma simulada).

Este error es empleado en el proceso de actualización de los pesos neuronales *v* y *w*. El esquema de aprendizaje para las redes HRBF se muestra en la siguiente figura.

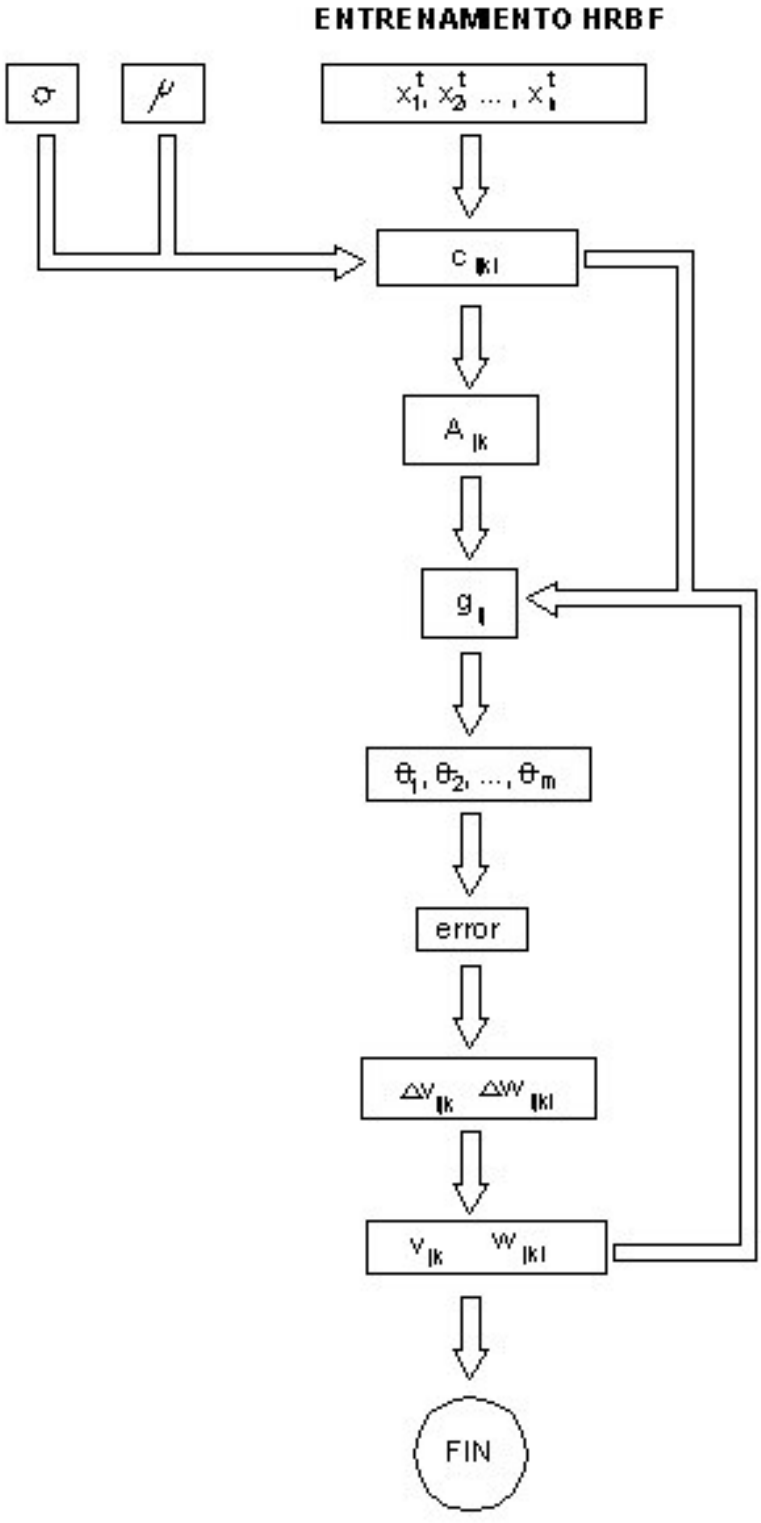

Figura 12. *Diagrama de bloques del proceso de entrenamiento de una red HRBF*

3.2.4 Fase de operación

Empleando los pesos obtenidos en el entrenamiento, se calcula la matriz *G* correspondiente a cada entrada, y haciendo uso de las expresiones (32) y (33), se obtiene la salida de la red, que se corresponderá con las posiciones angulares de cada articulación que hacen que el efector final se posicione en el objeto, tal y como muestra la figura 13.

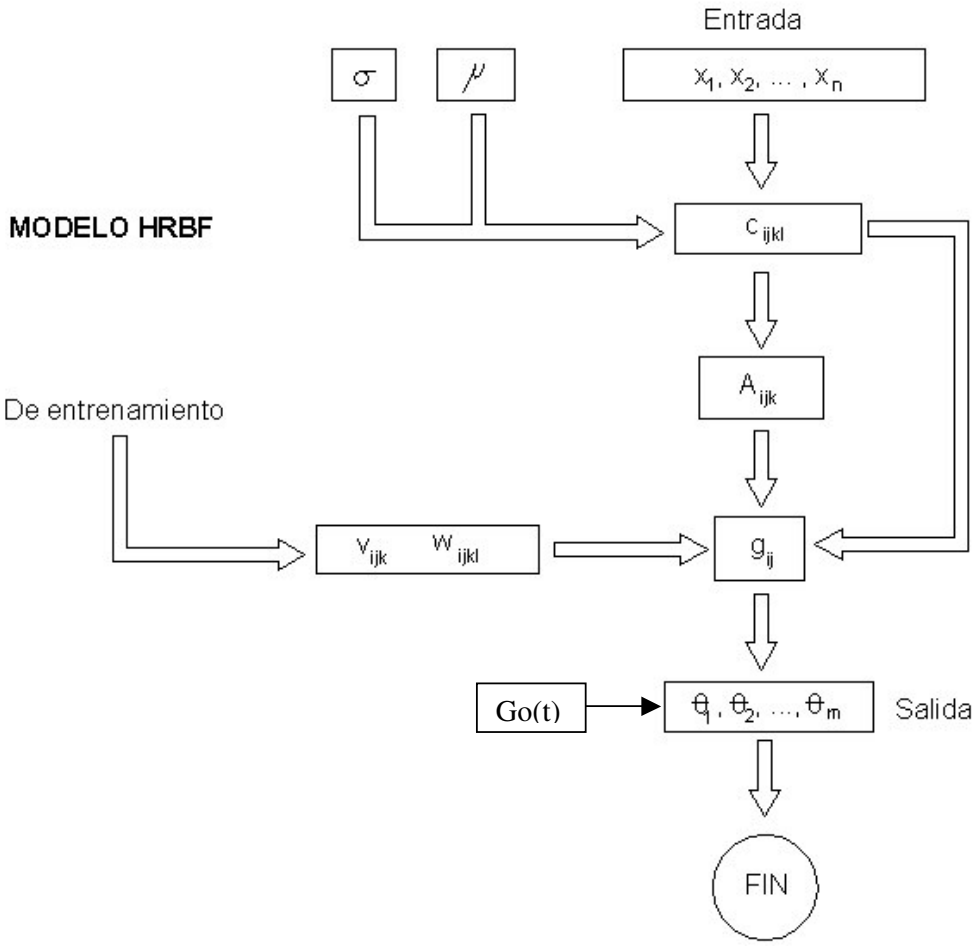

Figura 13. *Diagrama bloques de la fase de operación del modelo HRBF*

Como se puede observar, la salida se multiplica (o modula) por una señal Go(t) que permite limitar el movimiento del brazo robot para que se pueda controlar la velocidad del movimiento del brazo, así como el error de aproximación del robot hacia el objeto, puesto que, como ya se ha comentado, esta fase se realiza en lazo cerrado.

4. Mapas asociativos

4.1 Modelo VITE

Uno de los primeros modelos desarrollados para explicar la generación de los comandos de articulación θ basados en cómo los sistemas biológicos llevan a cabo una determinada trayectoria ha sido el modelo VITE (*Vector Integration To EndPoint*), cuyo esquema básico se representa en la figura 14.

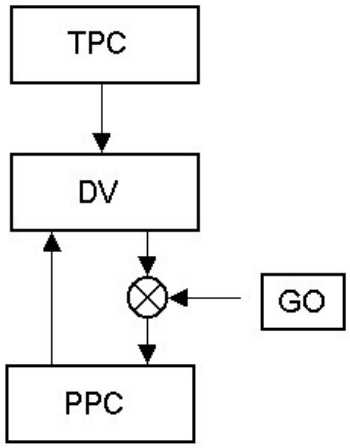

Figura 14. *Diagrama esquemático del modelo VITE. TPC (Target Position Command) es el comando de posición del objeto, PPC (Present Position Command) es el comando de posición actual y DV el Vector de diferencias entre ambas posiciones (actual y deseada en coordenadas de ángulos de articulación). La señal GO actúa como una señal moduladora de control de la velocidad del movimiento*

El comando de posición destino TPC está formado por una población de neuronas que codifican la configuración final deseada del brazo en coordenadas motoras. El comando de posición presente PPC representa la posición actual del brazo en coordenadas motoras, es decir, el ángulo de las articulaciones. El vector de Diferencias DV (*Difference Vector*), calcula la diferencia entre la posición deseada (TPC) y la configuración actual del brazo (PPC), por lo que representa el error de posicionamiento.

La señal GO actúa de compuerta, empieza valiendo cero, y para que el movimiento comience, debe tomar un valor positivo. Esta señal indica la velocidad a la que las neuronas motoras del nivel PPC deben reducir la señal recibida del DV. La realimentación permite autocontrolar el posicionamiento final. La forma adecuada de la señal GO hace que el perfil de velocidad del movimiento del brazo presente un suave crecimiento inicial y una deceleración igualmente suave al alcanzar la posición final deseada. Es decir, se obtiene un perfil de velocidad en forma de campana, aunque también puede ser definida como constante para simplificar los problemas de alcance. En este modelo, no es necesario especificar ninguna trayectoria para que el PPC se aproxime al TPC. Esta propiedad permite indicar un nuevo destino una vez iniciado un movimiento dirigido hacia un primer objetivo. El resultado es que al especificar un nuevo TPC cambia rápidamente la dirección de movimiento al nuevo destino, evitando así los problemas que presentan los controladores basados en la planificación previa de trayectorias.

El modelo VITE parte de la base de que tanto el TPC y el PPC están representados o expresados en el mismo sistema de coordenadas (coordenadas motoras), por lo que el DV opera con magnitudes expresadas en las mismas unidades (sin necesidad de ninguna conversión). Sin embargo lo más habitual en el problema de alcance robot es que la posición a alcanzar esté expresada en un sistema de coordenadas distinto al de las coordenadas motoras del brazo, como por ejemplo en coordenadas cartesianas o en coordenadas de visión, de forma que el TPC viene expresado en otras coordenadas como las espaciales.

4.2 Modelo AVITE

El modelo AVITE introduce esta particularidad, y permite calcular el DV consistentemente realizando la transformación de coordenadas mediante aprendizaje supervisado. Por lo tanto para transformar las coordenadas externas (espaciales XYZ) a coordenadas internas (motoras θ) existen conexiones adaptativas entre el TPC y el DV, es decir, pesos de aprendizaje que se calculan después de una fase de entrenamiento (denominada ERG). La figura 15 muestra el esquema del modelo AVITE.

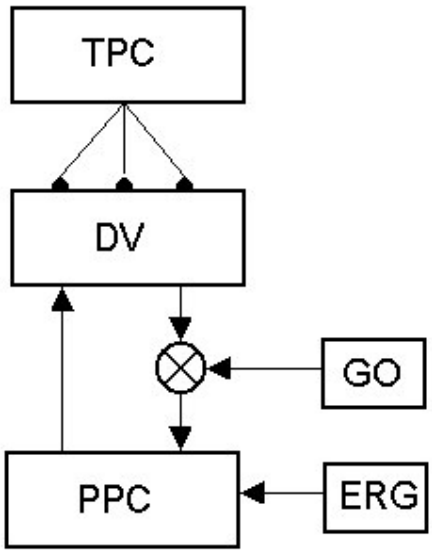

Figura 15. *Diagrama esquemático del modelo AVITE*

Para llevar a cabo el aprendizaje un generador aleatorio interno ERG (*Endogenous Random Generator*) produce señales o movimientos aleatorios del brazo robot (al igual que lo hace un recién nacido en su fase de aprendizaje de alcance de objetos) que son integradas por el PPC de forma que el brazo alcanza diferentes posiciones del espacio. Estas señales son recibidas a intervalos regulares de tiempo y se van almacenando en una matriz de entrenamiento. Al comienzo de cada movimiento de entrenamiento, el TPC es actualizado por el sistema de visión (o por la información suministrada por el SW de simulación), representando, en ese momento, el TPC y el PPC la misma posición del brazo aunque en coordenadas diferentes. Esta particularidad hace que el vector de diferencias no sea nulo. El aprendizaje tiene lugar haciendo tender el DV a cero conforme se va aprendiendo la transformación de coordenadas. El aprendizaje se estabiliza después de varias fases de movimiento, parada y adaptación de pesos, calculándose adecuadamente el DV y pudiendo funcionar el sistema como el modelo VITE. El aprendizaje puede ser continuo durante la fase de operación una vez conseguida una calibración inicial o dejar esa calibración inicial para toda la fase de funcionamiento del sistema (suponiendo claro está que no cambien los parámetros del robot).

Cuando el sistema está ya calibrado y TPC y PPC representan diferentes posiciones, el DV es diferente de cero. La generación de movimiento hace que la posición actual PPC se aproxime a la deseada TPC, con lo que el vector de diferencias converge a cero y se alcanza la posición deseada.

Aunque este modelo es muy sencillo, puesto que la relación entre la entrada y la salida es lineal (como se verá posteriormente), presenta una ventaja y a la vez un inconveniente para el problema de generación de trayectorias o de alcance con sistemas robóticos. La ventaja es la rapidez de cálculo, que hace posible realizar una tarea de alcance en tiempo real e incluso programar el algoritmo en dispositivos hardware dedicados tipo DSP (*Digital Signal Processor*). El inconveniente es que la relación entre Δx y $\Delta\theta$ en un sistema robot es, normalmente, no lineal por lo que el error de aproximar una función no lineal con un modelo lineal como el AVITE puede ser muy elevado. Esta situación justifica el trabajo realizado en este proyecto para el diseño de una red neuronal propia (denominada SOM-AVITE) basada en este modelo AVITE para garantizar una buena precisión de alcance y, a la vez, una elevada respuesta del modelo para que se pueda implementar en escenarios de tiempo real (como alcance de objetivos en movimiento o cambio brusco de posición del objetivo).

4.3 Etapas de Aprendizaje y funcionamiento

La estructura de aprendizaje del modelo AVITE se encuadra dentro de las que se han denominado "*aprendizaje supervisado*", y que ya han sido comentadas anteriormente, y cuyo esquema general se muestra en la siguiente figura:

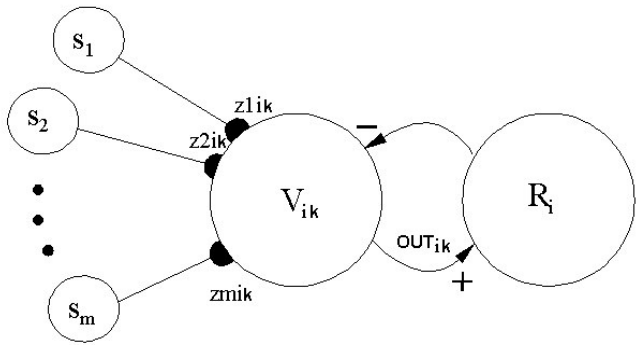

Figura 16. *Estructura neuronal para el modelo AVITE*

En esta figura, *S* corresponde al vector de entrada (Δx en nuestro caso), *Z* es la matriz de pesos (o coeficientes de la red neuronal), *R* es el vector de la salida conocida para esa entrada ($\Delta\theta$ en nuestro caso) y *Out* es el vector de salida que genera el modelo matemático AVITE ($\Delta\theta$'). Todo esto es para cada una de las neuronas *V* de la red AVITE, sabiendo que:

$$\begin{aligned} &\vec{\Delta}\theta = \mathbf{Z}\cdot\vec{\Delta}x \\ &(1\times p) = (p\times m)\times(m\times 1) \end{aligned} \tag{41}$$

Para el cálculo de los elementos de la matriz **Z** de este modelo AVITE existen fundamentalmente dos técnicas de aprendizaje: una adaptativa u *on-line* (método del descenso por gradiente) y otra no adaptativa u *off-line* (denominada ajuste por mínimos cuadrados). Ambas están basadas en la minimización del error cuadrático entre la solución obtenida por la red neuronal y por la que realmente debe dar, conocida a través del proceso de aprendizaje. Las ecuaciones de ambos métodos se describen a continuación.

Método de ajuste de pesos por el algoritmo **DEL DESCENSO POR GRADIENTE:**

De la figura 16 se deduce que el error de la unidad V_{ik} es:

$$e_i = \left(r_i - out_{ik}\right) \tag{42}$$

El error cuadrático total, es la suma de los errores al cuadrado de todas las unidades V_{ik} ($i = 1...n$):

$$E = \frac{1}{2}\sum_{i=1}^{n} e_i^2 \quad \Rightarrow \quad E = \frac{1}{2}\sum_{i=1}^{n}(r_i - out_{ik})^2 \tag{43}$$

Sustituyendo el valor de out_{ik} dado en la ecuación (41) en la expresión anterior, queda:

$$E = \frac{1}{2}\sum_{i=1}^{n}(r_i - \sum_{j=1}^{m} z_{jik}s_j)^2 \tag{44}$$

Ahora se puede calcular el gradiente del error cuadrático total respecto a los pesos, es decir la dirección en la cual se incrementa el error cuadrático, mediante la derivada de esta función con respecto a los pesos que se quieren calcular:

$$\frac{\partial E}{\partial z_{jik}} = -(r_i - \sum_{j=i}^{m} z_{jik} s_j) s_j \tag{45}$$

Para la minimización del error se obliga a que la evolución de los pesos sea proporcional al gradiente negativo del error cuadrático total, de forma que en cada iteración el error vaya decreciendo con respecto a la iteración anterior:

$$\frac{dz_{jik}}{dt} = \mu(-\frac{\partial E}{\partial z_{jik}}) \tag{46}$$

que expresado en forma discreta queda:

$$z_{jik}(t+1) = z_{jik}(t) + \Delta z_{jik}(t) \tag{47}$$

siendo,

$$\Delta z_{jik}(t) = \mu(-\frac{\partial E}{\partial z_{jik}}) = \mu(r_i - \sum_{j=1}^{m} z_{jik} s_j) s_j \tag{48}$$

El parámetro μ (entre 0 y 1) se denomina parámetro de convergencia o también *"learning rate"* e indica la velocidad con la que un algoritmo adaptativo como éste, hace converger los pesos hacia la solución final. Si este parámetro es muy pequeño, se necesitarán muchas iteraciones del algoritmo para alcanzar la solución final deseada, mientras que si es muy elevado, lo hará más rápido pero con oscilaciones sobre ese valor deseado.

Por lo tanto la evolución de forma iterativa de los pesos en cada intervalo de tiempo *k* será:

$$z_{jik}(t+1) = z_{jik}(t) + \mu(r_i - \sum_{j=1}^{m} z_{jik}(t) s_j) s_j \tag{49}$$

Ajuste de los pesos es mediante el **MÉTODO DE MÍNIMOS CUADRADOS:**

Este método al contrario del anterior no es iterativo. De la figura 16 se deduce que los incrementos de rotación r_i se calculan directamente desde los incrementos espaciales mediante la expresión lineal:

$$r_i = z_{1ik} s_1 + \cdots + z_{jik} s_j + \cdots + z_{mik} s_m \tag{50}$$

Ahora, el objetivo es estimar los pesos $z_{1ik},...,z_{jik},...,z_{mik}$, de forma que minimicen una función objetivo, que será un error cuadrático. Para ello se realizan p ensayos (ciclos de acción-percepción donde el módulo ERG genera vectores r de direcciones motoras aleatorias) durante p intervalos de tiempo, en los que se generan los siguientes datos: p direcciones motoras (vectores r) y p direcciones espaciales (vectores s). Los pesos estimados se expresan mediante la notación (51), siendo ε el error residual entre la solución estimada y la real.

$$r_i = \hat{z}_{1ik} s_1 + \cdots + \hat{z}_{jik} s_j + \cdots + \hat{z}_{mik} s_m + \varepsilon_i \tag{51}$$

Las p ecuaciones puestas en forma matricial:

$$\begin{bmatrix} r_i^1 \\ r_i^2 \\ \vdots \\ r_i^p \end{bmatrix} = \begin{bmatrix} s_1^1 & \cdots & s_j^1 & \cdots & s_m^1 \\ s_1^2 & \cdots & s_j^2 & \cdots & s_m^2 \\ \vdots & & \vdots & & \vdots \\ s_1^p & \cdots & s_j^p & \cdots & s_m^p \end{bmatrix} \begin{bmatrix} \hat{z}_{1ik} \\ \vdots \\ \hat{z}_{jik} \\ \vdots \\ \hat{z}_{mik} \end{bmatrix} + \begin{bmatrix} \varepsilon_i^1 \\ \varepsilon_i^1 \\ \vdots \\ \varepsilon_i^p \end{bmatrix} \tag{52}$$

Utilizando una notación abreviada:

$$R = S \times Z + \varepsilon \Rightarrow \varepsilon = R - S \times Z \tag{53}$$

El objetivo es minimizar la matriz de errores cuadráticos:

$$min \quad E = \sum_{h=1}^{p} (\varepsilon^h)^2 = \overline{\varepsilon}^t \overline{\varepsilon} \tag{54}$$

Para calcular los pesos que minimizan este error *E* se calcula la derivada:

$$\frac{dE}{dZ} = \frac{d}{dZ}(\bar{\varepsilon}^t \bar{\varepsilon}) = \frac{d}{dZ}((R - S \times Z)^t (R - S \times Z)) = 0 \tag{55}$$

De esta ecuación se llega a:

$$-2S(R - S \times Z) = 0 \tag{56}$$

La segunda derivada es positiva por lo que se ha obtenido un mínimo. Así pues operando en la ecuación anterior se llega a la solución final:

$$Z = (S^T S)^{-1} S^T R \tag{57}$$

De esta forma se obtienen unos pesos que minimizan el error cuadrático de una unidad (V_{ik}). Si se hace esto para todas las unidades V_{ik} (i=1...n) se obtendrán unos pesos que minimicen el error cuadrático total, al igual que con el método del descenso por el gradiente, pero de forma matricial y en una sola operación. Como es evidente el problema de este último método es que se precisa invertir una matriz, lo que en ocasiones dificulta su cálculo en procesos que requieren rapidez.

5. Redes Autoorganizativas

5.1. Mapas de Kohonen

Estas redes se denomina autoorganizativas o SOM (*Self-Organizing Map*) puesto que el cálculo de los pesos de la red no se hace de forma supervisada (midiendo el error entre la señal estimada y la real) sino no supervisada (o competitiva), por lo que el ajuste de los pesos se hace tendiendo en cuanta únicamente cuál es la entrada de la red y cómo están distribuidos los pesos en ese momento. En estas redes, la capa de salida está constituida por un conjunto de neuronas dispuestas según un entramado n-dimensional, tal y como se aprecia en la figura 17. Esta red se va a implementar en este

proyecto (tal y como se describe posteriormente) para realizar un *clustering* o segmentación espacial del espacio de trabajo del robot para reticular el espacio de igual modo que ocurre en los sistemas de telefonía celular. Utilizando este símil, sería como aprender de forma automática el radio y la posición de cada celda de forma que agrupe el mayor número de entradas de la red (posiciones del robot en este caso y núcleos de población de habitantes en el otro)

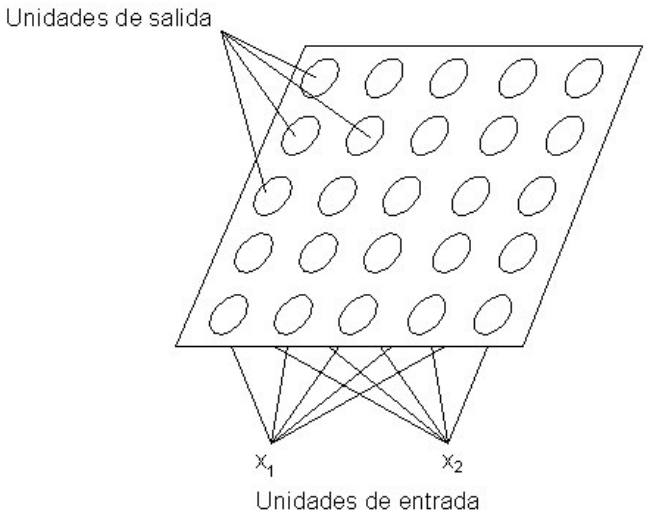

Figura 17. *Esquema general de una red Kohonen*

A través de un proceso de aprendizaje competitivo, en el que las propiedades del espacio de entrada determinan la ordenación final de las neuronas sobre el entramado, se genera un mapa neuronal del que se pueden aprovechar las siguientes propiedades:

- El mapa constituye una buena aproximación del espacio entrada a partir de un número reducido de pesos de la red. De esta manera, una de las principales ventajas de los algoritmos SOM es la de poder almacenar la información proporcionada por una gran cantidad de vectores del espacio entrada en una pequeña cantidad de pesos.

- La ordenación de las neuronas sobre el mapa determina dominios que representan regiones de características homogéneas en los vectores del espacio

entrada. Ésta es una propiedad especialmente útil a la hora de afrontar problemas de clasificación.

Las características del sistema determinan que se requiera un algoritmo de aprendizaje no supervisado o competitivo. Además, durante el entrenamiento, una entrada cualquiera de los patrones de entrenamiento (en nuestro caso serán posiciones de efector-final del sistema robot) da lugar a una determinada configuración activa de la red, que además tiene en cuenta el número de entrenamientos cada vez, de forma que las variaciones bruscas de los pesos ocurran durante el inicio del proceso de aprendizaje.

Al final de este proceso, la red de Kohonen puede ser utilizada para proporcionar al sistema de alcance una segmentación de las posiciones más frecuentes en las que va a operar el robot en coordenadas XYZ. Cada región tridimensional será diferente y estará caracterizada por la posición de lo que se denomina un centroide (que serán los parámetros que hay que aprender con esta red neuronal) y por su ámbito de influencia o fronteras denominadas de Voronoi. Las ecuaciones de aprendizaje de este modelo así como los pasos que se han de realizar, de forma general se describen a continuación:

- Inicialmente N centroides w_{ijk} son posicionados aleatoriamente en el espacio de trabajo 3D del efector final del brazo robot.
- A continuación, se realizan muchos movimientos aleatorios del brazo robot. Por medio de un sistema de visión estereoscópica y por un algoritmo de detección de imágenes se calcula la posición del efector final para cada uno de los movimientos del brazo robot. Este vector se denominará $\vec{P}$
- A continuación cada proceso de aprendizaje competitivo se inicia. Para ello se calcula la distancia euclídea entre la posición $\vec{P}$ y cada uno de los N centroides del espacio de trabajo. Al final, se selecciona el centroide ganador (que se denominará w_{ijk}*, escogiendo aquél que está mas cerca del punto $\vec{P}$.

- Finalmente, se actualiza el valor de todos los centroides (o pesos) utilizando la siguiente expresión de aprendizaje:

$$w_{ijk}(t+1) = w_{ijk}(t) + \alpha(t) \cdot \left[P - w_{ijk*}(t)\right] \tag{58}$$

siendo α(t) el "*leraning rate*" variable con el número de veces que se han actualizado las posiciones de los centroides

$$\alpha(t) = \frac{1}{t} \tag{59}$$

- Este proceso se repetirá varias veces hasta que se alcance la situación de convergencia, es decir, cuando la posición de todos los centroides permanece prácticamente inalterable ante sucesivas entradas.
- Las representación gráfica de las fronteras de Voronoi permitirá visualizar el proceso de generación de celdas en el espacio de trabajo 3D del robot.

Una vez que se ha realizado el proceso de aprendizaje descrito y que la posición de los centroides ha sido obtenida, la siguiente etapa será la del proceso de operación o ejecución de la red.

En esta fase, cuando se quiere clasificar un punto del espacio (en el cual se situará el objetivo a alcanzar por el robot), este algoritmo de Kohonen calcula la celda en la cuál se encuadra este punto, mediante la determinación del centroide más próximo a esa posición. Como la posición XYZ del centroide y la postura del robot necesaria para alcanzar dicho punto, son conocidos, en una primera aproximación, se puede posicionar el robot de forma que el efector final esté situado en el centroide de la celda a la que pertenece el objeto a alcanzar. Esta será la etapa de *alcance grueso.*

A continuación (como se explicará más tarde) otra red neuronal calculará la cinemática inversa de forma que la distancia que existe entre el objeto y el efector final se pueda reducir con una red lineal sencilla, puesto que si la distancia es pequeña una red lineal sí puede ser una buena solución al problema del cálculo de la cinemática inversa de un robot.

5.2. Redes ART2

De forma similar a cómo se comporta la red de Kohonen descrita en el apartado anterior, la red ART es un modelo autoorganizativo que tiene la habilidad de resolver el dilema que se conoce como de Estabilidad-Plasticidad, es decir aprender nuevos patrones sin olvidar la aportación de los anteriores. Su modelo de aprendizaje es también competitivo y básicamente se diferencia del modelo de Kohonen en que en la red ART el número de centroides *N* no se define inicialmente sino que la propia red los va creando según los vaya necesitando, en base a un criterio de *clustering*. La finalidad de esta red en este proyecto es idéntica a la de Kohonen, pero con la ventaja descrita de autogenerar el número de celdas. Se trata pues de una mejora al modelo anterior para el caso de problemas de alcance con robots. En realidad, el modelo original ART se diseñó para señales digitales y el ART2 desarrollado por Carpenter es una extensión para señales analógicas. De nuevo, cada región 3D o celda será diferente y estará caracterizada por la posición de su centroide y por la frontera de Voronoi, implicando la configuración de la dimensión final de cada celda. La estructura del modelo ART2 permite controlar el número final de celdas *N* en las que se divide el espacio de trabajo, controlado por lo que se denomina parámetro de vigilancia ρ y por el número de entrenamientos de la red. La estructura de la red ART2 se representa en la siguiente figura 18.

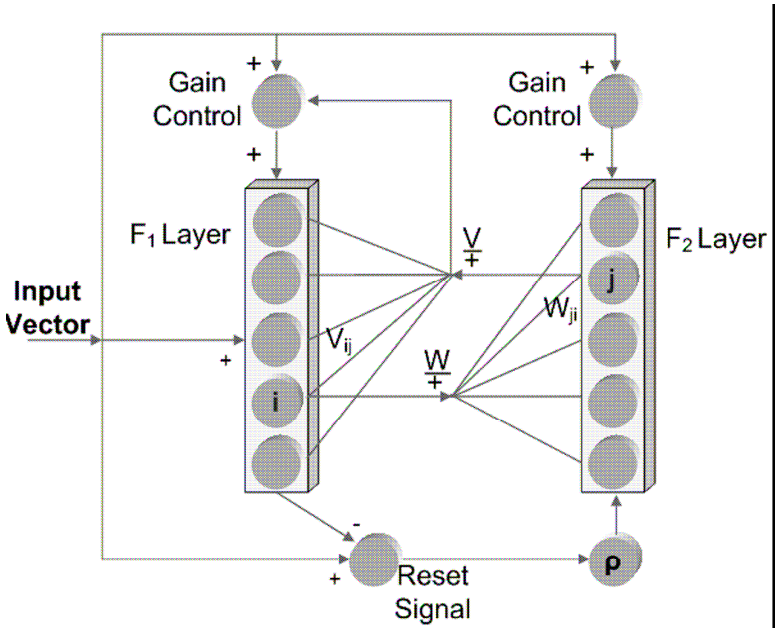

Figura 18. *Estructura de la red ART2*

Como se puede ver en la figura anterior, la capa de entrada F_1 tiene la misma dimensión que el vector de entrada, que en nuestro caso será un vector XYZ de posición del efector final en cada movimiento aleatorio. En la capa de salida F_2 se determina la posición de los centroides que se van generando y de los que se van moviendo de posición. Así, las neuronas (centroides) de la capa F_2 son los patrones que deben ser clasificados, W_{ji} son los pesos de las conexiones de los pesos hacia delante (*feed-forward*), mientras que V_{ji} representan los pesos de las conexiones hacia atrás (*feed-back*).

Otros parámetros fundamentales en esta red son el *Control de Ganancia* utilizado para asegurar la estabilidad de la red por control de inhibición o activación de las capas F_1 y F_2; la *Señal de Reset* utilizada para anular el nivel de asociación del centroide más cercano al patrón de entrada cuando se genera un nuevo centroide y, finalmente, ρ es el mencionado parámetro de vigilancia que decide cada vez que llega un patrón a la red, si se le asigna a un centroide (o celda) directamente o está lo suficientemente lejos como para crear una nueva celda, donde ese patrón será inicialmente el centroide.

En la *fase de aprendizaje*, de nuevo se generarían movimientos aleatorios del brazo robot, para detectar mediante un sistema de visión o con el propio simulador las posiciones XYZ del efector final. Por ejemplo, consideremos para una determinada postura del robot, el vector P_{xyz} como la posición del efector final. Este vector será la entrada de la red ART2. La celda o centroide ganador definido por W_{ij}* es seleccionado a través del criterio de ser el más cercano a P_{xyz}. Entonces, el valor de cada peso asociado a estos centroides (que han sido definidos aleatoriamente tanto en número como en valor) se actualizará en cada iteración k, según la expresión de aprendizaje definida por:

$$w_{ji}(k+1) = v_{ij}(k) = \frac{e_i(k) + w_{ji}(k) \times N_j(k)}{N_j(k) + 1} \tag{60}$$

siendo $e_i(k)$ la componente *'i'* del vector de entrada P_{xyz} y N_j representa las veces que la neurona (o centroide) *'j'* de la capa F_2 ha sido resultado ganadora hasta ese momento. Este proceso se irá repitiendo hasta que se alcance la situación de convergencia al igual

que pasaba en las redes de Kohonen, es decir, cuando ya no se muevan los centroides significativamente o cuando no se generen más.

La misión del parámetro de vigilancia es comparar, en cada iteración, la distancia euclídea entre la posición del nuevo patrón de entrada y la del centroide más cercano o centroide ganador. Esta distancia se compara con el valor del parámetro de vigilancia ρ. Si es mayor, la red decidirá generar una nueva celda o centroide, y si es menor, actualiza según la ecuación anterior el valor (o posición) del centroide ganador. Una vez se han generado todas las celdas el siguiente paso es idéntico al descrito para la red de Kohonen, es decir, calcular la posición del objeto, determinar a qué celda corresponde y mover rápidamente el robot hacia el centroide de esa celda, pues esa postura es conocida. El hecho de añadir la red ART2 a una operación de alcance con un robot, permite generar este *clustering* tridimensional con independencia del robot elegido y sin necesidad de conocer su modelo cinemática interno.

6. Estrategia de alcance basada en modelos HRBF

En esta apartado se describirá la aplicación concreta que se ha realizado del modelo HRBF para su adaptación al problema de cálculo de la cinemática inversa, abordado en este proyecto. Para ello se emplearán las ecuaciones descritas en el apartado 3.2).

Una vez que se inicia el proceso de aprendizaje, cada movimiento aleatorio del robot θ genera un incremento de rotaciones de articulación $\Delta\theta^R$, cuyo valor es medido por los sensores de posición o encoders de los motores de cada articulación del brazo robot. Ello provoca un desplazamiento del efector final del brazo robot Δx, de dimensión (1x3) y cuyo valor es suministrado por un algoritmo de procesamiento de imágenes en color implementado en el sistema de visión estereoscópica (en el caso de la plataforma real). Esto se repite para cada entrenamiento y, según las ecuaciones descritas anteriormente, es posible aproximar el Jacobiano inverso del robot. Para ello,

las posiciones de los centros y de las varianzas de las funciones Gaussianas se han considerados constantes para todo el proceso. La figura 19 muestra un esquema del funcionamiento del modelo HRBF diseñado, siendo Go(t) la señal que modula la velocidad de cómo el brazo robot se va acercando en cada iteración *'t*' hacia su objetivo, mientras que la figura 20 muestra el detalle de los parámetros internos del modelo HRBF.

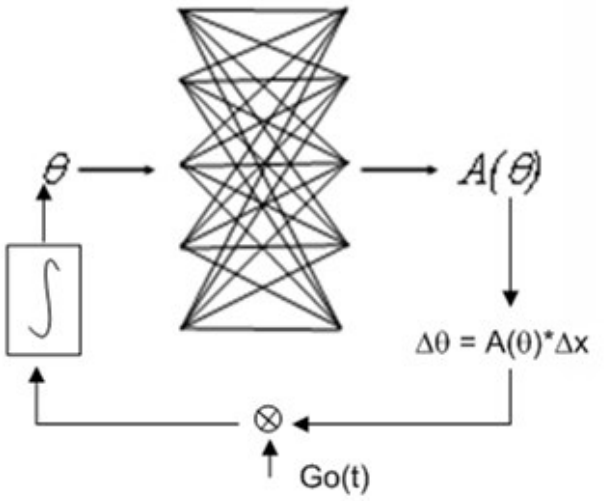

Figura 19. *Esquema del funcionamiento de un sistema de alcance robot con la estrategia en lazo cerrado basada en el modelo HRBF*

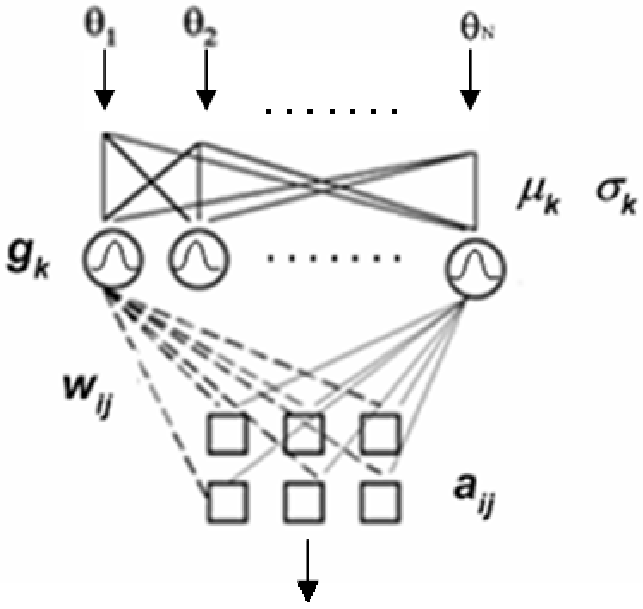

Figura 20. *Parámetros internos del modelo HRBF*

Como ya se ha comentado, la función Go(t) multiplica las rotaciones de las articulaciones que han sido calculadas por las red HRBF. Cuando esta señal es positiva (no puede ser cero) los movimientos de efector final se realizan de forma proporcional al valor de esta señal en ese instante, por lo que $\Delta\theta = \Delta\theta \cdot Go(t)$. En los experimentos realizados en este trabajo, esta función se ha considerado con un perfil del tipo:

$$Go(t) = \frac{a \cdot t^3}{b + t^3} \tag{61}$$

7. Estrategia de alcance híbrida

7.1. Descripción y ventajas del modelo híbrido

Uno de los principales *'topics'* en robótica reside en el problema de resolver la cinemática inversa en sistemas robóticas visuo-motores con brazos redundantes para aplicaciones de alcance, sobre todo en tiempo real (caso típico de alcanzar objetos en movimiento). La mayoría de los sistemas propuestos están basados en sistemas de control en lazo cerrado como el anterior HRBF. Estos sistemas son altamente dependientes de las características de precisión del sistema de visión, a la vez que necesita hacer el seguimiento de la trayectoria entera del efector final del brazo robot. Aunque estos sistemas están siendo continuamente empleados en robótica y permiten obtener buenos resultados, la coordinación visuo-motora del sistema biológico humano no requiere el seguimiento visual continuo de las articulaciones, puesto que la información propioceptiva o interna del movimiento del brazo ya ha sido aprendida en los primeros años de crecimiento, en base a movimientos aleatorios de los brazos (ciclos de acción-reacción).

El principal objetivo de esta novedosa estrategia propuesta en este proyecto es dar una solución al cálculo de la cinemática inversa en robots de alcance, sin el conocimiento de las propiedades físicas internas del brazo robot, tales como la longitud de las articulaciones, y umbrales de recorrido para cada articulación. Una de las dificultades de este trabajo ha sido la de obtener el mapa completo que asocia todas las

posturas del brazo robot con la correspondiente posición del efector final, de forma que se pueda evitar tener un sistema que trabaje en lazo cerrado.

La solución propuesta ha sido la de dividir, de forma autónoma para cada robot, el espacio de trabajo 3D en pequeñas estructuras o regiones que se han denominado *celdas de aprendizaje*. El modelo completo ha sido desarrollado basándose en la idea de resolver simultáneamente el problema de la precisión en el alcance y la velocidad del movimiento por medio de la interconexión de dos redes neuronales: una SOM y otra lineal.

Como resultado, la estrategia de alcance propuesta es capaz de combinar información visual (sobre la posición del objeto y del efector final del brazo robot) y la información propioceptiva del robot de forma que se puedan realizar tareas tales como seguimiento de trayectorias durante la cual sólo en la última fase de la tarea de alcance se produce el control realimentado (tal y como lo hace el sistema humano). Esta arquitectura ha sido implementada en un robot industrial (así como en plataformas de simulación) para garantizar su efectividad y sus múltiples capacidades de robustez, flexibilidad, rapidez…incluso cuando se producen cambios en la posición del objeto o movimientos del mismo.

Es por ello que, si bien la estrategia anteriormente propuesta HRBF para resolver el problema de alcance tiene la ventaja de obtener una buena precisión (esencial para tareas posteriores de agarre y manipulación), sin embargo tiene el inconveniente de operar durante toda la tarea en lazo cerrado (sistema realimentado), lo que ralentiza el proceso mucho y lo limita a alcance de objetos casi estáticos.

Sin embargo, en este proyecto se ha propuesto esta estrategia completamente innovadora que permite obtener un buen comportamiento en ambas facetas: precisión y rapidez. Para ello se ha pensado en interconectar dos redes completamente diferentes: una SOM (del tipo Kohonen o ART2) para crear celdas tridimensionales cuyas posiciones centrales (centroides) son conocidas por el robot (alcance grueso), y otras AVITE para resolver el problema de alcance en un pequeño tramo final (alcance fino). Esto permite dotar al sistema de alcance de un proceso anticipatorio en la planificación de la trayectoria, debido a la rapidez del primer movimiento del robot. De esta forma, el

movimiento se produce rápidamente sin necesidad de un control en lazo cerrado y, posteriormente, se ajusta la posición mediante un sistema lineal (que puede estar también realimentado). Esto es válido para robots redundantes, pues no se impone ningún tipo de restricción al sistema de aprendizaje. Lo más importante es que el sistema no precisa de un conocimiento previo del diseño mecánico ni características físicas ni de movimiento del brazo robot, puesto que la cinemática inversa es aprendida por esta red de forma natural. Este modelo, finalmente, aporta la característica importantísima de estar inspirada en cómo el sistema biológico humano realiza las operaciones de alcance. Esta arquitectura integra diferentes algoritmos, cada uno de ellos ejecutando diferentes tareas. La consistencia de la comunicación entre ambas redes garantiza la robustez final de la arquitectura completa.

7.2. Arquitectura SOM-AVITE

Como ya se ha comentado, la estructura de esta estrategia de alcance esta basada en la interconexión de dos modelos neuronales los cuales se ejecutan de forma secuencial (SOM primero y AVITE después) y que dan como resultado una proyección del efector final sobre la posición del objeto a agarrar. La base de este sistema de control consiste en discretizar, por medio de posiciones aleatorias del brazo robot, el espacio de trabajo del efector final en celdas de aprendizaje de dimensiones y posiciones (definidas por un centroide o centro de celda), en cuyo centro la posición o postura que debe realizar el brazo robot para alcanzar dicha posición es conocida.

A partir de ahí, el modelo neuronal autoorganizativo SOM (bien sea Kohonen o ART2) recibe la posición 3D del objeto del sistema de visión y comienza un proceso competitivo para seleccionar la celda ganadora (*módulo estimación de distancia*), es decir aquélla a la que se asigna la posición del objeto a alcanzar y, de esta forma, alcanzar en un primer paso (alcance grueso) la posición más cercana del brazo robot en el cuál el error de alcance ya es pequeño. Esta posición es el centro de la celda ganadora.

Después, en una segunda fase, el modelo AVITE permite obtener un mapa de pesos neuronales para cada celda que asocia posturas del robot con posiciones del efector final. Esto es, en definitiva, el cálculo de la matriz **Z** de la ecuación (57). Esto permitirá un rápido decrecimiento (en unos pocos movimientos en lazo cerrado mediante el módulo *DV compensation*) de la diferencia entre la posición del objeto y la actual del efector final (vector DV). Esta aproximación es realizada por la sencilla red AVITE que aún siendo lineal permite realizar con precisión esta última maniobra de alcance puesto que al ser tramos cortos dentro de una celda, se permite una buena aproximación lineal. Todo lo que se acaba de exponer queda explicado en el esquema general de este modelo híbrido SOM-AVITE que se presenta en la siguiente figura.

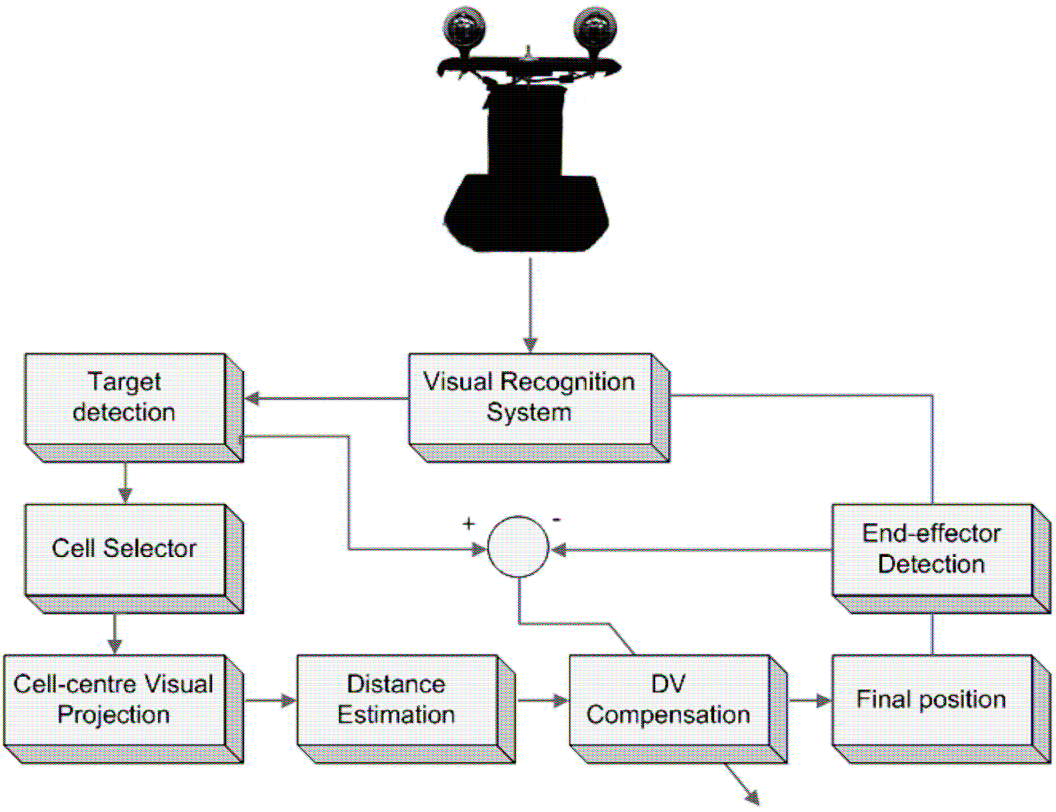

Figura 21. *Esquema general del modelo híbrido SOM-AVITE para resolver problemas de alcance rápido y con precisión. Está formada por dos modelos para discretizar el espacio de trabajo 3D del robot (modelo SOM), de forma que se en la primera maniobra se obtenga una rápida y buena aproximación, y para compensar la distancia espacial entre la posición después de la primera maniobra y la del objeto (red AVITE)*

La siguiente figura muestra una representación gráfica de la discretización del espacio de trabajo del brazo robot mediante la aplicación de las redes tipo SOM.

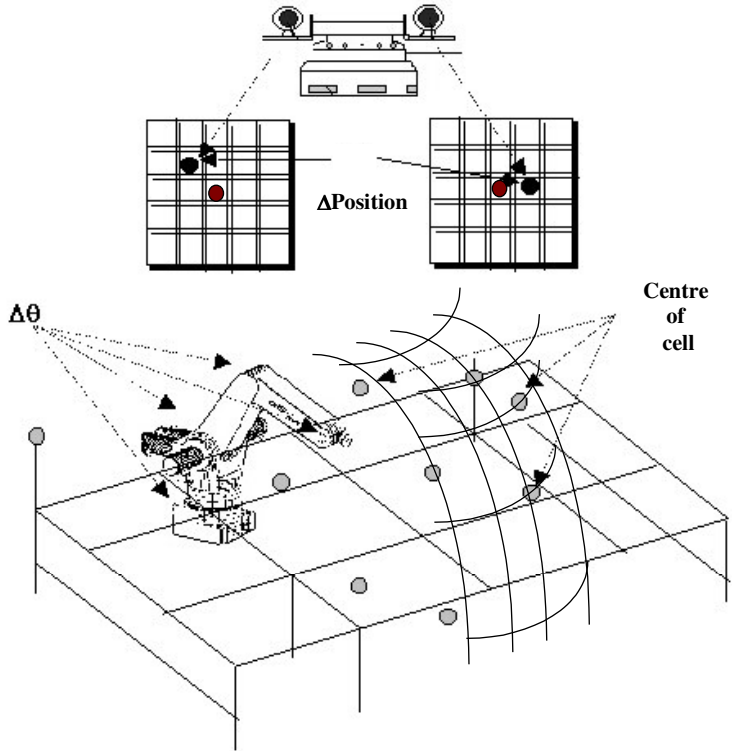

Figura 22. *Escena de discretización del espacio de trabajo para poder aplicar posteriormente el modelo AVITE. El espacio 3D es dividido en pequeñas celdas de aprendizaje de forma automática para que, posteriormente, el modelo AVITE obtenga diferentes mapas (o matrices* **Z***) en cada una de las celdas, de forma que se pueda calcular la cinemática inversa dentro de cada una de ellas. Esto aporta precisión al sistema y rapidez en los movimientos*

Como se puede observar, en la fase de ejecución, el modelo AVITE carga la matriz **Z** correspondiente a la celda en la que se ha detectado que se encuentra el objeto, y a continuación se calcula el incremento de rotaciones que hay que dar a cada articulación para alcanzar la solución final. *Sin embargo esta fase se puede hacer en varios pasos de forma realimentada con el fin de mejorar la precisión del sistema.* Esta es la opción por la que se ha optado en este proyecto. Esto permite, además poder alcanzar el objeto aunque éste empiece a moverse en un determinado momento, tal y como se verificará

posteriormente en los resultados obtenidos. La siguiente figura muestra el módulo de aprendizaje de todas las matrices **Z** para cada una de las celdas.

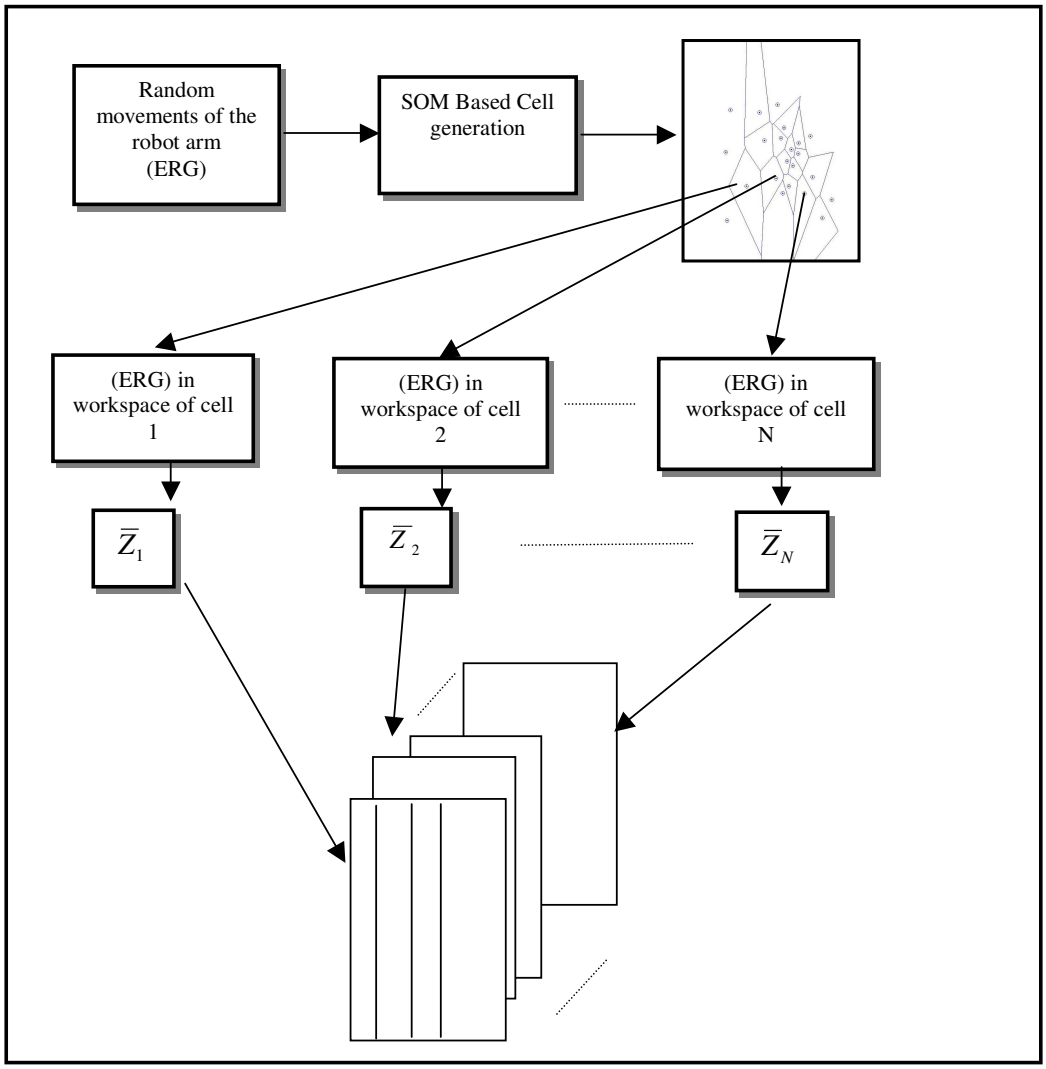

Figura 23. *Esquema general de la estrategia de alcance híbrida durante la fase de aprendizaje de las matrices* **Z**. *Como resultado final se obtiene una matriz multidimensional de pesos neuronales formada por todas las matrices Z (3xD) en cada celda de aprendizaje, siendo D el número de grados de libertad del robot y 3 la dimensión del espacio de trabajo. Así, al final se obtiene una matriz de dimensión (Nx3xD), siendo N el número de celdas generadas por la red SOM.*

Como ya se ha comentado, el segundo modelo de esta estrategia de alcance lo constituye el modelo AVITE, cuya misión es compensar el error de posición dentro de cada celda. Cada celda tiene un comportamiento diferente de las demás, lo que quiere decir que sólo se selecciona una en cada iteración mientras que las demás no se consideran (a no ser que se mueva el objeto, produciéndose entonces un cambio de celda brusco). Una vez que se ha detectado la celda, el modelo AVITE compensa la diferencia final en la operación de alcance. No obstante, en los experimentos realizados, se han considerado objetos en movimiento, por lo que los cambios de celda y su comportamiento serán analizados. La siguiente figura muestra la ejecución del AVITE en cada celda.

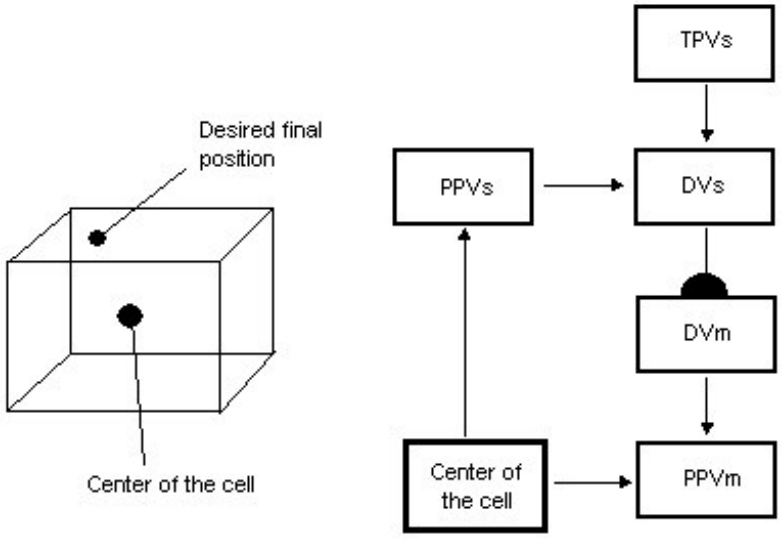

Figura 24. *Esquema general de ejecución del algoritmo AVITE en cada celda.*

La explicación del funcionamiento en este caso es similar a cómo se planteó la descripción de este modelo en apartados anteriores, siendo *TPVs* la posición deseada del robot en coordenadas espaciales o visuales (en realidad es la posición del objeto), *PPVs* la posición espacial o visual del centro de la celda (posición actual del efector final), *PPVm* la posición angular o los ángulos de la posición actual del brazo robot, *DVs* la diferencia entre *TPVs* y *PPVs* y, finalmente, *DVm* que es el incremento de posición de cada articulación y que tiene una relación lineal con *DVs.*

Cuando una celda es seleccionada, el centro de la celda proporciona su matriz de pesos **Z**. El vector *DVs* calcula la diferencia entre el centro de la celda y la posición deseada (la del objeto) en coordenadas espaciales. El producto de **Z** por *DVs* permite calcular el vector de diferencias motoras *DVm*.

Es importante destacar que esta segunda maniobra de reducir el error final de aproximación presentará una cota de error que puede reducirse sin más que volver a aplicar este algoritmo varias veces (lazo cerrado). Esto permitirá mejorar la precisión a cambio de unos cuantos pequeños movimientos de compensación del brazo robot. Para ello, se debe multiplicar la salida (al igual que se hacía con el modelo HRBF) por una señal de modulación o de control de velocidad *Go(t)*, cuya descripción ya ha sido realizada anteriormente. Por tanto, el esquema de la figura 24 queda modificado de la forma que se muestra en la siguiente figura.

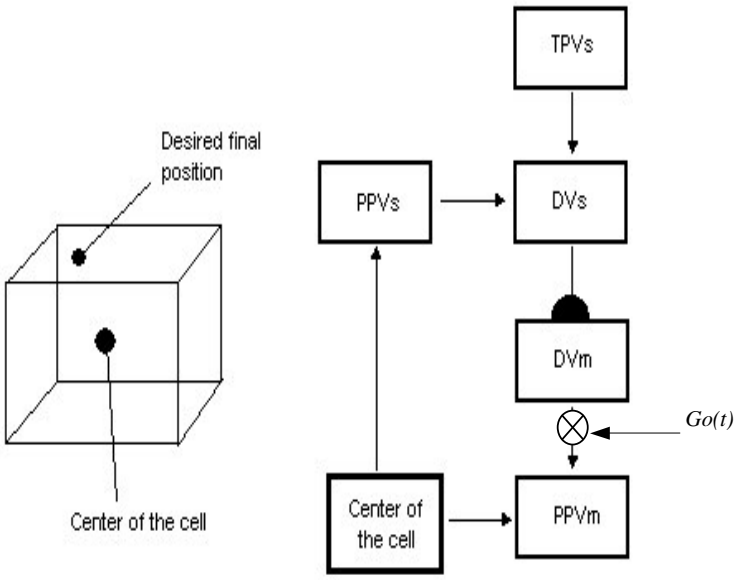

Figura 25. *Esquema general de ejecución del algoritmo AVITE en cada celda, considerando una serie de movimientos en lazo cerrado de compensación del error o de ajuste fino. La velocidad de estos pequeños movimientos es compensada por la señal Go(t)*

Aunque las ecuaciones del modelo AVITE ya han sido descritas de forma general, para este caso, nuevamente se tiene, para cada celda que la compensación del error medido por el vector *DVs* se debe realizar mediante un incremento de posiciones de las articulaciones del brazo robot *DVm* de la forma:

$$\vec{DV}_m = \mathbf{Z} \times \vec{DV}_S \tag{62}$$

habiéndose calculado cada uno de los elementos de la matriz **Z** para cada celda según la expresión adaptativa basada en el algoritmo de descenso del gradiente, según la ecuación:

$$z_{ijk}[n+1] = z_{ijk}[n] + \mu \cdot \left(DVm_i - \sum_j Z_{ijk}[n+1] \cdot DVs_j \right) \cdot DVs_j \tag{63}$$

donde *'i'* indica el número de articulación del brazo robot y *'j'* la coordenada espacial (x,y ó z) del incremento de posición entre la actual del efector final y la del objeto a alcanzar.

8 Descripción de la instalación

Para la implementación y testeo de las dos estrategias de alcance de objetivos con un brazo robot, y que han sido descritas en este proyecto, se han utilizado dos plataformas de muy diferente naturaleza.

Por un lado una plataforma de simulación de entornos robóticos, en los que el problema de la cinemática directa y de la cinemática inversa está internamente resuelto, por lo que la relación entre la posición del robot y la del efector final la proporciona el propio software. Este simulador ha sido desarrollado por el Grupo de Robótica del Departamento de Ingeniería de Sistemas y Automática de la Universidad Politécnica de Valencia y su nombre "comercial" es Virtual Robot Simulator (VRS) [5]. VRS presenta un entorno de simulación virtual de robots industriales comercializados en el mercado

(entre los que se encuentra el ABB-1400 y el CRS-A255 disponibles en el Laboratorio NEURCOR de la UPCT) y permite realizar programas en entorno de Visual C++ bajo Windows y ejecutar aplicaciones para actuar sobre los robots definidos en esta plataforma. La siguiente figura muestra un ejemplo de simulación de entorno de robots con VRS.

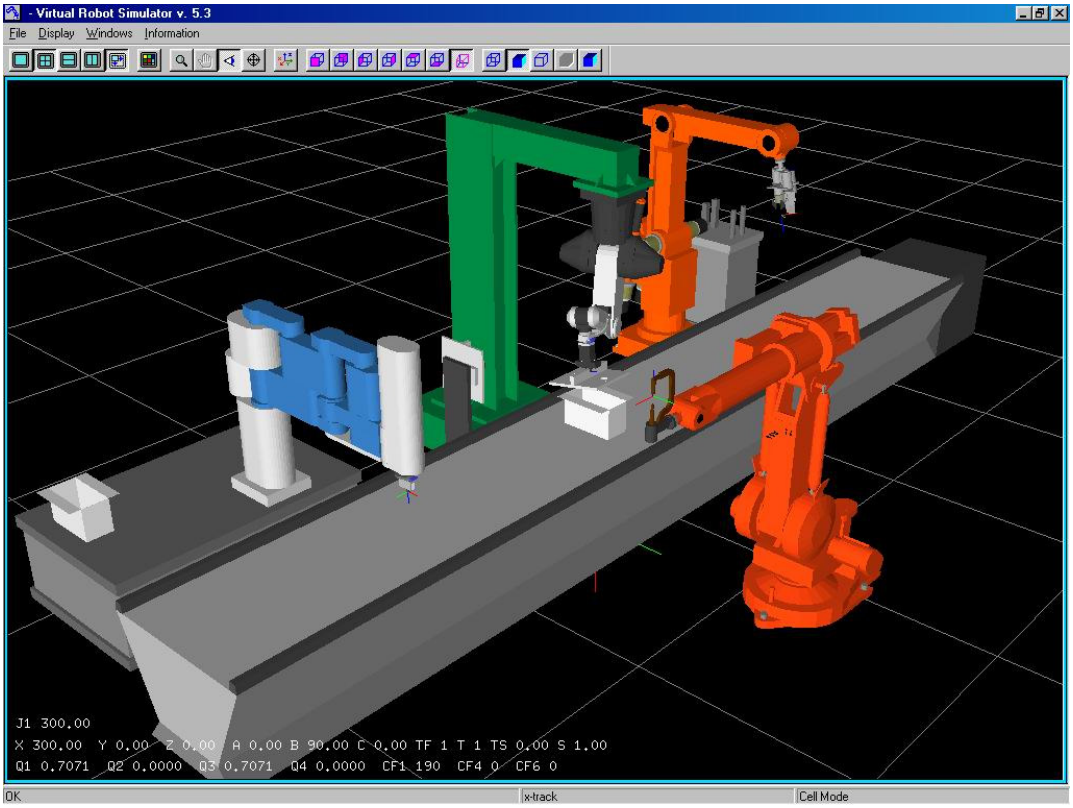

Figura 26. *Representación gráfica de un entorno virtual de 4 robots industriales en una cadena de montaje utilizando el simulador VRS*

Este software va a permitir realizar la mayor parte de las pruebas de las dos estrategias de alcance propuestas en este proyecto, debido a la facilidad de programación de este entorno y a la rapidez de respuesta. Sin embargo, con el objeto de verificar las capacidades de los modelos propuestos en una plataforma real, se ha utilizado la plataforma robótica visuo-motora desarrollada por el grupo NEUROCOR y que está formada por un robot industrial (operable desde un PC), una cabeza robot con un sistema de visión artificial y un manipulador robot (bien una pinza o bien una sofisticada mano antropomorfa) para operaciones de aprendizaje y alcance.

8.1 Plataforma de simulación Virtual Robot

Como ya se ha mencionado, el simulador VRS permite trabajar sobre robots industriales (cuyos modelos cinemáticas ya están incluidos en el propio software) sobre todo para probar modelos de operación que están basado en aprendizaje, como es el caso de los modelos de redes neuronales. Este software permite realizar todos los entrenamientos que se deseen sin problemas de que una equivocada programación pueda derivar en un accidente mecánico, así como permitir un ahorro de tiempo fundamental para el testeo de estos modelos. Así, mientras una plataforma real puede necesitar 4 horas de entrenamiento para tener un conjunto de patrones válido para el cálculo de los pesos de la red, esta plataforma puede tardar 15 minutos en las mismas condiciones. La siguiente figura muestra un ejemplo del entrono VRS donde el robot ABB-1400 ha sido cargado inicialmente.

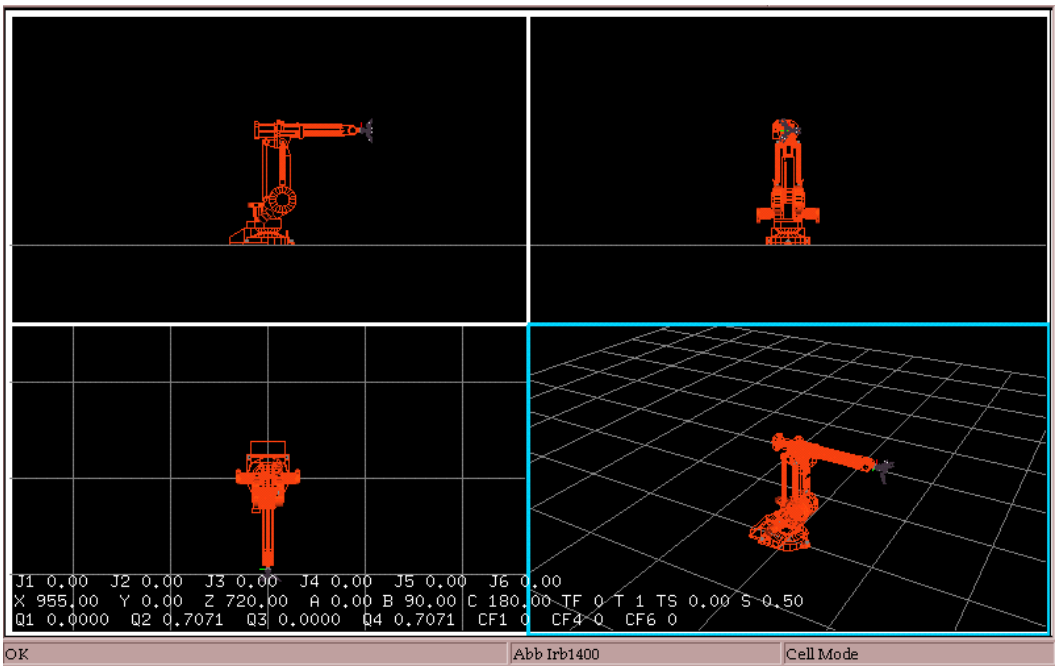

Figura 27. *Cuatro vistas del robot ABB-1400 representado mediante el simulador VRS.*

Como se puede observar, en la parte inferior de la pantalla, se suministran los valores (en grados) para las 6 articulaciones (J1..J6) o grados de libertad de este robot (Base, Hombro, Codo y 3 de la muñeca). Al mismo tiempo, las coordenadas de posición X, Y, Z del efector final son proporcionadas (en milímetros), tomando como sistema de

referencias la base del robot. Otros parámetros como la orientación de la pinza son también visualizados, aunque no se utilizarán para este trabajo.

Con el objeto de validar los modelos propuestos, en otras plataformas virtuales, el robot CRS-A255 – ver figura 28- ha sido también utilizado para las simulaciones. Esto va a permitir comprobar la independiera de estos modelos con respecto a las plataformas robóticas utilizadas.

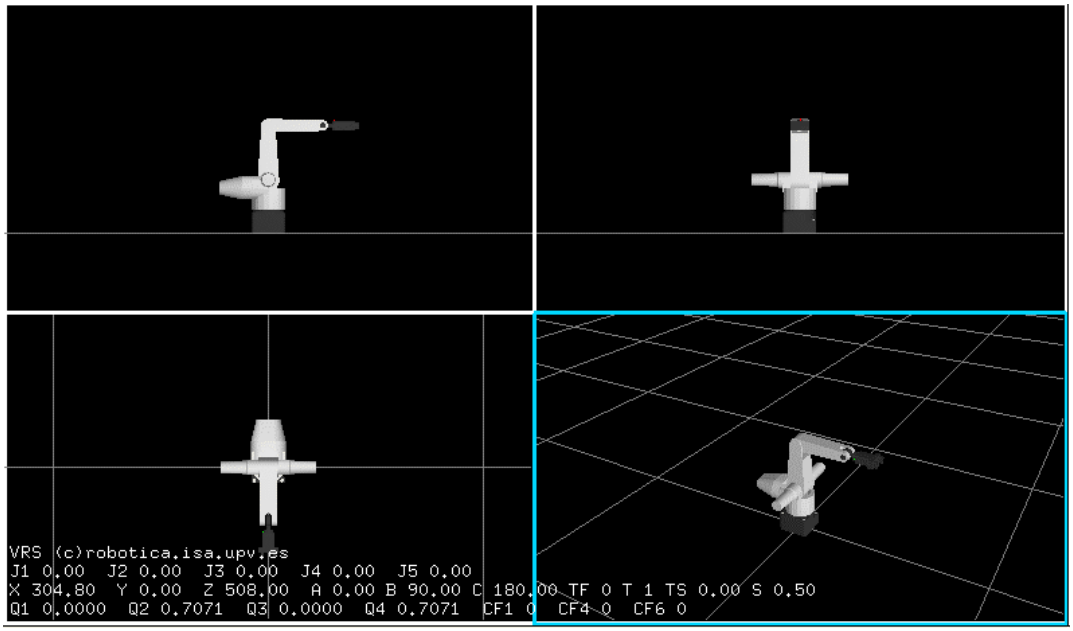

Figura 28. *Simulación del robot CRS-A255 de 5 grados de libertad (J1..J5) de diferentes dimensiones que el ABB, utilizado para mostrar las capacidad de flexibilidad de aprendizaje de las redes neuronales.*

Sobre este entorno CAD, es posible desarrollar una aplicación en Visual C++ de forma muy sencilla, puesto que las librerías para mandar posiciones a los robots, las de lectura de las posiciones de articulación del brazo robot y las de lectura de posición XYZ del efector final, así como las de interactuación con el entorno CAD, son proporcionadas por el software VRS. La siguiente figura muestra un ejemplo de un programa de aplicación que se ha realizado inicialmente en este proyecto para familiarizarse con el entorno.

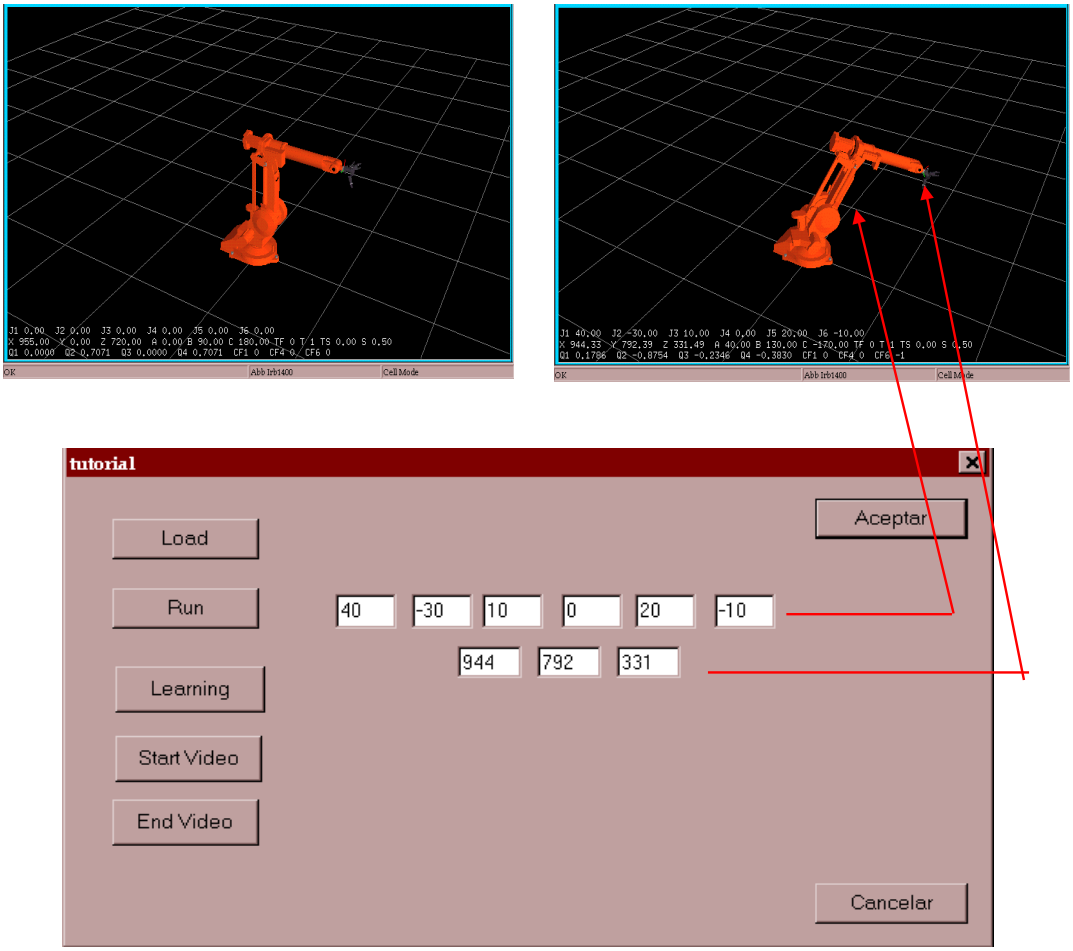

Figura 29. *Ejemplo de una sencilla aplicación para testera el funcionamiento de las librerías del software VRS.*

Estas interesantes funciones que presenta este software es la de poder colocar objetos (esferas, cilindros, prismas…) en la escena, cuyas coordenadas son conocidas por medio del simulador. Esto permite que las operaciones de alcance de objetivos sean visuales, lo que facilita en análisis de funcionamiento de los modelos propuestos, sobre todo cuando se trata de objetos en movimiento o que cambian rápidamente de posición.

Puesto que las dos estrategias de alcance de objetivos son completamente diferentes, ha sido necesario realizar dos aplicaciones para el entrenamiento y testeo de ambos modelos. Aunque la secuencia de entrenamiento es la misma para ambas estrategias (mover aleatoriamente el brazo robot un número determinado de veces y guardar las posturas por un lado y la posición del efector final, por otro), la ejecución es

completamente diferente. Las siguientes figuras muestran el interfaz para ambas estrategias.

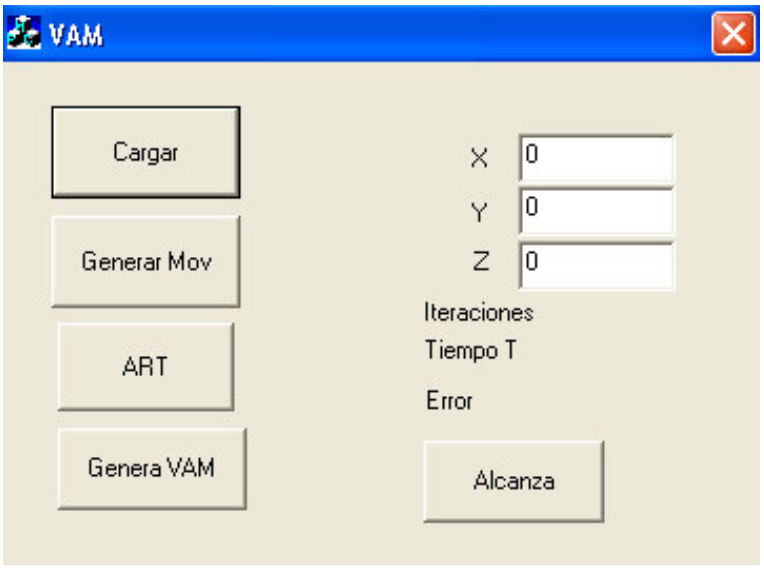

Figura 30. *Interfaz para entrenamiento y ejecución del modelo SOM-AVITE*

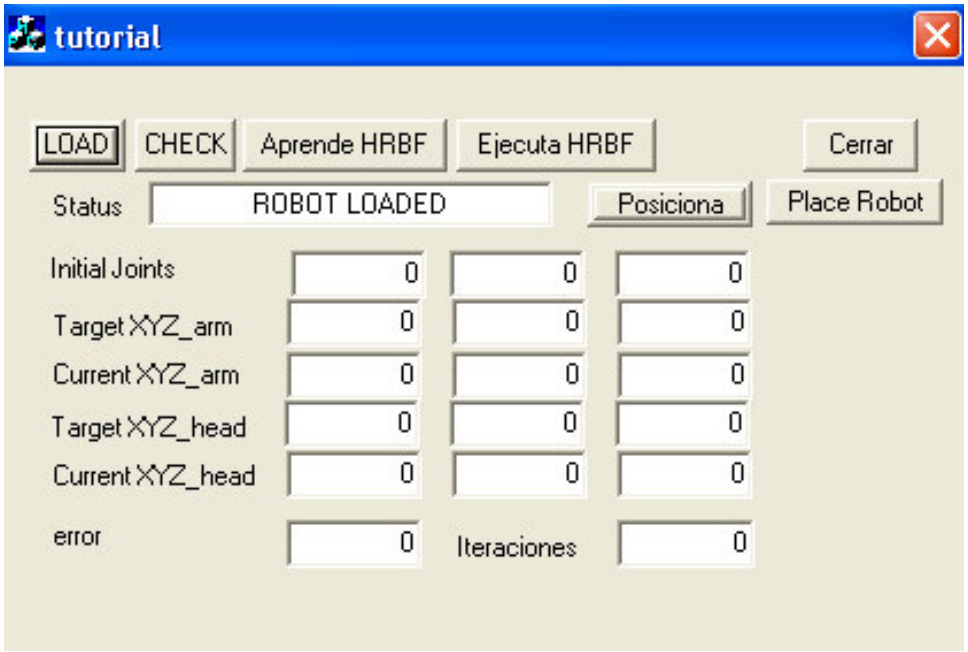

Figura 31. *Interfaz para entrenamiento y ejecución del modelo HRBF*

8.2 Plataforma robótica

Para la última parte de la fase de experimentación de este proyecto, se han aplicado los modelos, previamente testeados en el software de simulación VRS, a una plataforma real, formada por un sistema robot cabeza-brazo-mano, tal y como muestra la siguiente figura

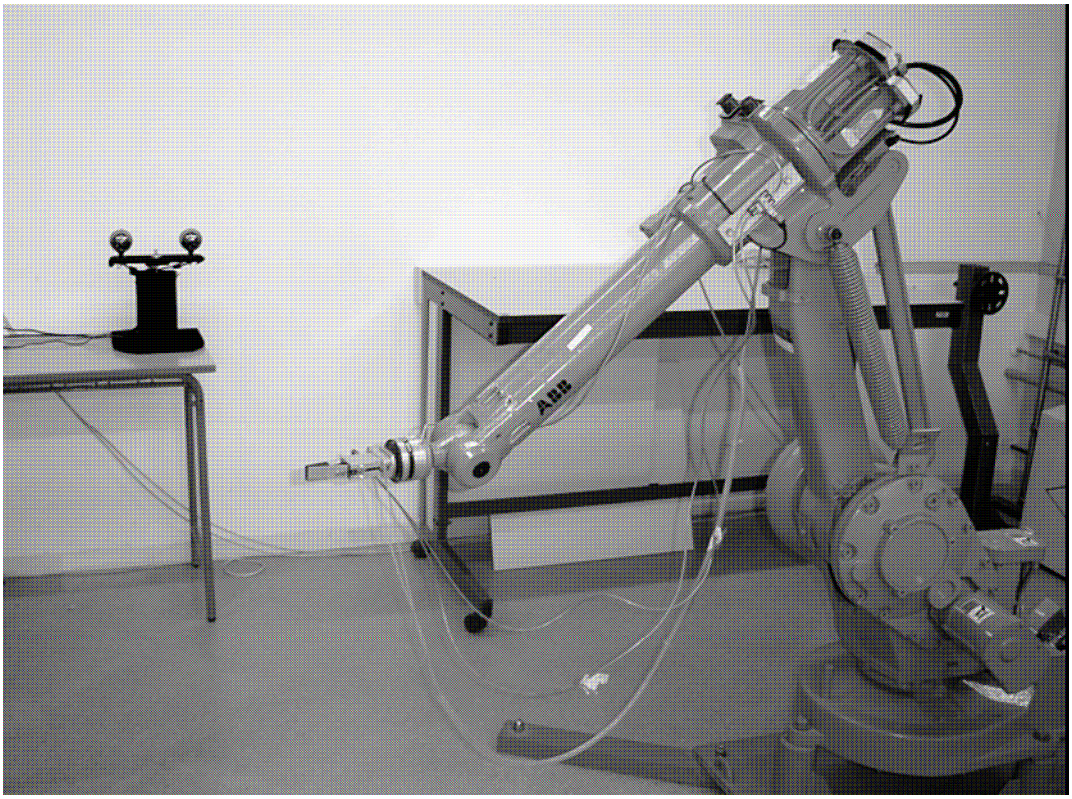

Figura 32. *Plataforma robótica NEUROCOR*

Como se puede observar, esta plataforma consta de un cabezal robot de diseño propio y 5 grados de libertad, con dos cámaras a color para la identificación del efector final (que para la fase aprendizaje ha consistido en una pequeña pinza coloreada de un determinado color para la correcta segmentación del sistema de visión), un brazo robot comercial ABB-1400 de 6 grados de libertad (tronco, hombro, codo y 3 para el movimiento de la muñeca) y una mano robot y/o una pinza robot. La siguiente figura muestra una foto de la plataforma con la mano robot de 3 dedos incorporada en una operación de alcance y agarre de un cilindro de color rojo (diferente del utilizado para el efector final que fue amarillo).

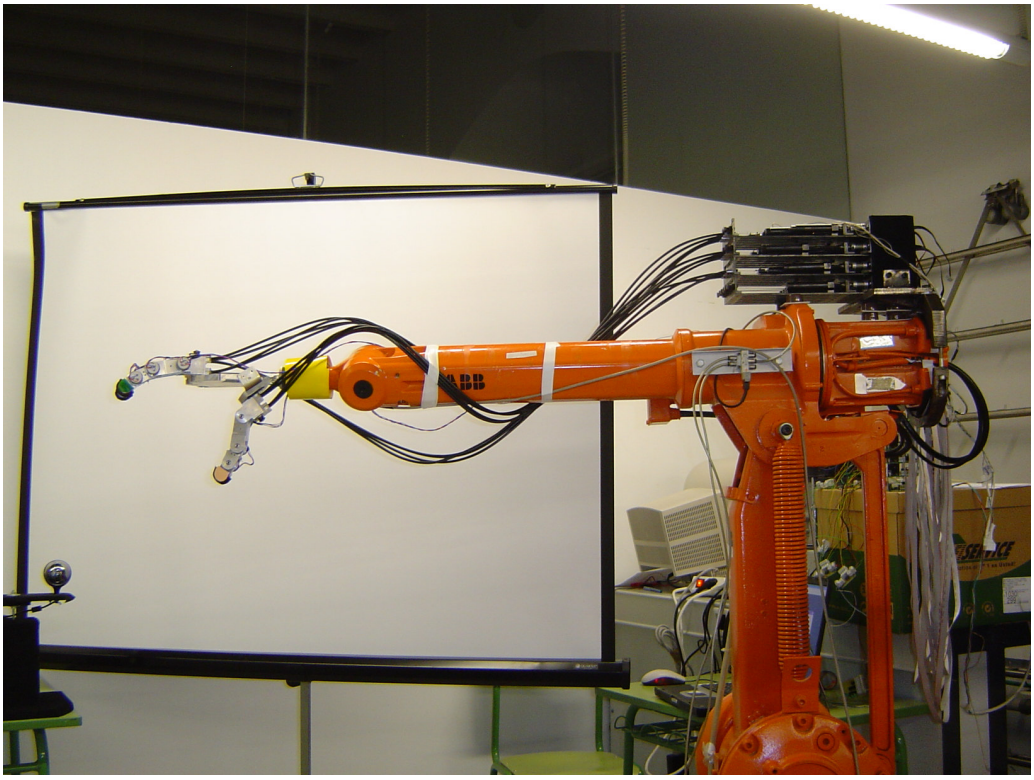

Figura 33. *Plataforma robótica NEUROCOR en una operación de alcance-agarre*

Las características principales del brazo robot son:

- 6 grados de libertad
- Capacidad de Carga: 5Kg
- Repetibilidad: +/- 0.05mm
- Alcance máximo: 1,4 metros
- Rango de movimiento:

  Eje 1: 340º Eje 2: 140º Eje 3: 135º
  Eje 4: 300º Eje 5: 230º Eje 6: 600º

- Velocidad nominal del efector final: 1016 mm./sg.
- Peso del robot: 225Kg
- Dimensiones de la base: 620x450mm.
- Peso del controlador: 300Kg
- Sistema de control cerrado. Esto implica que no se puede interrumpir la trayectoria del robot durante un movimiento, ni leer la posición en la que se

encuentra. Esto supone una gran dificultad para los algoritmos que trabajan en lazo cerrado. Por este motivo, la arquitectura propuesta SOM-AVITE aporta una enorme ventaja de control para este tipo de robots.

- Grandes retardos de ejecución desde un PC: entorno a 2 segundos para cada comando u orden de posicionamiento.
- Secuencia de ejecución de movimiento desde un PC:
    - Envío de comando con una función en C
    - Traducción de esa función al lenguaje del robot (RAPID)
    - Almacenamiento en disco de ese comando RAPID
    - Carga de ese comando en el controlador del robot
    - Ejecución del movimiento

Finalmente, el algoritmo de visión (previamente desarrollado por el grupo de Visión) permite detectar simultáneamente la posición del objeto a agarrar y del efector final. Como esta posición en detectada en ambas cámaras, por triangulación es posible determinar la posición 3D de ambos objetivos a partir de la información detectada en cada una de ellas. Las siguientes figuras muestran una representación de la segmentación del algoritmo de visión para la determinación de ambos objetivos, que son diferenciados por tener un color diferente y conocido por el sistema.

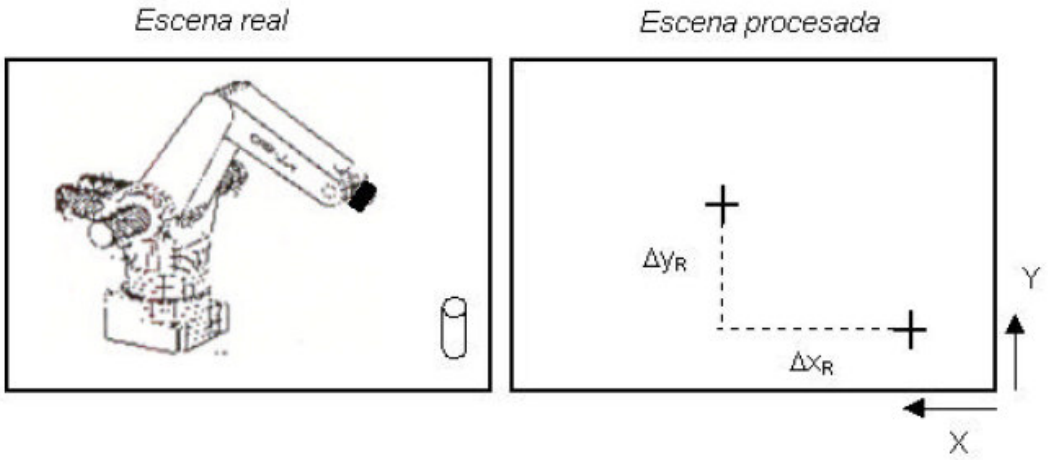

Figura 34. *Esquema de reconocimiento de imágenes de una cámara de visión*

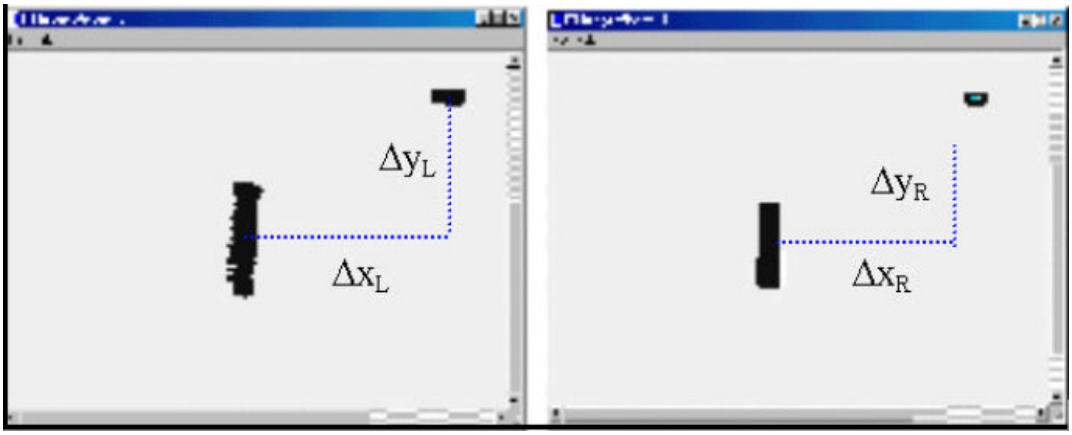

Figura 35. *Obtención de los parámetros de distancia entre efector final y objetivo, en coordenadas de las cámaras de visión. Posteriormente un algoritmo de triangulación permite obtener la posición de ambos objetivos en coordenadas XYZ*

9 Resultados experimentales

Con el objetivo de testear las dos estrategias propuestas para la generación de un movimiento de alcance bajo diferentes características tanto del entorno como del robot, se ha desarrollado una serie de experimentos tanto en plataformas de simulación (utilizando Matlab y posteriormente Virtual Robot Simulator) como en la plataforma robótica real del laboratorio Neurocor, considerando las siguiente situaciones:

a) Diseño, programación, entrenamiento y análisis de comportamiento del modelo de Kohonen para la generación de celdas de aprendizaje para el modelo híbrido SOM-AVITE para un sencillo robot de 5 articulaciones simulado con Matlab.

b) Diseño, programación, entrenamiento y análisis de comportamiento del modelo ART-AVITE para un modelo de robot ABB-1400 de 3 grados de libertad, haciendo uso del software VRS y una aplicación desarrollada en Visual C (Figura 30).

c) Diseño, programación y análisis de comportamiento del modelo HRBF para un modelo de robot ABB-1400 de 3 grados de libertad, haciendo uso del software VRS (Figura 31).

d) Análisis comparativo de las dos estrategias de alcance propuestas en este proyecto utilizando la plataforma VRS con un robot ABB-1400, utilizando 3 grados de libertad.

e) Análisis comparativo de la estrategia en lazo cerrado HRBF implementado con el simulador VRS en dos robots diferentes (ABB-1400 y CRS-A255), con las mismas aplicaciones en Visual C, mostrada sen las figuras 30 y 31, y considerando 4 grados de libertada para cada uno (redundancia).

f) Programación en la plataforma real formada por el robot ABB-1400 y el cabezal estereoscopio de diseño propio del modelo HRBF, incluyendo las etapas de aprendizaje y de testeo.

9.1. Plataformas de Simulación

A continuación se muestran los resultados de la aplicación de los modelos de alcance propuestos, tanto en la fase de aprendizaje como en la de ejecución de la tarea, en los entornos de simulación de Matlab y VRS, tal y como se ha descrito en el apartado anterior

9.1.1. Modelo Híbrido Kohonen-AVITE

Con el objetivo de verificar las capacidades del modelo híbrido propuesto, basado en redes de tipo autoorganizativo (SOM) con una red de tipo AVITE, se ha realizado un sencillo simulador en Matlab de un brazo robot con 5 articulaciones o grados de libertad para, posteriormente simular su precisión de alcance en dos fases (alcance grueso al centro de la celda y alcance fino dentro de la propia celda) con el modelo, en este caso, de Kohonen para la discretización del espacio de trabajo 3D.

El primer conjunto de experimentos ha sido enfocado a la generación de las celdas de aprendizaje ("*learning cells*") tridimensionales, considerando diferentes parámetros de aprendizaje de la red. Así, la Figura 36 muestra la evolución de la posición XYZ de una de las celdas del modelo de Kohonen después de 40 iteraciones de este algoritmo. La

evolución de las coordenadas de este centroide equivale a la evolución o aprendizaje de los pesos asociados a esa celda.

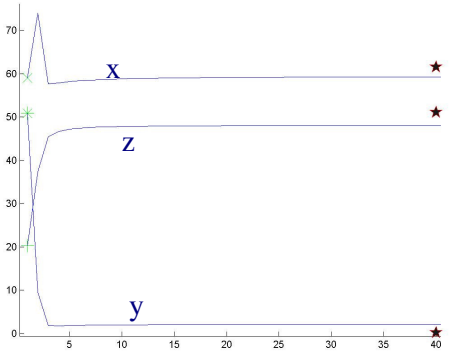

Figura 36. *Evolución de la posición de un centroide en el algoritmo de Kohonen*

Las curvas de la Figura 36 muestran el número de iteraciones del algoritmo no supervisado de Kohonen, necesarios para alcanzar el valor final de convergencia de la posición (o pesos) de esa celda de aprendizaje. Para el entrenamiento, se han generado 500 posiciones aleatorias del brazo robot, obteniéndose 500 puntos del efector final. Una vez obtenidos, se han definido 20 celdas de aprendizaje cuyos centroides se han dejado evolucionar durante 40 iteraciones del algoritmo. Cada curva representa las coordenadas X, Y, Z de la posición del centroide. El símbolo '*' indica la posición final corregida de la posición del centroide, que es la más próxima de las 500 posiciones a esa posición. En ella la relación entre posición del brazo y posición del efector final es conocida. La figura 37 muestra la representación gráfica de las fronteras o regiones de Voronoi generadas, proyectadas sobre los planos XZ y XY.

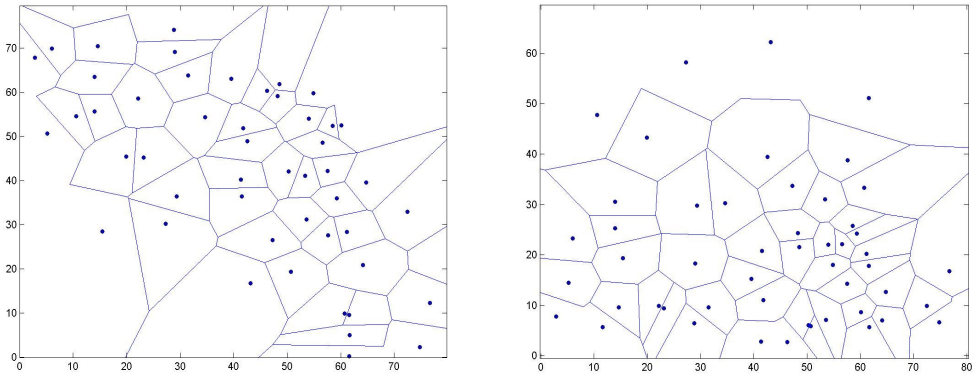

Figura 37. *Resultado de la implementación de los mapas de Kohonen para la generación de regiones de Voronoi, representadas mediante proyecciones sobre los planos XY (derecha) y XZ (izquierda).*

Como se puede observar de esta figura, el número de celdas que se genera es mayor dónde el brazo robot tiene más posibilidades de quedar configurado. De esta forma es posible obtener una mayor precisión en las zonas de alcance más frecuentes. Una representación de las regiones 3D de Voronoi es mostrada en la siguiente figura para 3 celdas de un modelo de Kohonen. Como se puede observar, cada región está definida por un número diferente de esquinas y aristas. La intersección de las superficies limita el espacio de trabajo o el ámbito de cada celda. Posteriormente, la siguiente fase de aprendizaje del modelo Kohonen-AVITE necesitará implementar una red AVITE dentro de cada celda.

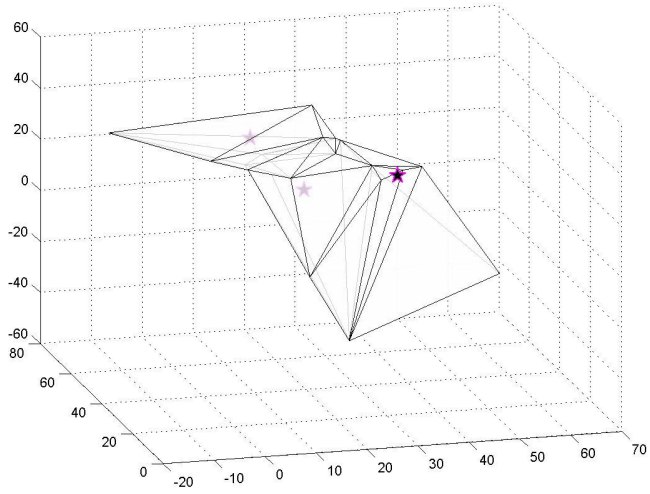

Figura 38. *Ejemplo de representación de la generación de celdas de Voronoi 3D para 3 neuronas, donde el símbolo '*' indica la posición final del centroide de cada celda.*

El modelo neuronal de Kohonen que ha sido implementado para la generación de celdas de aprendizaje, comienza desde un parámetro *N* que indica el número final de celdas en los cuales el espacio de trabajo va a quedar dividido. Este valor es relativamente crítico para la precisión del algoritmo en operaciones de alcance.

Con el objetivo de testear el desarrollo del algoritmo supervisado AVITE para compensar la diferencia especial existente entra la primera aproximación del brazo robot hacia el centroide de la celda donde se encuentra el objeto y la posición del mismo, esta red calcula la el incremento de posición de las articulaciones del brazo robot. Para ello, se ha realizado una fase previa de aprendizaje para cada celda,

obteniendo *N* matrices de pesos **Z**, tal y como se describió anteriormente. El resultado será una matriz de dimensión 3x5 (en este caso) de pesos neuronales para cada celda.

La siguiente figura muestra la posición final del centro de las celdas y del efector final después de la fase de aprendizaje. En la gráfica de la izquierda, se muestran los resultados de la fase de entrenamiento del modelo AVITE. Para cada celda, se han generado pequeños movimientos aleatorios del brazo robot dentro de la región de Voronoi. Con esos datos, y aplicando el método de mínimos cuadrados se ha obtenido la matriz **Z** para cada celda. En la gráfica de la derecha, se muestra la distribución de la posición final

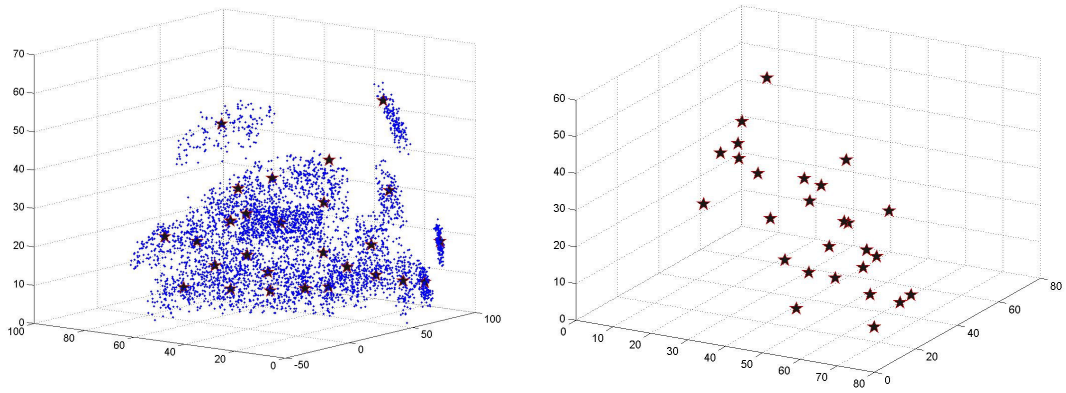

Figura 39. *Representación de la fase de aprendizaje del modelo AVITE en cada celda del mapa de Kohonen*

Finalmente, la siguiente figura muestra una simulación gráfica a este problema de alcance en 2 movimientos como resultado de la aplicación del algoritmo propuesto Kohonen-AVITE. La posición final del brazo robot es obtenida en dos pasos de forma secuencial: 1) aproximación al centro de la celda donde se encuentra el objeto y 2) compensación de la distancia por medio del modelo AVITE. Aunque son dos pasos de ejecución, los cálculos se realizan previamente y se manda la posición de una sola vez, por lo que el resultado es bastante aparente, tratándose de problemas no lineales en 3D. En esta secuencia, la primera gráfica representa la posición inicial del brazo robot y del objetivo, la segunda representa la posición 3D de la celda ganadora y la configuración del brazo robot posicionado de forma que el efector final quede en el centro de esa celda y, finalmente, la tercera muestra la posición final del brazo robot. Los parámetros para

esta secuencia de alcance simulada han sido Objetivo: {26, 71, 22}, Posición deseada: {40, 40, 30} y Número de Celdas: N = 40. Para este caso, el error final relativo obtenido ha sido de 1,27%

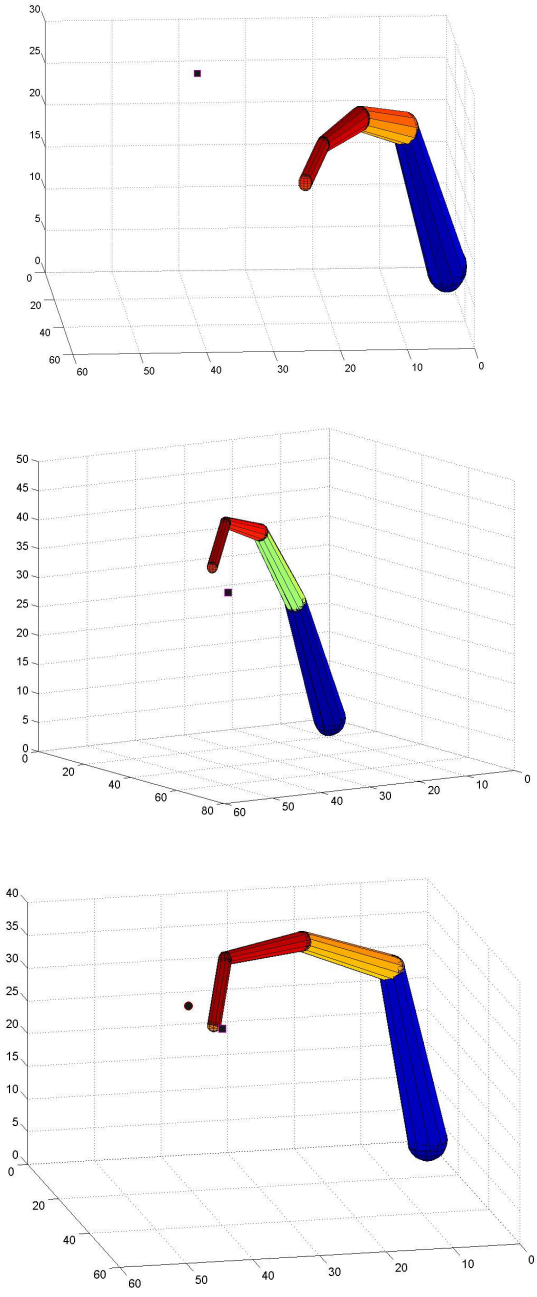

Figura 40. *Representación de una operación de alcance con una simulación de un robot con 5 grados de libertad con el modelo Kohonen-AVITE*

9.1.2. Modelo Híbrido ART-AVITE

La implementación de este modelo ha sido realizada en el simulador VRS descrito anteriormente. Este simulador tiene la ventaja de poder simular el robot ABB-1400 y de conocer la cinemática directa del robot sin necesidad de ningún algoritmo. Aunque el modelo ABB-1400 tiene 6 grados de libertad, para este caso se ha limitado a los 3 primeros, es decir, tronco, hombro y codo. El hecho de utilizar una red ART en lugar de una red de Kohonen, está determinado por la conveniencia de que el número de celdas de aprendizaje se genere automáticamente en función del entrenamiento inicial del robot y del parámetro de vigilancia ρ. En primer lugar se ha realizado una generación de celdas con el modelo ART, dentro del espacio de trabajo del brazo robot. Para 600 entrenamientos de la red ART y escogiendo ρ = 0,18 se ha obtenido la siguiente disposición de centroides en el simulador VRS.

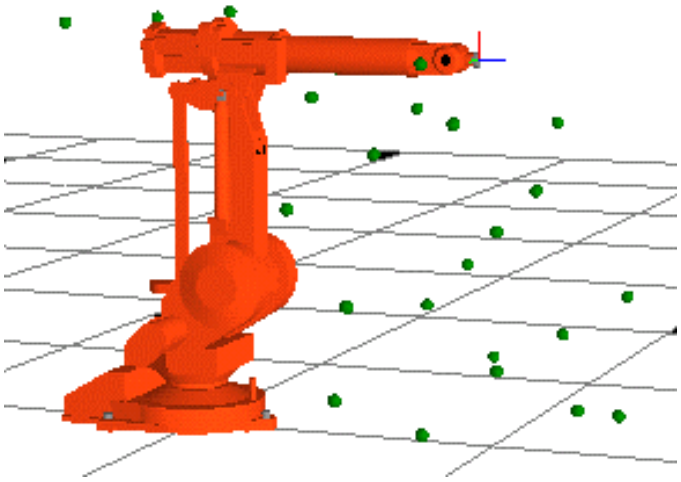

Figura 41. *Simulación de la distribución de centroides en una red ART*

Con el objeto de testear y verificar el modelo ART-AVITE para aplicaciones de alcance cuando el objeto a alcanzar no se mueve, se ha realizado una serie de experimentos cuyos resultados se muestran en la Tabla 1. En esta tabla, el comportamiento del modelo ha sido comparado mediante la ejecución del algoritmo con diferentes valores del número de celdas de aprendizaje generadas y el error final en milímetros alcanzado después de aplicar **en lazo cerrado** el modelo AVITE.

ρ	error (mm)	objetivo (mm)	celdas	tiempo (sg)	ρ	error (mm)	objetivo (mm.)	Posición del efector final (mm.)	Tiempo (sg.)
0,12	8,6	T_1	53	3,4	0,13	<1	T_1	{100,5; 900,1;1000,6}	6,0
0,16	7,5	T_1	27	3,4	0,13	<5	T_1	{102,3; 900,5;1002,9}	5,2
0,18	9,4	T_1	21	3,5	0,13	<10	T_1	{104,9; 901,1;1006,2}	4,8
0,12	6,7	T_2	53	3,3	0,13	<1	T_2	{299,6;- 900,0;800,3}	3,2
0,16	7,0	T_2	27	5,5	0,13	<5	T_2	{301,5;- 900,7;799,2}	2,9
0,18	5,9	T_2	21	4,5	0,13	<10	T_2	{294,0;- 900,2;804,1}	2,9

Tabla 1. *Diferentes escenarios de aplicación del modelo ART-AVITE.*

Tal y como muestra la Tabla 1, se han considerado diferentes valores para el parámetro ρ, posiciones del objetivo y cotas máximas de error deseado. En todos los casos se ha tomado Go =1, una posición inicial del efector final {900,100,400} y dos posiciones del objetivo: T_1={100,900,1000} y T_2={300,-900,800}. La influencia del parámetro ρ en la precisión final, así como el tiempo necesario para alcanzar el objetivo, puede observarse en la Tabla.

Con el objeto de validar esta arquitectura cuando se produce una **inesperada o repentina variación de la posición del objetivo**, se ha realizado otra serie de experimentos con desplazamientos instantáneos del objeto, desde la posición P_1={100,500,900} hasta la P_2={700,-900,400}. Los resultados obtenidos se muestran en las siguientes figuras.

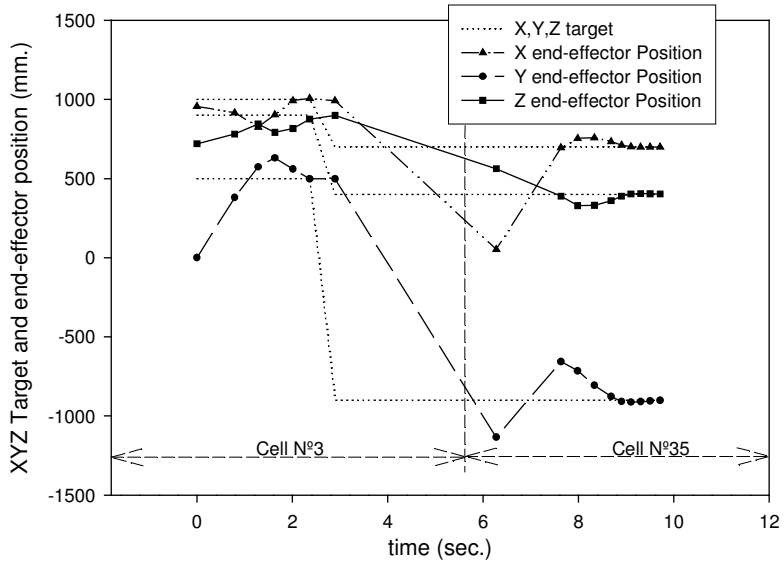

Figura 42. *Algoritmo ART-AVITE para alcance de objetos con perturbación*

En este caso, la posición del objeto varía desde la celda Nº 34 hasta la Nº 35. Eso implica que cuando se produce el cambio de celda, el modelo AVITE que se aplica para la reducción de la distancia entre efector final y objetivo, se ejecuta con la matriz **Z** correspondiente a la nueva celda.

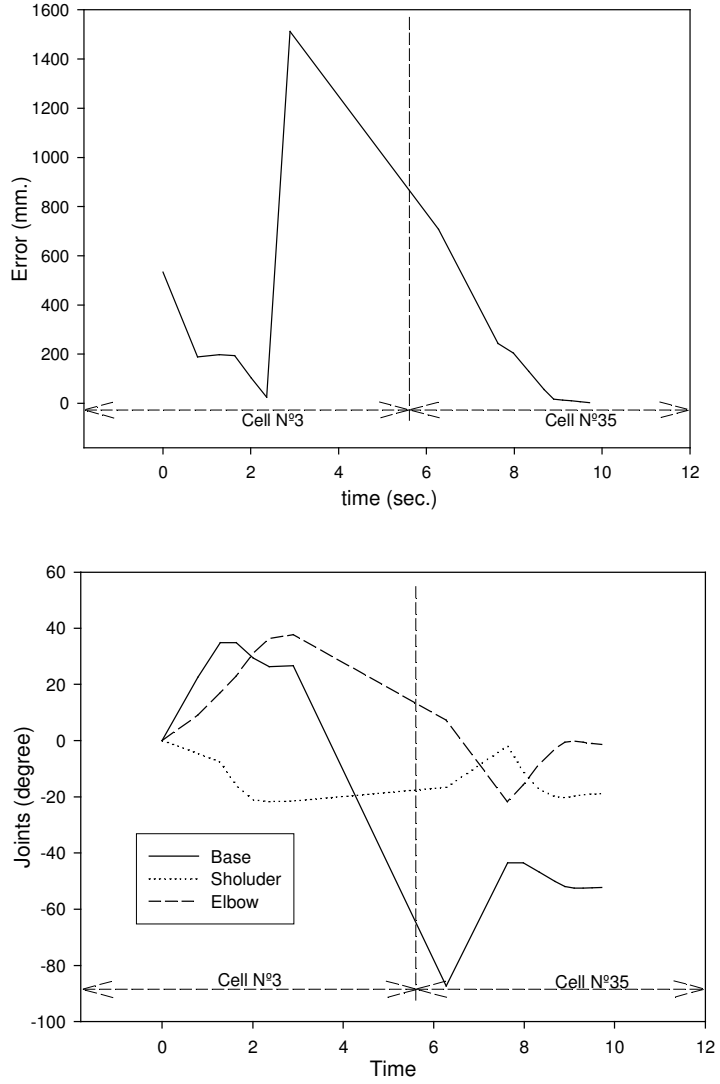

Figura 43. *Evolución del error (superior) y de la posición del efector final del brazo robot (inferior) cuando se produce un movimiento rápido del objetivo*

Finalmente, para testear las capacidades de la arquitectura propuesta para **seguimiento de objetivos**, se han generado movimientos constantes del objeto a alcanzar y se ha ejecutado simultáneamente este modelo híbrido de alcance con la configuración anterior.

Como parece lógico, en este caso se van a producir varios cambios de celda, en función de la velocidad o dirección del objetivo a alcanzar. Dentro de cada una de las celdas, el modelo AVITE se ejecutará con la matriz **Z** aprendida para cada una de ellas y tratará de compensar la diferencia entre el efector final y el objetivo lo más rápidamente posible antes de que se produzca un nuevo cambio de celda. Los resultados se muestran a continuación:

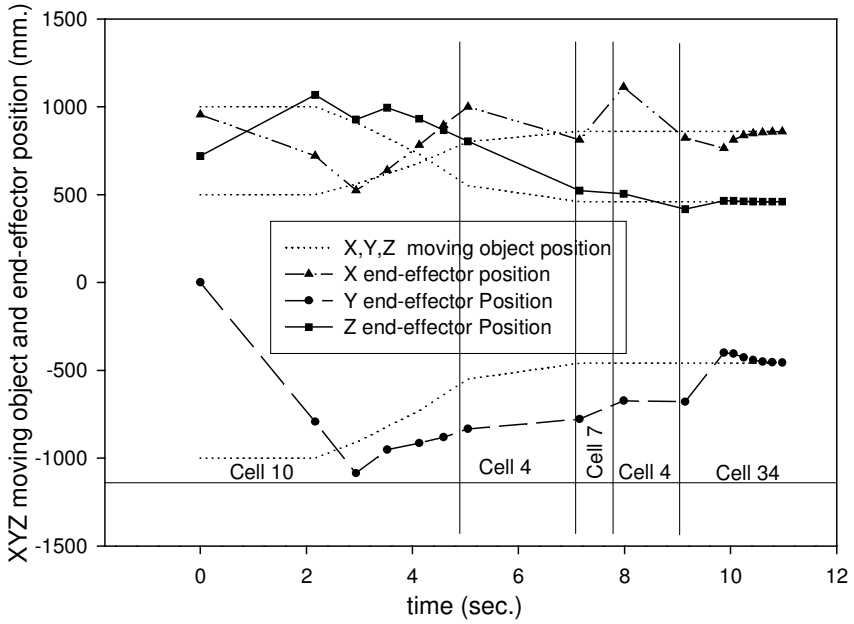

Figura 44. *Evolución 3D del efector final del brazo robot para seguimiento de un objetivo en movimiento con una velocidad de 7,6 cm/sg, para el cual se producen 5 cambios de celda antes de ser alcanzado*

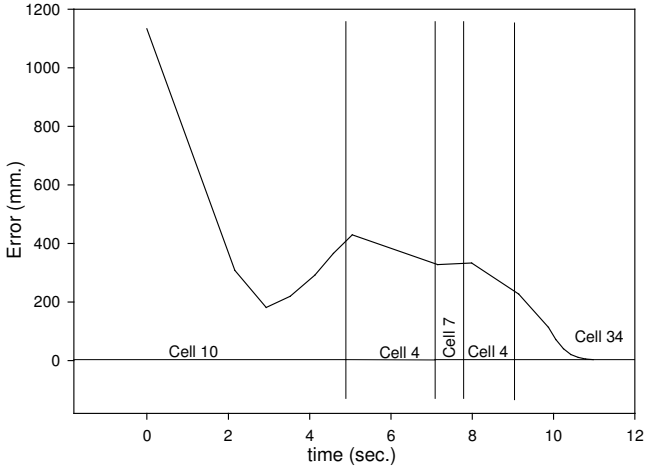

Figura 45. *Evolución del error*

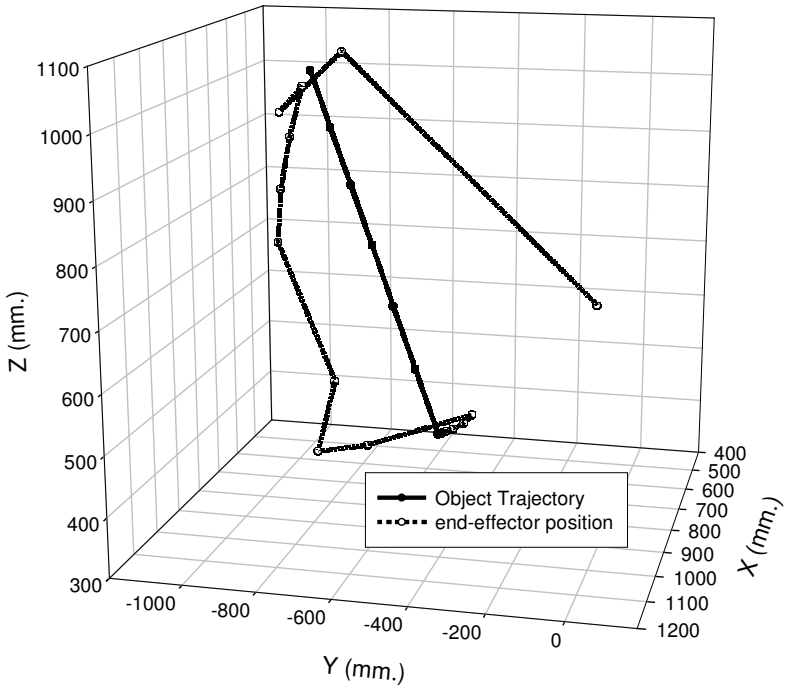

Figura 46. *Evolución del efector final, junto con el del objetivo*

9.1.3. Modelo en lazo cerrado HRBF

La implementación de este modelo, ya descrito anteriormente, ha sido realizada en la plataforma de simulación VRS. Como se comprobará, los resultados obtenidos van a permitir la verificación y testeo de las capacidades de esta arquitectura neuronal de alcance en lazo cerrado, mediante la adaptación de la configuración de la red al caso de un robot ABB-1400 considerando los tres primeros grados de libertad. Los experimentos desarrollados han estado enfocados al análisis de la precisión, en función del umbral de error deseado o la cota máxima de error permitida, robustez y capacidad de seguimiento del modelo para:

- Alcance de objetivos
- Alcance de objetivos cuando se consideran perturbaciones rápidas en su posición
- Seguimiento de objetos en movimiento.

La expresión considerada para la señal que modula la velocidad del movimiento del brazo robot durante su trayectoria ha sido, en este caso:

$$GO(t) = \frac{KGo \cdot t^3}{300 + t^3} \tag{63}$$

La configuración de los experimentos para los experimentos desarrollados se muestra en la Tabla 2.

	Error Umbral	Posición inicial articulaciones	Posición inicial efector final	Posición del Objetivo	Perturbaciónn
P1	1	10,-50,15	2623,-66,742	2150,1250,150	NO
P2	5	10,-50,15	2623,-66,742	2150,1250,150	NO
P3	10	10,-50,15	2623,-66,742	2150,1250,150	NO
P4	1	10,5,-5	1855,-202,503	3 posiciones	NO
P5	5	10,-40,10	2736,-46,944	2150,1250,400	1600,-450,-200
P6	5	10,-20,15	2129,-153,542	Movimiento lento	SI
P7	1	10,-20,15	2129,-153,542	Trayectoria circular YZ	SI

Tabla 2. Distintos escenarios experimentales

En todos los experimentos P1 a P7, la posición relativa entre la cabeza y el brazo, definida por el vector *R,* tal y como muestra la Figura 47 (tomada de la plataforma real), ha sido: {-3000, 0, 800}, la ganancia *KGo* de la señal Go(t) ha sido 20, el número de entrenamientos para la red neuronal HRBF ha sido de 3000, la dimensión de la matriz de pesos de [3 x 3 x 250], mientras que para la matriz de centros y de varianzas de las funciones Gaussianas, la dimensión ha sido de [3 x 250].

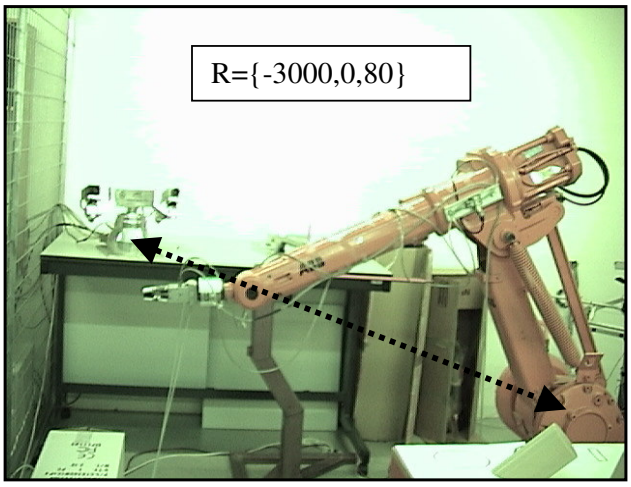

Figura 47. *Posición relativa entre la cabeza y el brazo*

La posición del efector final del brazo robot es obtenida directamente por el software de simulación, así como el de la posición del objeto (esfera). A continuación se detallan los resultados de todos los experimentos.

ALCANCE DE OBJETOS

Para este caso, se han realizado los experimentos señalados en la tabla como P1, P2 y P3, en los que el objeto está siempre fijo. Los resultados obtenidos se muestran en las siguientes gráficas:

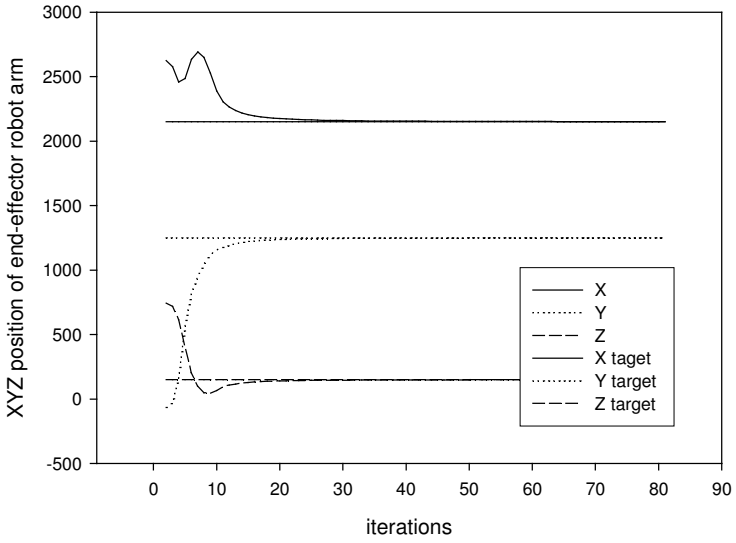

Figura 48. *Exp. P1. Evolución del efector final*

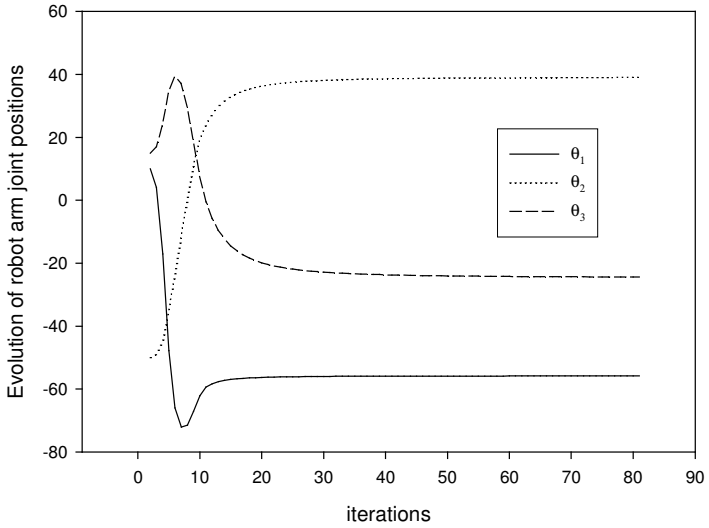

Figura 49. *Exp. P1. Evolución de las posiciones de articulación del robot*

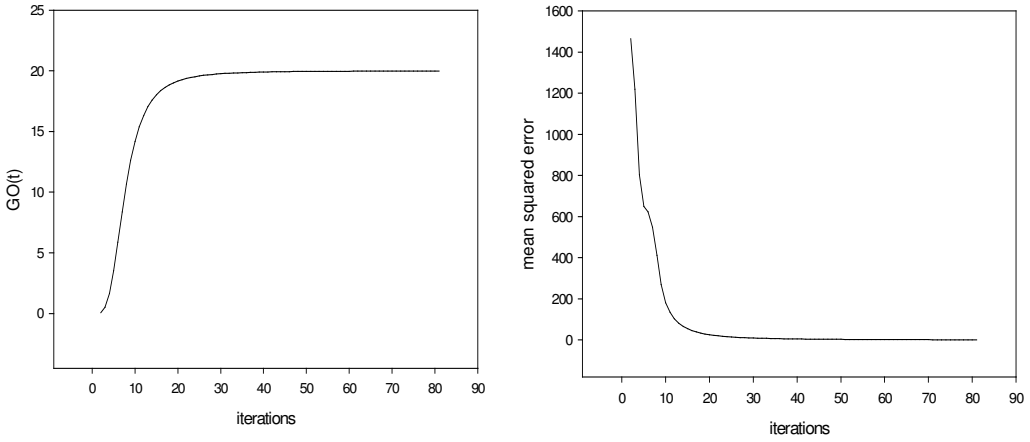

Figura 50. *Exp. P1. Evolución de la señal Go(t) y del error entre objetivo y posición actual del efector final*

Con el objeto de analizar la relación entre el número de movimientos del robot hasta alcanzar el objetivo y la precisión obtenida, se han realizado los experimentos P2 y P3 en las mismas condiciones anteriores. Los resultados obtenidos se muestran en la siguiente tabla.

Nº Experimento	Error Final	Iteraciones	Posición final de las articulaciones
P1	0,9916	81	[-55.80, 39.02, -24.37]
P2	4,8596	40	[-55.86, 38.61, -23.70]
P3	9,5301	30	[-55.95, 38.10, -22.85]

Tabla 3. *Resultados comparativos entre velocidad y precisión del modeloHRBF*

La representación de la trayectoria curvada es una característica particular de la red HRBF. Para ver este efecto, en el experimento P4 se han realizado tres operaciones de alcance, cuyos resultados se han representado en formato 3D en la siguiente figura 51. En ella, se pueden observar las desviaciones de la trayectoria con respecto a la trayectoria óptima que debería seguir el brazo robot que es la línea recta entre la posición inicial y la del objetivo.

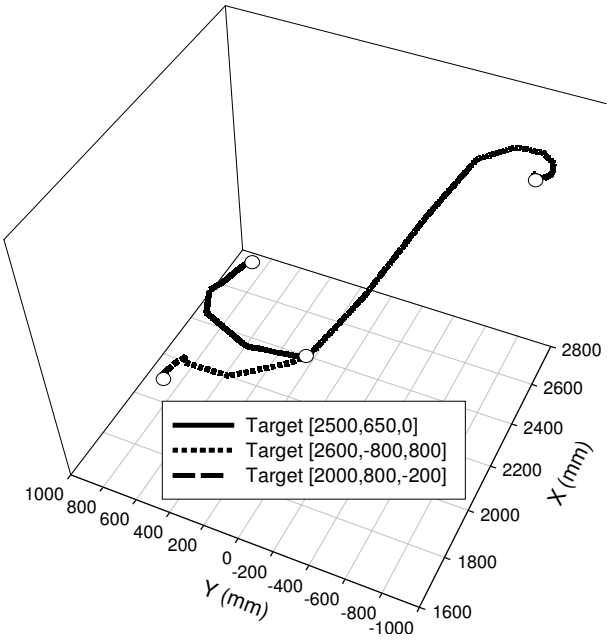

Figura 51. *Exp. P4. Representación 3D de las trayectorias para 3 objetivos diferentes*

ALCANCE DE OBJETOS CON PERTURBACIÓN

Para este caso, se han realizado experimentos cuando durante la trayectoria para alcanzar un objeto, éste cambia bruscamente su posición. Así, en el experimento P5, se ha realizado un cambio instantáneo de la posición de la esfera durante el alcance. Los resultados se muestran en las Figuras 52 a 54.

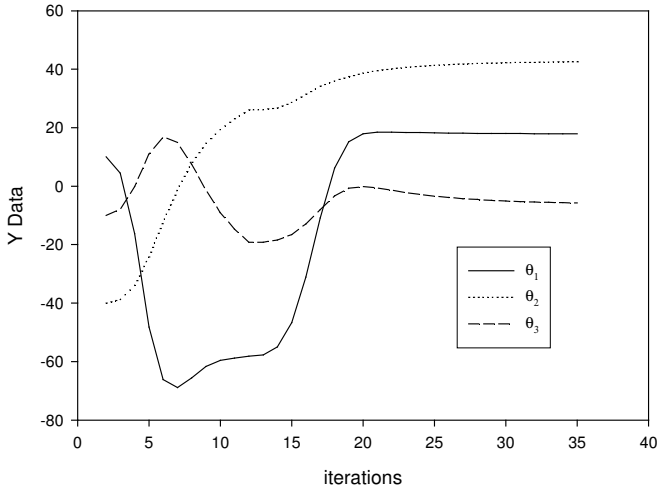

Figura 52. *Exp. P5. Evolución de las posiciones de articulación del robot ante una perturbación en la posición del objetivo*

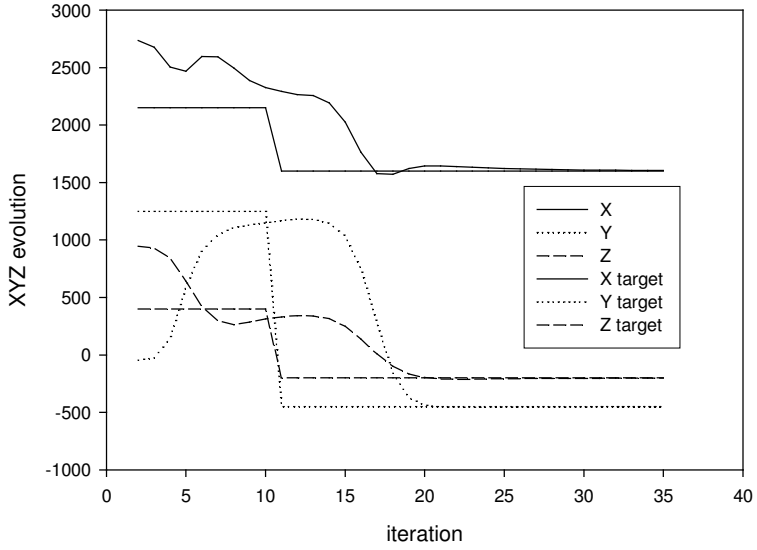

Figura 53. *Exp. P5. Evolución del efector final y del objetivo cuando ocurre una perturbación*

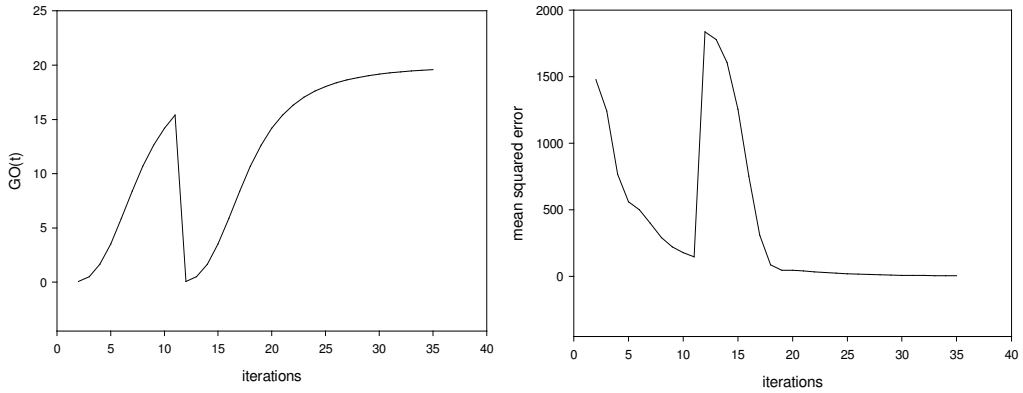

Figura 54. *Exp. P5. Evolución de la señal Go(t) y del error entre objetivo y posición actual del efector final, ante una perturbación*

ALCANCE DE OBJETOS EN MOVIMIENTO

Para este caso el modelo HRBF se ha ejecutado para intentar alcanzar objetos en movimiento. Obviamente, el éxito del alcance se encuentra, entre otras cosas, en la velocidad relativa entre el objeto y el brazo robot. El comportamiento del modelo HRBF para estos casos ha sido analizado con un movimiento aleatorio de la esfera (experimento P6) y otro circular en el plano YZ (experimento P7). Los resultados obtenidos se muestran en las siguientes figuras:

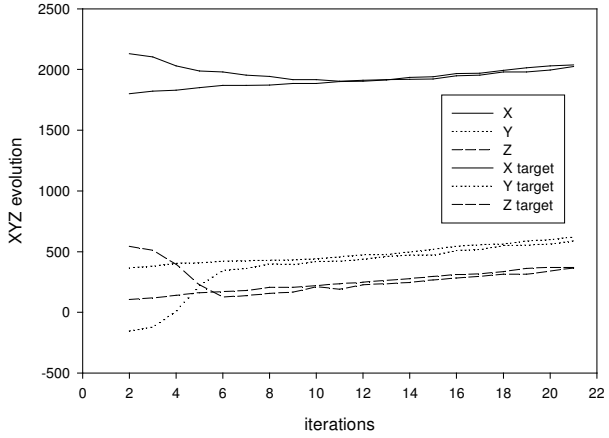

Figura 55. *Exp. P6. Evolución del efector final y del objeto*

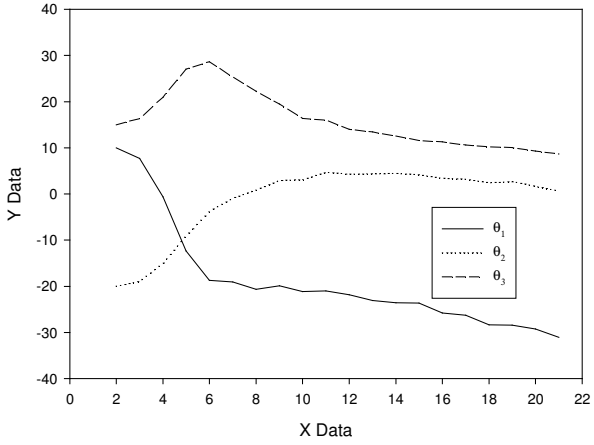

Figura 56. *Exp. P6. Evolución de la posición de las articulaciones del brazo robot*

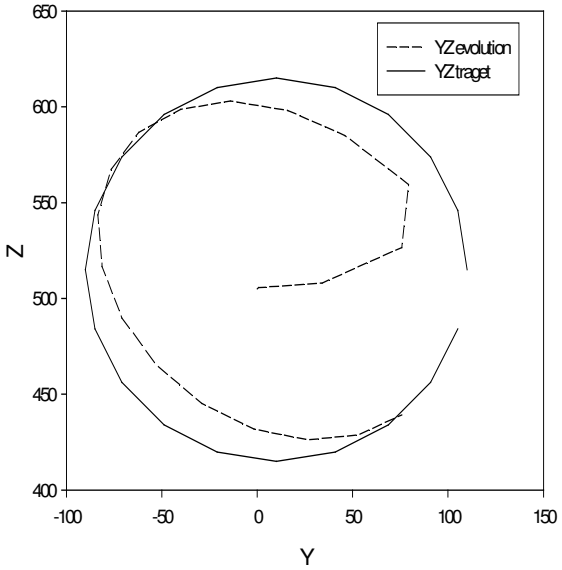

Figura 57. *Exp. P7. Evolución del efector final y del objeto*

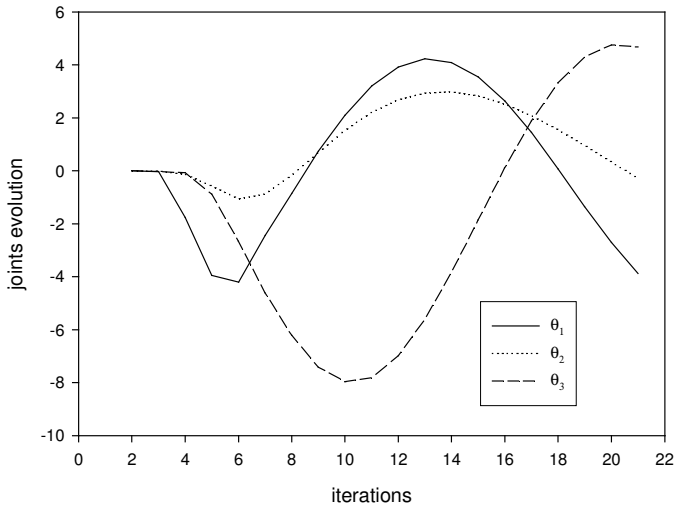

Figura 58. *Exp. P7. Evolución de la posición de las articulaciones del brazo robot*

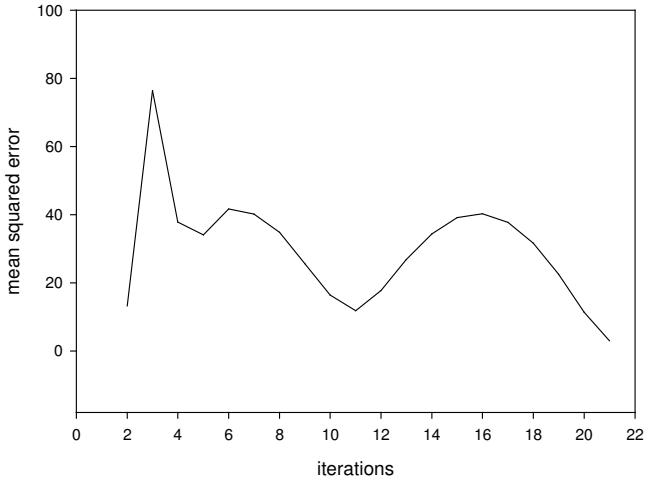

Figura 59. *Exp. P7. Evolución del error entre la posición del efector y la del objetivo*

9.1.4. Análisis comparativo de HRBF y ART-AVITE

Con el objetivo de establecer un análisis comparativo entre las dos estrategias de alcance propuestas en este trabajo, se ha realizado una serie de experimentos haciendo uso nuevamente del simulador VRS sobre un mismo modelo de robot (ABB-1400) con 3 grados de libertad. En ambos casos, se han realizado 600 entrenamientos de la red neuronal (600 movimientos aleatorios del brazo robot). Para la fase de aprendizaje del modelo híbrido ART-AVITE se han generado finalmente 23 celdas de aprendizaje. Para

evaluar el comportamiento de ambas estrategias en alcance de objetivos fijos, las posiciones iniciales del objeto y del efector final han sido: {100, 900, 1000} y {600, 850, 1000}, respectivamente. Los resultados obtenidos para la evolución del error, la aproximación del efector final hacia el objetivo, la evolución de las articulaciones del brazo robot y, finalmente, la comparación de la forma de la trayectoria de movimiento en ambos modelos, se presentan en las siguientes gráficas:

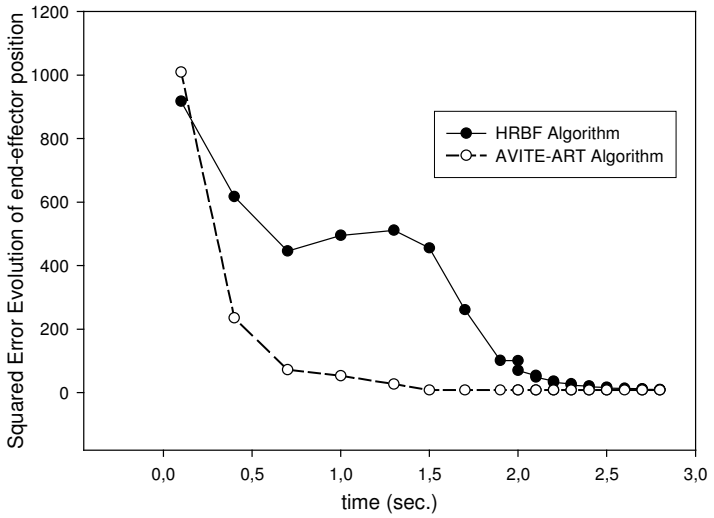

Figura 60. *Evolución del error en coordenadas espaciales para las coordenadas x,y, z del efector final del brazo robot, durante el movimiento de alcance*

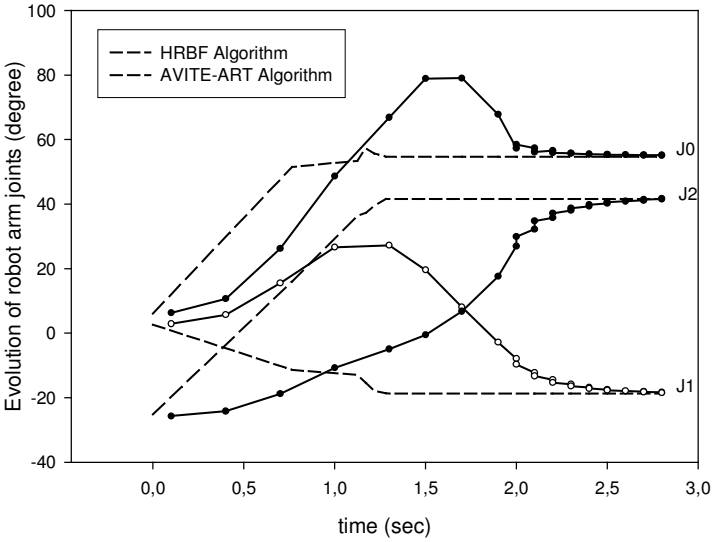

Figura 61. *Evolución del brazo robot en coordenadas motoras*

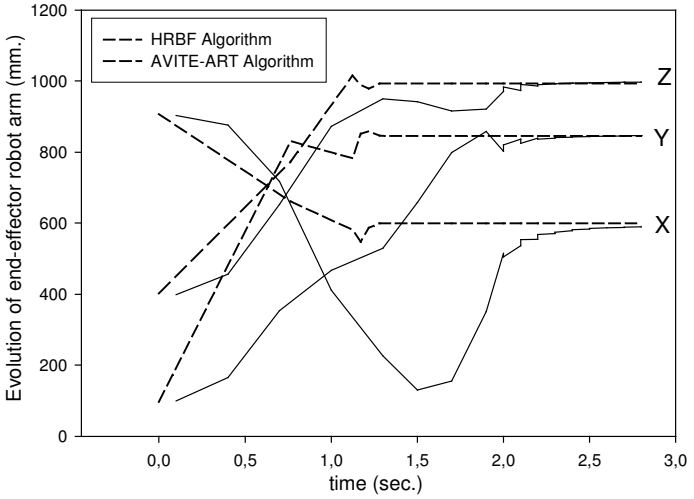

Figura 62. *Comparación de la evolución del efector final del brazo robot en ambas estrategias*

Para el caso de alcance de objetos cuando éstos cambian bruscamente de posición durante el movimiento de alcance del brazo robot, se ha considerado que el objeto se desplaza, en ambos casos, desde la posición {1000, 500, 900} hasta la {700, -900, 400}. Durante el entrenamiento de la red ART se han generado 37 celdas de aprendizaje, con el objeto de aumentar la velocidad en el alcance cuando se produce el desplazamiento del objeto. Las siguientes gráficas muestran la evolución del efector final para cada una de las dos estrategias.

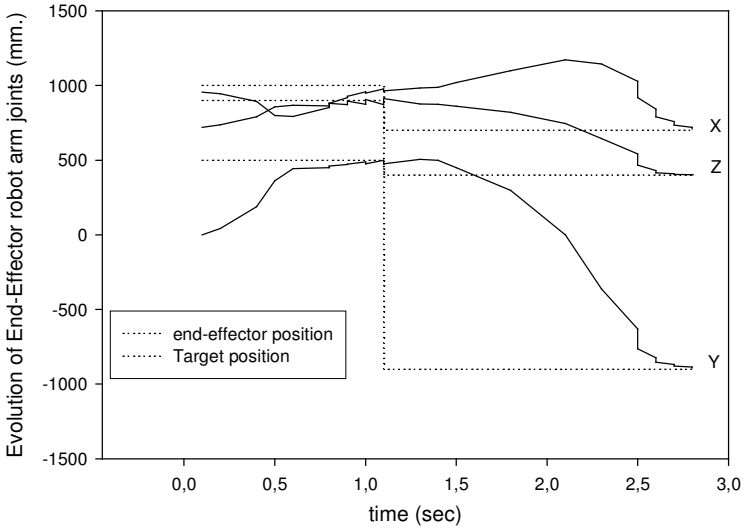

Figura 63. *Evolución de la posición espacial del efector final con el modelo HRBF*

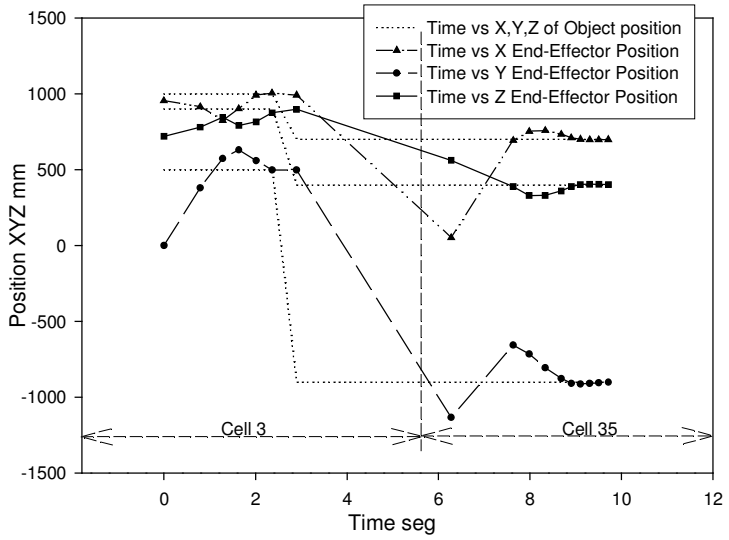

Figura 64. *Evolución de la posición espacial del efector final con ART-AVITE*

Como se puede observar de la figura 64, el cambio de posición del objeto, produce también un cambio de celda y, por tanto, un cambio en la matriz **Z** que permite compensar la diferencia de posición entre efector final y objetivo. Para ilustrar aún mejor la trayectoria que se realiza con cada uno de los modelos, cuando se quiere alcanzar un objeto fijo, se ha representado una gráfica 3D con ambas trayectorias, tal y como muestra la siguiente figura:

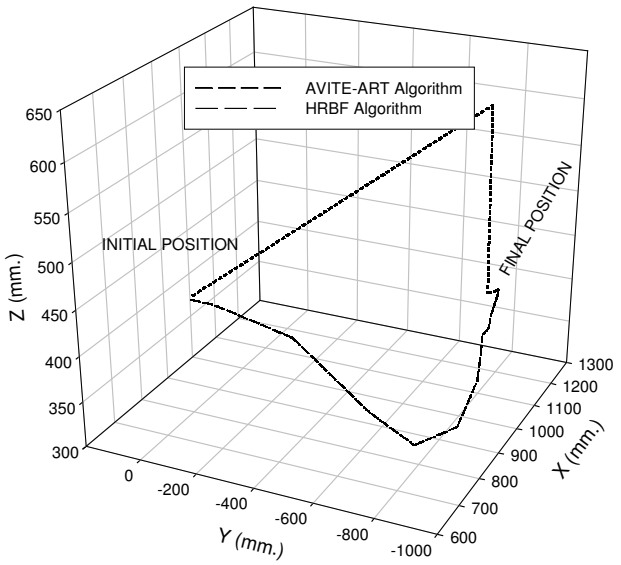

Figura 65. *Comparación de trayectorias 3D con ambas estrategias de alcance desde la posición inicial {900, 96, 400} hasta la del objetivo en {1200, -800, 400}*

En la figura anterior se puede observar claramente el comportamiento casi-lineal del modelo propuesto ART-AVITE, lo que permite que el alcance sea mucho más rápido que en el caso del modelo HRBF, con el que se realiza una trayectoria más curvilínea, debido a su característica fuertemente no lineal, aunque permite controlar mejor el error de precisión.

9.1.5. Análisis comparativo de aplicación de HRBF sobre dos plataformas robóticas simuladas

En este caso, se ha pretendido evaluar el comportamiento de uno de los modelos de alcance (el HRBF en este caso) en dos plataformas robóticas diferentes, incluyendo, además un cuarto grado de libertad para incluir redundancia en el movimiento del brazo robot. Estas plataformas simuladas son la del ABB-1400 y la del robot CRS-A255, utilizando, para ambos, los 4 grados de libertas siguientes: tronco, hombro, codo y rotación de la muñeca.

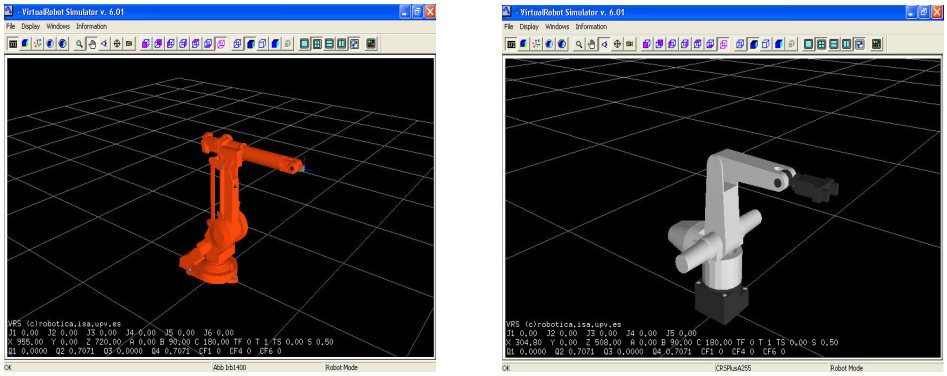

Figura 66. *Plataformas robots simuladas para testear la capacidad de adaptabilidad del modelo de alcance HRBF*

En ambos casos el número de entrenamientos en la fase de aprendizaje ha sido de 300, mientras que el número de neuronas consideradas ha sido de 500, lo que supone el doble de las que se han empleado en los experimentos anteriores. Esto es necesario, debido a que en este caso se tienen 4 grados de libertad, por lo que el aprendizaje debe ser mayor.

Después de entrenar separadamente la red HRBF con las dos plataformas, se han realizado operaciones de alcance de objetos cuando un desplazamiento brusco de su posición es producida. Los siguientes resultados demuestran que el modelo obtiene una gran precisión en ambas plataformas. El error final obtenido en el alcance ha sido de 9,91mm y 9,68mm, respectivamente.

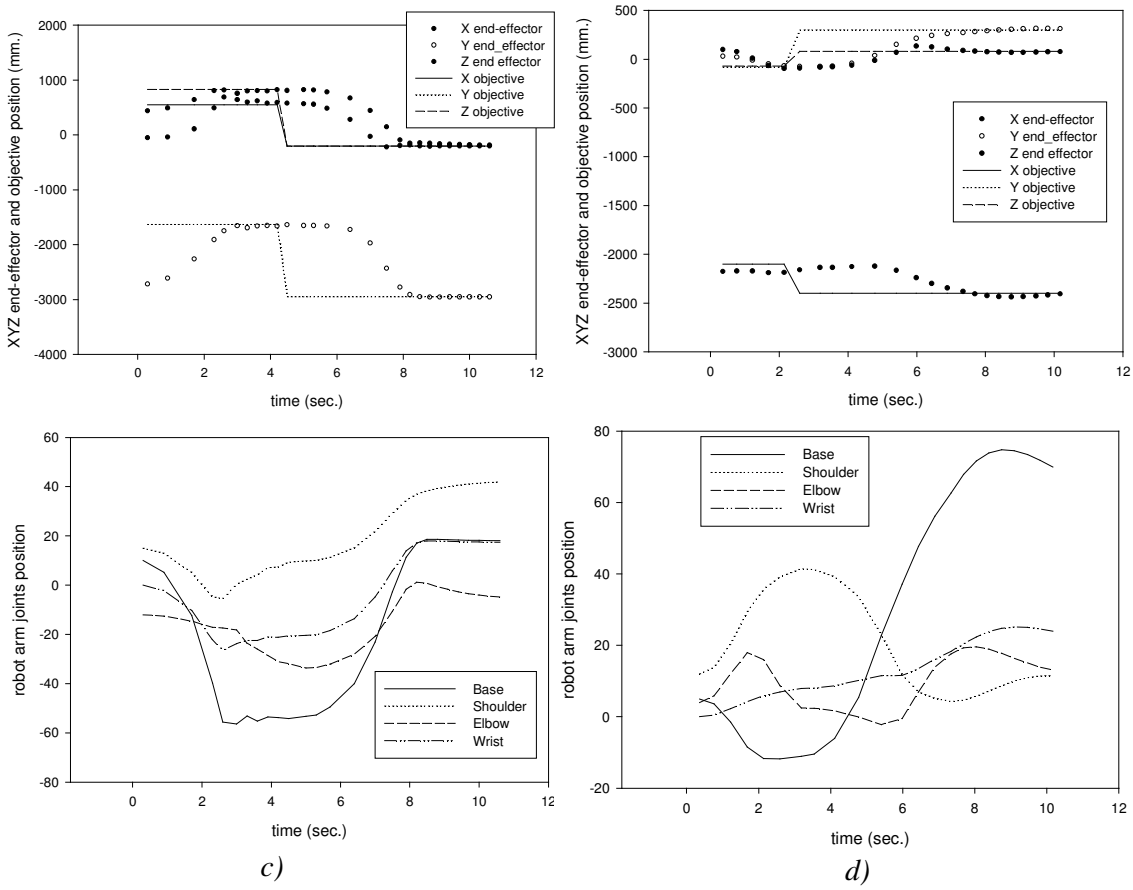

Figura 67. *Alcance de objetivos con el modelo HRBF en ambas plataformas robóticas considerando perturbaciones en la posición de los objetivos. La primera columna corresponde al robot ABB-1400, mientras que la segunda corresponde al CRS-A255. En a) y b) se representan las evoluciones del efector final y del objeto, mientras que en c) y d) se representan los movimientos de las 4 articulaciones consideradas en cada uno de los brazos robots*

Finalmente, se han realizado 3 operaciones de alcance de objetos fijos, para tres posiciones diferentes de los mismos, en cada una de las plataformas. Después de

ejecutar el algoritmo HRBF para generar el alcance en las tres posiciones, las trayectorias realizadas en cada caso y para cada plataforma han sido calculadas y representadas en la siguiente figura.

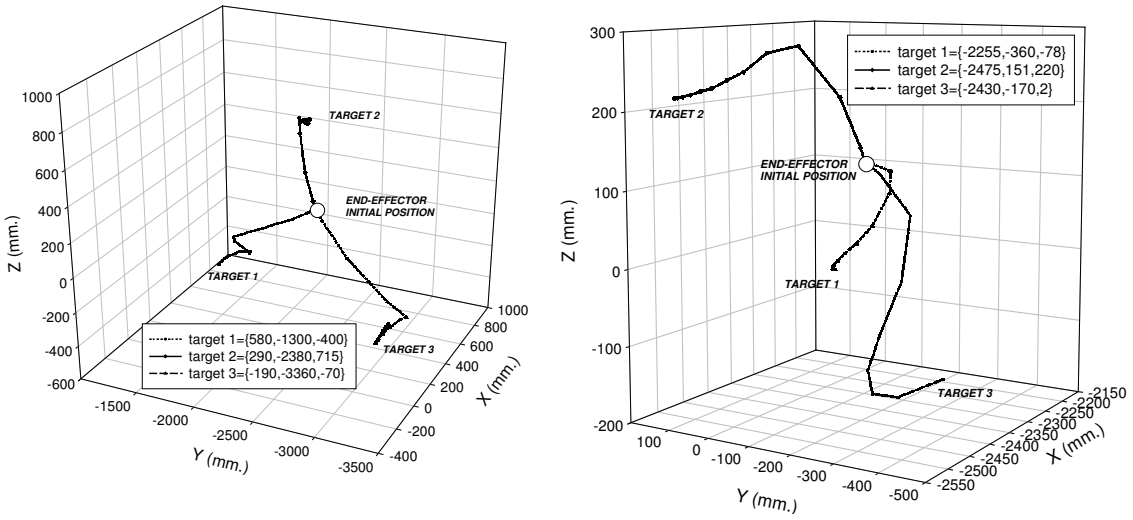

Figura 68. *Representación 3D de las trayectorias seguidas por el efector final de ambos robots para alcanzar 3 objetivos diferentes para cada caso.*

9.2 Plataforma Robótica Neurocor

Una vez que se han testeado los modelos de alcance en una plataforma simulada, las mismas aplicaciones ha sido trasladadas a la plataforma robótica del grupo Neurocor, formada por un brazo robot ABB-1400 de 6 grados de libertad, un cabezal estereoscópico con 5 grados de libertad y 2 cámaras a color para la detección del objeto y del efector final del brazo robot, y una pinza robot.

La posición 3D del efector final y del objetivo se obtiene por triangulación de las dos cámaras del cabezal robot y mediante un algoritmo de procesamiento de imagen (que nos es objetivo de este proyecto) que es capaz de distinguir colores diferentes para los objetos y para el efector final. Para ello, se colocan etiquetas de colores envolviendo cada uno de los objetos (rojo en este caso) que se quieren alcanzar, así como del efector final (de color amarillo). Con el objeto de simplificar la programación y el

entrenamiento, sólo se han considerado 3 grados de libertad para el ABB (tronco, hombro y codo) puesto que los otros 3 sólo intervienen para la orientación del efector final y no para el alcance. Para la experimentación con el robot real sólo se ha implementado el modelo HRBF, por ser más sencillo en su programación. Primeramente se realiza la fase de entrenamiento para la cuál, se ordena al robot que realice un número similar al empleado para las simulaciones, de movimientos aleatorios. En cada movimiento el cabezal robot calcula la posición 3D del efector final y se almacena en una matriz junto con las coordenadas de articulación (tronco, hombro y codo) del brazo robot. Una vez concluida la fase de entrenamiento, se calculan los pesos de las neuronas de la red HRBF, así como la posición de los centros y anchuras (varianza) de las Gaussianas. Finalmente, se ejecuta el modelo de alcance.

Para ello se coloca el objeto rojo en una determinada posición, y el robot en una determinada postura. A continuación el cabezal detecta la posición del objetivo y la del efector final y calcula la distancia o vector diferencia DV entre ambas posiciones. Este vector es cargado en la entrada de la red HRBF que calcula el incremento de posiciones de articulación que hay que proporcionar al brazo robot. Este incremento es multiplicado por la señal *Go* para modular o controlar la velocidad de alcance. Una vez ejecutado este primer movimiento de aproximación, se vuelven a repetir estos pasos para acercar en lazo cerrado el efector final al objeto. Aunque estas operaciones se documentan con archivos de vídeo, las siguientes figuras muestran una secuencia de alcance con una mano robot utilizando el modelo HRBF.

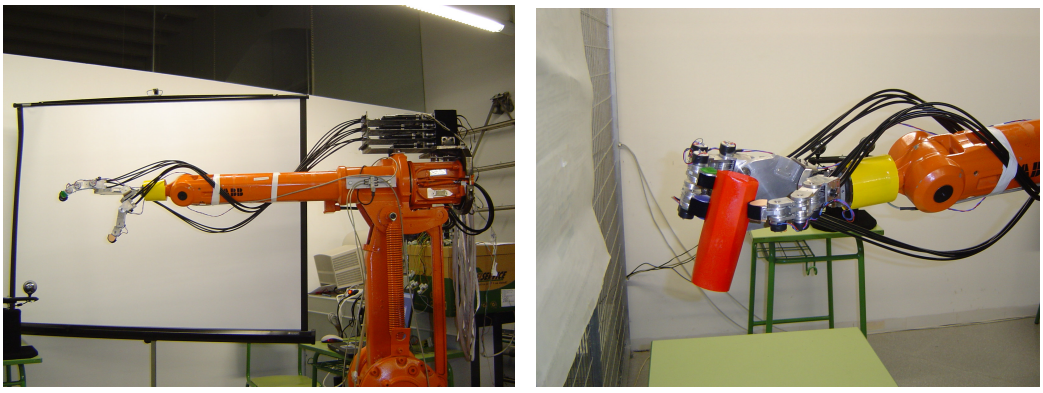

Figura 68. *Secuencia de un movimiento de alcance en la plataforma real Neurocor, con la mano robot incorporada como efector final del brazo.*

Conclusiones

En este proyecto se ha abordado uno de los temas que aún hoy en día permanece en constante estudio en el campo de las aplicaciones neuronales para la robótica de alcance y agarre y el aprendizaje de la cinemática inversa del robot. En concreto se han planteado dos estrategias de generación de trayectorias para alcance de objetivos con brazos robots mediante el uso de técnicas adaptativas. Para ello se han planteado los principales problemas existentes con los robots industriales para este tipo de tareas, que son fundamentalmente la precisión y la velocidad de alcance. Esto ha motivado el diseño e implementación de una arquitectura en lazo cerrado basada en la red HRBF desarrollada en la Universidad de Boston y, posteriormente, una arquitectura nueva, desarrollada a lo largo de este proyecto, que permite aportar una solución al compromiso entre precisión y velocidad de alcance. Esta última estrategia, que trabaja prácticamente en lazo abierto, está basada en cómo funciona el sistema senso-motor humano para la realización de este tipo de tareas y puede ser aplicado a otros campos diferentes como el de la planificación de celdas en telefonía celular. Con el fin de validar estas dos estrategias, se ha implementado un interfaz y una aplicación sobre un simulador CAD virtual denominado Virtual Robot Simulator, desarrollado por el Departamento de Ingeniería de Sistemas y Automática de la Universidad Politécnica Valencia. En esta herramienta se han simulados dos plataformas robóticas diferentes cuyas réplicas reales se encuentran operativas en el laboratorio Neurocor de la UPCT. El aprendizaje y ejecución de los pesos neuronales en ambas estrategias de alcance se ha realizado sobre distintas configuraciones del brazo robot (con 3 y 4 grados de libertad) y con objetos tanto estáticos como en movimiento. Los resultados obtenidos permiten validar las dos estrategias desarrolladas y evaluar en cada caso la más conveniente desde el punto de vista de su comportamiento en cuanto a velocidad, precisión y carga computacional. Posteriores implementaciones sobre la plataforma real han permitido cerrar la experimentación propuesta como objetivo al inicio de la realización de este proyecto. Como aplicación inmediata de los modelos propuestos se aporta una solución al problema de las arquitecturas cerradas de los robots industriales, cuyos controladores precisan de un elevado tiempo para la ejecución de comandos de movimiento, lo que impide que se puedan utilizar con redes adaptativas en entornos dinámicos

Bibliografía

[1] J.L. Pedreño Molina, Arquitectura neuronal de inspiración biológica para la integración táctil en sistemas visuo-motores. (2001). Tesis Doctoral.

[2] Haykin S., "Neural Networks, a comprehensive foundation". Capítulo 20.

[3] Guenther F. and Barreca D., "Neural models for flexible control of redundant systems". Pietro G. Morasso and Vittorio Sanguineti (eds.) *"Self organization, Computational Maps, and Motor Control"*, pp. 383-421, , Elsevier 1997.

[4] J.R. Hilera y V.J. Martínez, Redes Neuronales Artificiales. Fundamentos, modelos y aplicaciones. Ed. Ra-ma 1995. Capítulos 6 y 7

[5] Virtual Robot Simulator (VRS), 2003. Grupo de Robótica. Departamento de Ingeniería de Sistemas y Automática. Universidad Politécnica de Valencia