Control de Movimiento de un Robot Utilizando Mapas Autoorganizados de Kohonen

Oscar Sánchez – Raúl Fernández
2005
Universidad de Buenos Aires – Departamento de Computación

Contenido

Abstract

Los mapas autoorganizados como Extended Kohonen Maps (EKMs) pueden ser muy útiles en el aprendizaje de la coordinación sensor-motor para la realización de tareas por medio de robots móviles. Este trabajo utiliza un nuevo punto de vista denominado EKMs cooperativos, el cual fue introducido por Kian Hsiang Low [L04], de forma de poder utilizarlo en la resolución de tareas complejas. Se presentan dos métodos utilizando este tipo de redes neuronales, como son el mapeo directo y el mapeo indirecto, analizando las ventajas y desventajas de cada uno de ellos. Finalmente se introduce una modificación al mapeo directo de forma de obtener un movimiento de robot suave y preciso, de forma de poder operar en un espacio continuo en lugar de discreto de controles de robot.

Introducción

La mejor forma de comenzar el estudio de robótica según Ronald C. Arkin es planteándose una pregunta: Si uno pudiera crear robots inteligentes, ¿como deberían ser, y que cosas deberían ser capaz de realizar?. Contestar la primer parte de esta pregunta – ¿Cómo debería Ser? – requiere una descripción de la estructura física del robot (su apariencia) y su performance (comportamiento). Sin embargo la segunda parte de la pregunta – ¿Que debería ser capaz de hacer? – encasilla la respuesta para la primer parte. Los robots necesarios para mover objetos deben poder trabajar con ellos, robots que se quiere que puedan atravesar puertas necesitan sistemas de locomoción capaz de realizar movimientos aun en condiciones adversas, robots que deben cumplir funciones durante la noche necesitan sensores para poder operar en estas situaciones.

Pero, ¿qué es realmente un Robot?, existen muchas definiciones o respuestas a esta pregunta. Para el común de los individuos un robot es un dispositivo mecánico, el cual en la mayoría de los casos tiene semejanza en cuanto a su estructura a la de un ser humano y son capaces de realizar tareas como lo haría un ser humano. Este tipo de percepción es la que hace que las investigaciones en este campo suelan ser para concederle a estas "máquinas" capacidades perceptivas como son el oído, vista, tacto entre otras. El presente trabajo surgió de la misma forma, haciendo la pregunta, porque un robot no puede tener la capacidad de "pensar" de la misma forma que lo hace un ser humano?. Esto motivo a los autores al estudio de las redes neuronales artificiales, donde las investigaciones tratan de buscar modelos neuronales que puedan ser plasmados en una computadora siguiendo el comportamiento del cerebro para determinadas circunstancias. Entonces. ¿por qué no hacer que un robot posea esta capacidad?, esta pregunta se la han hecho muchos, de ahí el surgimiento de diversas disciplinas para resolver este problema.

La robótica en general y la robótica móvil en particular es un área de intenso estudio por la comunidad científica debido a las múltiples aplicaciones que ofrece. En el uso cotidiano del término se considera un robot a cualquier dispositivo mecánico que realiza tareas automatizadas ya sea por medio de un programa, por supervisión directa de un humano o mediante técnicas de Inteligencia Artificial[O04].

El empleo de brazos robóticos está implantado desde hace tiempo en la industria moderna, siendo su utilización muy extendida para la realización de tareas repetitivas o de riesgo: atornillar piezas, pintar coches, realizar soldaduras, desplazar objetos pesados, etc. Estos aparatos trabajan en entornos controlados y preparados específicamente para ellos, a los que el operador humano no suele tener acceso mientras están en funcionamiento. Aparte de los brazos robóticos industriales, existen otras utilidades que se les están dando cada vez con mayor frecuencia a máquinas de este tipo. Así, es cada vez más habitual la utilización de brazos robóticos para realizar operaciones quirúrgicas, sobre todo las relacionadas con la microcirugía y cirugía no invasiva debido al alto grado de precisión que se puede obtener. También se utilizan como implantes para miembros amputados, dentro de la ingeniería biónica. Estos robots tienen un grado de autonomía limitado y no se engloban en la categoría de robots móviles autónomos. Desde esta perspectiva, un robot es un agente con un cuerpo físico con capacidades motrices y situado en un entorno real con el que el robot

interactúa[O04]. La autonomía en un agente de este tipo requiere de capacidades complejas como trabajar durante largos períodos de tiempo sin supervisión en entornos dinámicos en los que es imposible predecir el estado del mismo. Esto implica a su vez capacidad de navegar en el entorno, además de técnicas de autoalimentación, detección de fallos, auto-regulación y auto-generación de reglas de comportamiento. El grado de autonomía en los robots móviles disponibles hoy día está todavía lejos de ajustarse a esta definición. Es de destacar que el estado del arte en el desarrollo de sistemas robóticos autónomos no se corresponde con lo que de ellos se espera por parte de la sociedad. Los avances en la tecnología no se reflejan a la misma velocidad en la robótica móvil. Seguramente, la pregunta más oída por un investigador del área en ámbitos sociales sea: ¿Para cuándo uno que pase la aspiradora o que corte el césped? Tareas que, aunque no triviales, no suponen grandes dificultades, están lejos de ser realizadas por un robot móvil con un grado de éxito satisfactorio. Obviamente, las aplicaciones de este tipo de sistemas son infinitas, desde robots guías en museos y centros públicos, hasta la exploración marítima y espacial. Por otro lado, hoy en día se están construyendo dentro de la robótica autónoma muñecas o mascotas artificiales de compañía para personas mayores, debido a que en los países desarrollados la esperanza de vida crece pero no el índice de natalidad, lo que hace que existan cada vez más personas ancianas que viven en soledad y que pudieran verse beneficiadas por robots de este tipo. Así, se han construido robots imitando a "mamíferos" que muestran cierto grado de sensibilidad en el trato con las personas, pero suponen todavía un costo no abordable para lograr un uso generalizado.

No hay soluciones "mágicas" para resolver los problemas; incluso una tarea tan simple como la de evitar obstáculos depende fuertemente tanto de la morfología del robot y de los sensores de que dispone, como del propio entorno y del grado de dinamismo que presenta el mismo. La navegación con objetivo en entornos con un grado de estructuración pequeño, más allá del puro deambule, algo casi "instintivo" para los animales, es uno de los mayores problemas a resolver para conseguir robots fiables y convertirlos así en algo rutinario en los espacios comunes.

Hasta prácticamente finales de los 80 el rumbo tomado para el desarrollo de agentes autónomos se fundamentaba en la Inteligencia Artificial. Se consideraba que, una vez conseguida la "mente artificial inteligente", el paso de añadirle un cuerpo a esa "mente" sería, si no inmediato, un problema de ingeniería de menor dificultad. Claro reflejo de esto es el vago número de referencias a robots físicos reales que puede hallarse en la literatura. A finales de los 80, R. A. Brooks revolucionó el área atacando la filosofía seguida hasta entonces desde su base. Postuló que para conseguir robots autónomos inteligentes debe empezarse por construir agentes físicos reales con capacidades básicas que puedan actuar en entornos reales, y que una vez logrado ese objetivo, se podría aspirar a ir aumentando las habilidades del robot y, en definitiva, su grado de autonomía. La irrupción de las ideas de Brooks dió lugar a lo que hoy día se conoce como Sistemas basados en el comportamiento.

Trabajos Relacionados

Las dificultades de alcanzar un funcionamiento en tiempo real y control exacto mientras que se ejecuta una tarea sofisticada en nuestro mundo complejo e imprevisible, sobre todo, el problema fundamental del movimiento dirigido hacia un objetivo o meta determinada libre de colisiones y autónomo en un ambiente no estructurado, ha sido el objetivo de los esfuerzos en muchas investigaciones y se ha dedicado considerable atención a estudiar la coordinación entre sensores y motores para resolver este problema.

Es aquí donde se estudia la asociación de señales que provienen de varios sensores a comandos de motor en vista a una tarea dada. En particular esta tarea es frecuentemente necesaria en la inspección de espacios desconocidos, limpieza, vigilancia, búsqueda y rescate y guías. Todas estas aplicaciones requieren de un robot móvil que realice una tarea realizando movimientos que no le produzcan impactos con obstáculos u otros robots. Podemos, entonces, abordar el problema de control de movimiento como sigue: dado un estado inicial descripto por un vector (input) que contiene la configuración de los sensores, determinar las secuencias de control de motor, libre de colisiones, que mueve el robot hacia un objetivo deseado.

Según Kian Hsiang Low, existen básicamente tres clases de algoritmos que han sido investigados para aprender el control de sensor-motor que es requerido para una tarea dada: multivariate regression, reinforcement learning y feature mapping. La primer clase formula el problema como un problema no lineal de regresión multivariada y entrena un perceptrón multicapa para realizar un mapeo continuo desde el espacio de sensores al espacio de control de motor. Este método ofrece una buena capacidad de generalización. Sin embargo para entrenar a la red , se deben obtener ejemplos de entrenamiento, indicando para cada situación la acción a tomar en cada paso que posea el robot desde su estado inicial hasta su objetivo. Este proceso de búsqueda de ejemplos es muy dificultoso y tedioso, si no imposible para un robot móvil.

En ausencia de señales de errores cuantitativos precisos para el entrenamiento, los algoritmos de aprendizaje por refuerzo pueden ser usados si señales feedback cualitativas están disponibles. Sin embargo, sufre de problemas de generalización y continuidad. Muchos métodos de aprendizaje por refuerzo codifican estados de sensor discretos y comandos de motor que no pueden ser aplicados directamente a dominios de sensor-motor continuos de tareas de control en el mundo real. Una discretización a priori del espacio continuo introduce estados ocultos y una generalización débil. En combinación con funciones de aproximación son capaces de generalizar a través de entradas de sensor y el espacio de salida de control de motor, superando la limitación. De todas maneras, la generalización con funciones no garantiza que el algoritmo puede aprender para producir comandos de motor continuos que puedan variar suavemente y en forma precisa en respuesta a continuos cambios en el estado de sensores. Como efecto los algoritmos de aprendizaje por refuerzo son combinados con funciones de aproximación mapeando la entrada continua de sensores a comandos discretos de control de motor, que es exactamente lo que hace un EKM Para resolver el problema, algunos métodos mapean al espacio continuo de control de motor pero son modificados con búsquedas muy lentas iterativas para una acción óptima.

La tercera clase utiliza Mapas Autoorganizados de Kohonen como el Extended Kohonen Map (EKM) que particiona el espacio continuo de entradas (o salidas) en regiones locales (Kohonen, 2000). La capacidad de generalización de feature mapping se origina desde la autoorganización durante la etapa de entrenamiento tal que cada neurona es entrenada para mapear una región local sensorizada a una salida de control de motor deseada.

En un perceptrón multicapa todos los datos de entrenamiento son usados para obtener un modelo global o representativo. Por lo tanto, durante el aprendizaje, todos los pesos de la red son susceptibles de obtener interferencia negativa que puede derivar a un cambio dinámico de la distribución de datos. Por otro lado, un Mapa Autoorganizado obtiene regiones localizadas de datos, en lugar de la región entera de interés, dentro de modelos locales o representativos, localizando los efectos de interferencias. Como consecuencia, el aprender un nuevo dato de entrenamiento afecta solo unos pocos pesos en el Mapa a diferencia de lo que ocurre con MLP (Multilayer Perception). El costo de entrenar a la red en un Mapa Autoorganizado se mantiene bajo dada la topología a través de la cual las neuronas involucradas en el aprendizaje en un paso dado sólo son un subconjunto de vecinos a ella. Inicialmente el vecindario es largo, resultando en un aprendizaje rápido, durante el proceso de aprendizaje, el tamaño del subconjunto se va reduciendo para refinar el mapeo haciéndolo cada vez más local. Esto permite una performance eficiente y un entrenamiento preciso sobre muchas neuronas y facilita la escalabilidad del Mapa Autoorganizado sobre un mayor número de neuronas para mejorar la exactitud.

RBF o Basis function network es otro tipo de modelo de red similar al EKM, en cuanto particiona el espacio de entrada en regiones locales. Sin embargo es arquitecturalmente diferente a un EKM en que cada dato de entrada de sensor es reducido a la activación de funciones básicas, que son linealmente mapeadas a un control de salida independiente. Para hacer esto, los pesos de salida de todas las neuronas RBF son requeridas de forma de predecir el movimiento para alcanzar el objetivo. En contraste, un EKM usa solamente los pesos de salida de las neuronas ganadoras para mapear cada entrada de sensor a un control de salida. Durante el aprendizaje, RBF actualiza todos los pesos de salida con cada dato de entrenamiento, por otro lado, solamente los pesos de salida de las neuronas ganadoras y sus vecinos son actualizados. Por este motivo RBF experimenta mucha mas interferencia durante el aprendizaje online que el EKM.

Walter and Ritter propusieron un ensamble de múltiples mapas autoorganizados llamado PSOM jerárquico (Parameterized Self-Organizing Maps) para aprender el control sensor-motor de un brazo de robot bajo diferentes contextos. PSOM es una variante que puede aprender con un conjunto reducido de datos de entrenamiento y sin embargo mantiene buena exactitud. Para hacer esto, PSOM realiza una interpolación sobre un mapeo continuo múltiple utilizando un conjunto reducido de funciones predefinidas e independientes básicas y vectores de pesos construidos directamente de los datos de ejemplos. Para asegurar buena interpolación, los datos son ordenados topológicamente para obtener los vectores de pesos. Para un PSOM jerárquico cada PSOM es entrenado separadamente, resultando en diferentes valores de pesos para diferentes PSOM, en contraste con el trabajo realizado por Kian Hsiang Low en el cual se utilizan varios EKMs donde todos son entrenados simultáneamente para obtener el

mismo valor de pesos. Igualmente entrenar EKMs se hace en forma online y no offline como es el caso de PSOM.

Sistemas Basados en Sensores

Dado un robot de una determinada forma y tamaño, el conocimiento para un suave movimiento puede ser obtenido y expresado a partir de un conjunto de pares <percepción, acción>. Cada par refiere a un paso unitario en la trayectoria del robot y describe una configuración dada de sensores y la acción seleccionada para ella. Mas específicamente la percepción o configuración puede ser llevada a cabo con la lectura de una cantidad fija de sensores de proximidad, junto con la dirección y distancia del objetivo relativa al robot. Este par <percepción, acción> es la base para construir automáticamente sistemas basado en sensores, para este propósito puede ser utilizado un aprendizaje supervisado. El par <percepción, acción> es aprendido incrementalmente siguiendo los siguientes pasos:

- El sistema obtiene la "percepción" como un input.
- Se genera una acción tentativa acorde al conocimiento actual
- Esta acción es comparada con la acción que debería haber sido realizada y la diferencia entre estas es usada para cambiar y mejorar el comportamiento del sistema

La secuencia de pasos es repetida para cada ejemplo disponible. Como sistema basado en sensores se utiliza una Red Neuronal Artificial.

Como se puede observar el aprendizaje es difícil y no puede ser logrado solamente entrenando a la Red Neuronal con miles de ejemplos. Para facilitar el aprendizaje se puede tomar ventaja de una buena característica de los problemas de búsquedas de caminos, denominada "*similares percepciones requieren similares acciones" (similar perception require similar actions)*. Esto no es siempre verdadero pero si lo es en la mayoría de los casos, por ejemplo, dado una percepción de entrada la red neuronal primero determina cual es la percepción mas parecida según la experiencia de percepciones aprendidas hasta el momento, y segundo se dispara la acción asociada a la percepción seleccionada en el primer paso. El primer paso es una operación de agrupamiento que apunta a agrupar juntas similares experiencias dentro de una única percepción representativa. De esta forma similares percepciones contribuyen al aprendizaje de la misma (o similar) acción, mientras que percepciones diferentes (percepciones que requieren diferentes acciones) no interfieren unas con las otras dado que son mapeadas en diferentes percepciones representativas por el agrupamiento.

El primer paso de este procedimiento requiere de un aprendizaje, esto puede ser logrado con una red neuronal como lo es un Mapa de Kohonen.

Mapa de Kohonen

Un Mapa de Kohonen es una red de dos capas que consiste de una capa de entrada de neuronas directamente y totalmente conectadas con la capa de salida. Típicamente la capa de salida es una capa organizada como una grilla G. w_r es el vector de pesos de entrada asociado con la neurona ubicada en la posición r sobre G. La red es entrenada mediante un aprendizaje no supervisado sobre un conjunto de ejemplos {x(1).....x)T)}. Por cada ejemplo x, la siguiente secuencia de pasos es ejecutada:

1. x es presentada a la capa de entrada
2. Una competición entre las neuronas toma lugar. Cada neurona calcula la distancia entre su vector de pesos asociado y el vector de entrada. La neurona s cuyo peso es el más cercano a x es la ganadora de la competencia.
3. s es la neurona candidata para aprender el patrón de entrada, por ejemplo acercándolo un poco a él en el espacio de datos.
4. Un trato especial al algoritmo de Kohonen es que en el paso de aprendizaje es extendido también sobre un entorno centrado en la neurona ganadora s.

El Mapa de Kohonen puede ser utilizado para cuantización de datos: Un patrón de entrada x puede ser representado por un vector de referencial w_s que gana cuando x es presentado al Mapa de Kohonen, de aquí se desprende también, que entradas similares son mapeadas dentro de un mismo vecindario en la grilla.

Regresando al agrupamiento de percepciones, si un Mapa de Kohonen es entrenado con percepciones como entradas, entonces los pesos de las neuronas son el conjunto de percepciones representativas con lo cual cada neurona en la red representa una percepción, una dirección y una distancia hacia el objetivo.

El segundo paso del procedimiento que se mencionó en los párrafos anteriores, la red aprende a asociar las acciones a percepciones características. Para lograr esto se realiza una extensión al algoritmo de Kohonen que hace posible entrenar la red con un aprendizaje supervisado.

Extended Kohonen Map

Desde el punto de vista de la arquitectura, el Mapa de Kohonen es extendido agregando a cada neurona r sobre la grilla competitiva G un vector de pesos de salida z_r para almacenar el valor de salida de la neurona.

El proceso en una red EKM es el siguiente: Cuando una entrada x es presentada a la capa de entrada, las neuronas en G compiten para responder a ella. Esta competición involucra el peso de entrada de la neuronas w_r y consiste en el calculo de la distancia entre x y cada w_r. La neurona s cuyo peso de entrada es la mas cercana a x, es la ganadora de la competencia y su peso de salida z_s es tomado como la respuesta de la red a la entrada x. Durante la fase de entrenamiento, ambos, el patrón de entrada x y el valor deseado y propuesto de salida son aprendidos por la neurona ganadora y sus vecinos en la grilla. Este aprendizaje consiste en mover los vectores de pesos de entrada a las neuronas seleccionadas cercanas a x, y sus pesos de salida cercanos a y.

Este tipo de aprendizaje puede ser descripto como un entrenamiento competitivo-cooperativo. Competitivo porque las neuronas compiten a través de sus pesos de entrada para responder a un patrón de entrada. La regla es también cooperativa en que los valores de salida aprendidos por la neurona ganadora son parcialmente aprendidos por los vectores de salida de las neuronas vecinas.

EKM Cooperativo

En las últimas investigaciones de agentes autónomos se utiliza comúnmente la teoría de sistemas dinámicos y aprendizaje por refuerzo. Un agente o robot con un mecanismo de selección de acción (ASM) debe operar en un continuo espacio de estados y acciones tal que la interacción con un entorno complejo e impredecible puede ser versátil y robusto. En particular es esencial un alto grado de suavidad, flexibilidad y precisión en control de movimiento para ejecutar eficientemente tareas sofisticadas e interactuar con humanos.

Low Kian Hsiang propone utilizar una red neuronal autoorganizada para mapear el espacio continuo de estados en el espacio continuo de control de motor, para producir control de movimiento suave, eficiente y preciso [L04]. El mecanismo de selección de acción propuesto es implementado como un ensamble de EKMs, el cual es denominado EKMs cooperativo. Este método consiste de tres módulos.

1. Módulo de Alcance del Objetivo: constituido por 1 EKM para la localización del objetivo que es activado por la presencia del objetivo dentro del rango de alcance censado.
2. Módulo de Evasión de Obstáculos : constituido por h EKMs para la localización de obstáculos, los cuales son activados por la presencia de obstáculos dentro del rango de alcance que censa cada uno.
3. Módulo de Integración Neuronal: constituido por un EKMs de control de motor, que tiene el rol de interfase, integrando las señales de actividad desde los EKMs por competición y cooperación, produciendo señales de motor apropiadas.

Módulo de Alcance del Objetivo

Este módulo utiliza un EKM para autoorganizar el espacio censado de entrada U. Cada neurona i en el EKM tiene un vector de pesos $w_i=(\alpha_i,d_i)^T$, donde α_i, es la dirección del objetivo relativa al robot y d_i es la distancia al objetivo relativa al robot que codifica una región en U centrada en w_i. El EKM de localización del Objetivo es activado como sigue:

Dado una entrada *u* de localización del objetivo.

1. Determinar la neurona ganadora s en el EKM. La neurona ganadora es aquella cuyo vector de pesos $w_s=(\alpha_s,d_s)^T$ es el más cercano al vector $u=(\alpha_s,d_s)^T$:

   $$D(u,w_s)=\min_{i \in A(\alpha)} D(u,w_i)$$

   La distancia $D(u,w_i)$ es una distancia ponderada entre u y w_i:

   $$D(u,w_i)=\beta_\alpha(\alpha-\alpha_i)^2 + \beta_d(d-d_i)^2$$

   donde β_α y β_d son parámetros constantes. El mínimo es tomado sobre el conjunto $A(\alpha)$ de neuronas que codifican pesos muy similares con el ángulo α.

   En otras palabras, la dirección tiene prioridad sobre la distancia en la competición de las neuronas. Este método

permite al robot orientarse rápidamente de cara al objetivo mientras se mueve a el.

$$G_a(w_s, w_i) = \exp(-\frac{(\alpha_s - \alpha_i)^2}{2\sigma_{a\alpha}^2} - \frac{(d_s - d_i)^2}{2\sigma_{ad}^2})$$

2. Calcular la actividad de salida a_i de la neurona i.

   $A_i=G_a(w_s,w_i)$

   La función G_a es una Gausiana en la que el parámetro σ_{ad}

es mucho mas chico que $\sigma_{a\alpha}$, haciendo la Gausiana sensitiva al cambio de distancia e insensitiva al cambio de ángulo. Estos valores de parámetros alargan la gauseana en la dirección perpendicular a la dirección del objetivo.

Módulo de Evasión de Obstáculos

El robot tiene h sensores fijos de distancia alrededor de su cuerpo para detectar obstáculos, por lo cual cada sensor activado codifica una dirección fija α_j y una distancia variable d_j al obstáculo, relativa al robot. Cada sensor, con vector de entrada $u_j=(\alpha_j,d_j)^T$ alimenta un EKM de localización de obstáculo. Los EKMs de localización de obstáculos tiene el mismo numero de neuronas y valores de vectores de pesos que el EKM de localización de objetivo. Los EKMs son activados como sigue:

Por cada entrada u_j, j=1,....,h

1. Determinar la neurona ganadora s en el j-esimo EKM. El EKM de localización de obstáculo es activado de la misma forma que en el paso 1 de localización del objetivo.
2. Calcular la actividad de salida b_i de la neurona i en el j-esimo EKM:

   $B_i=G_b(w_s,w_i)$

   donde

$$G_b(w_s, w_i) = \exp(-\frac{(\alpha_s - \alpha_i)^2}{2\sigma_{b\alpha}^2} - \frac{(d_s - d_i)^2}{2\sigma_{bd}^2(d_s, d_i)})$$

la función G_b es una Gauseana alargada en el sentido y dirección α_s al obstáculo.

$$\sigma_{bd}^2(d_s, d_i) = \begin{cases} 2.475 & si \quad d_i \geq d_s \\ 0.02475 & en \ otro \ caso \end{cases}$$

Módulo de Integración Neuronal

Utiliza un EKM de control de motor para integrar la actividad de las neuronas en los EKMs de localización de objetivo y de evasión de obstáculos. Este EKM tiene el mismo número de neuronas y valores de vectores de pesos que en los EKMs de localización de objetivo y de evasión de obstáculos.

Este algoritmo funciona como sigue:

1. Calcula la actividad e_i de la neurona i en el EKM de control de motor.

$$e_i = a_i - \sum_{j=1}^{h} b_{ji}$$

   Donde a_i es la entrada excitatoria de la neurona i del EKM de localización del objetivo y b_{ji} es la entrada inhibitoria de la neurona i en el j-esimo EKM de evasión de obstáculos.
2. Determinar la neurona ganadora k en el EKM de control de motor. La neurona k es la de mayor actividad.

El EKMs de control de motor también tiene un vector de pesos de salida, el cual codifica la salida de la neurona.

Autoorganización de EKMs

A diferencia de los métodos fuera de línea (offline) existentes como los que utiliza MLP o EKM según la versión de Versino y Gambardella [VG95], en el presente trabajo se adopta un entrenamiento en línea (online). Inicialmente los EKM no han sido entrenados y los vectores de control de motor generados son incorrectos, pero a medida que el robot se mueve, la red se autoorganiza usando los vectores de control de motor *c* y los correspondientes desplazamientos *v* obtenidos, con el fin de mapear *v* a *c* indirectamente. Notar que se usa *v* para el entrenamiento, en lugar de *u*. Ya que los EKM no inicialmente no están entrenados el robot genera vectores de control de motor c inadecuados en respuesta a u, pero el robot aprenderá un mapeo erróneo del sensor-motor si se usa u para entrenar. Por el contrario, v es el desplazamiento real para el vector de control de motor c, usando v como el patrón de entrenamiento permitirá que el robot aprenda el correcto mapeo y se mueva de acuerdo a lo conocido. Por lo tanto el movimiento se vuelve más preciso. A esta altura se entiende que el entrenamiento en línea ajusta finamente el mapeo indirecto.

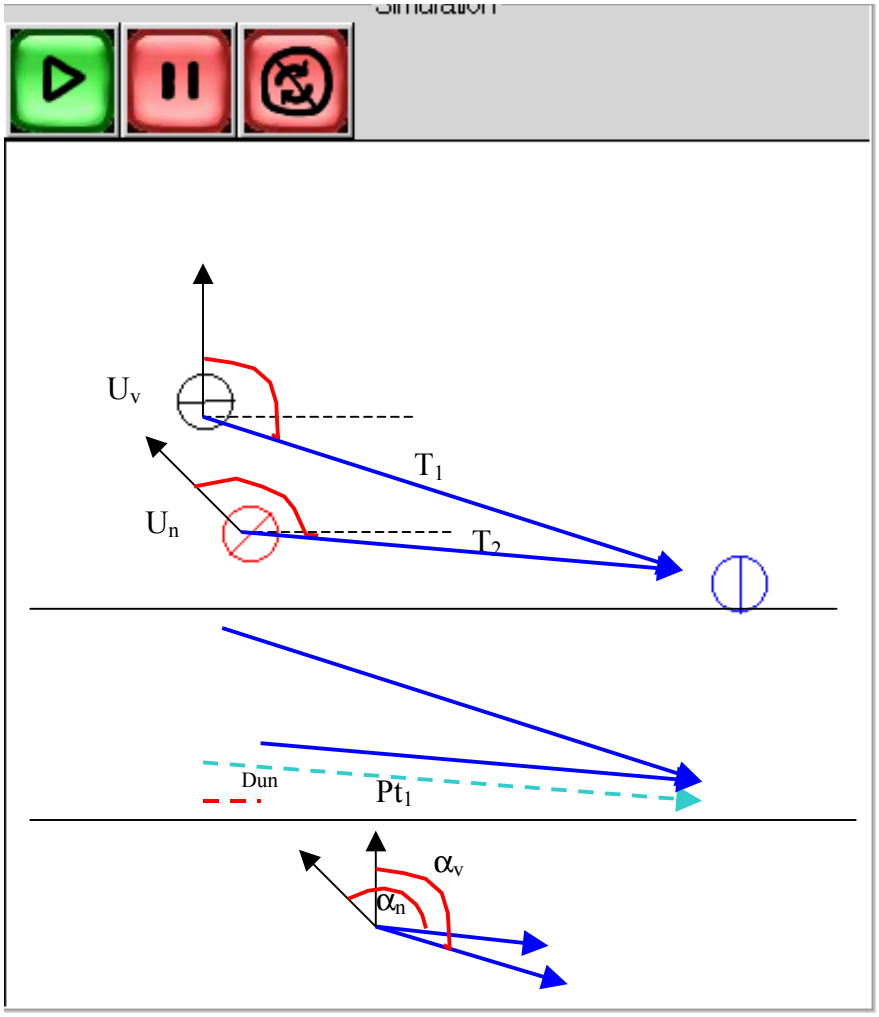
Entrenamiento

repetir

1. Obtener el vector censado u
2. Ejecutar el procedimiento de Seguimiento Objetivo y mover el robot
3. Obtener el nuevo vector censado u' y calcular el desplazamiento real *v* como la diferencia entre *u*' y *u*
4. Usar v como el patrón de entrenamiento para determinar la neurona ganadora
5. Ajustar los pesos w_i del EKM (todos los EKMs tienen los mismos pesos de entrada para las neuronas respectivas) en forma similar a como lo realiza el algoritmo de Kohonen
6. Actualizar los pesos de salida M_i en el entorno de la neurona k para minimizar una función de costo (ver mas adelante….), finalmente

$$\Delta M_i = \eta\, G(k,i)\, (c - M_i\, v)(w_s, w_i)\, v^T$$

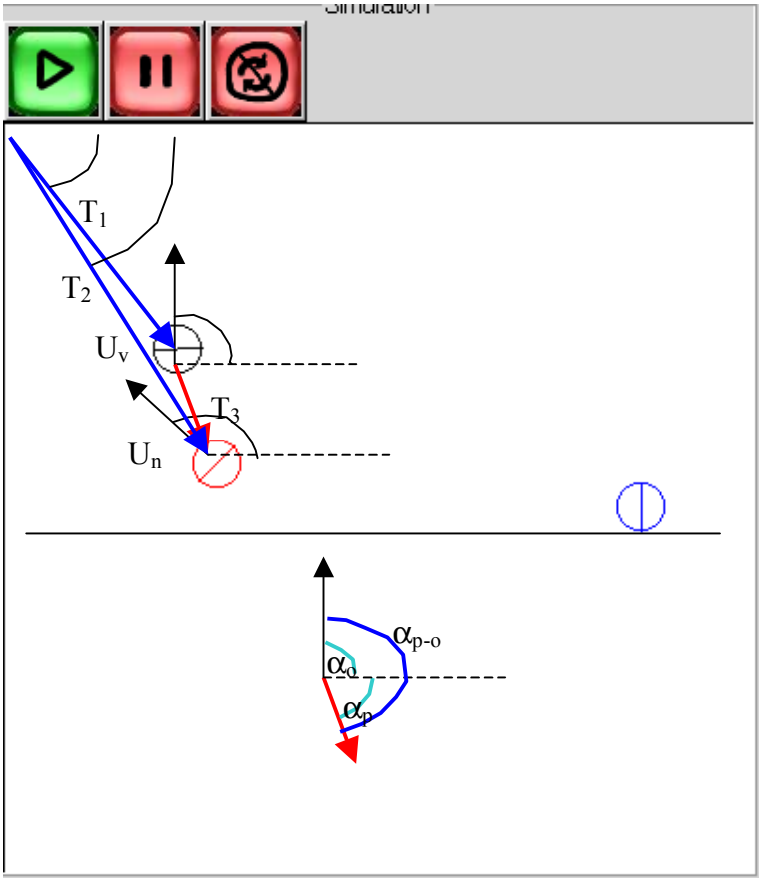
Calculo de Desplazamiento



Según Low U_v es igual al par (α,d) donde α es el ángulo que forma la posición del target relativa a la posición y orientación del robot y d es la distancia respectiva, en el dibujo ésto corresponde al vector T_1, lo mismo sucede con U_n=(α,d) en este caso representado por T_2, U_v para esta esquema corresponde a la posición en coordenadas polares que tiene el target respecto del robot en el momento t, y U_n corresponde a la posición que tiene el target en el momento t+1. Según Low para calcular el desplazamiento producido por el robot, el cual será utilizado para el aprendizaje, es calculado con la diferencia entre los vectores U_n y U_v (U_n - U_v), con lo cual la línea punteada verde corresponde a la proyección de T1 sobre T2 y la línea punteada roja corresponde a la diferencia de las distancias entre ambos vectores. La diferencia de los ángulos que forman estos vectores es la que será utilizada en el entrenamiento. Corresponde en el dibujo a α_{n-v}. Si se supone que el control de motor que produjo ese movimiento es C = (C_1,C_2), entonces se estaría entrenando a la red con el patrón de entrada V = (D_{un}, α_{n-v}) y el control de motor C = (C_1,C_2) como salida para ese desplazamiento. Si se estudia con detenimiento esta forma de calcular el

desplazamiento se puede observar en primer lugar que puede llegar a dar distancias negativas, por ejemplo si el desplazamiento del robot fuese hacia adelante, se estaría indicando que el robot se desplazó una distancia que es irreal. Esto puede ser subsanado tomando el módulo de la diferencia de las distancias, pero igualmente el concepto sigue siendo erróneo ya que esa misma distancia y ángulo pueden ser producidas por un control de motor totalmente diferente (ir hacia atrás o hacia adelante alienado al target, claramente son producidas por controles de motor diferentes, sin embargo se estaría mapeando el mismo desplazamiento teórico con controles de motor distintos), entre otras cosas esto es porque no se toma en cuenta la orientación del robot sino solo la posición del target con respecto al robot.

La idea del trabajo es que el robot aprenda sus movimientos, entonces en realidad lo que habría que calcular es el desplazamiento con respecto a si mismo, es decir el ángulo y la distancia que forma la nueva posición del robot respecto de la posición y orientación anterior. Esta idea se muestra en el siguiente gráfico que toma en cuenta la misma situación anterior.



La longitud de T_3 indica la distancia del desplazamiento real que tuvo el robot con respecto a su posición anterior, es decir cuanto avanzo o retrocedió. Luego lo que se debería considerar es el ángulo resultante de dicho desplazamiento, es decir, el ángulo que se obtiene de la coordenada polar del vector T3, así se determina realmente donde quedo posicionado el robot, de esta forma se consigue determinar el movimiento en cuanto a distancia recorrida, lugar donde quedo ubicado (relativa a donde se encontraba en el momento anterior) y la orientación en su posición final. Este movimiento o desplazamiento fue producido por el control de motor ejecutado, con lo cual cada vez que sea utilizado nuevamente se conseguirá el mismo desplazamiento.

Movimiento Suave

En la propuesta original del EKM Versino y Gambarderla, utilizan un mapeo directo entre los patrones de entrada y los controles de motor, esto es, en un EKM directo, cada neurona *i* tiene un vector de pesos $w_i=(\alpha_i,d_i)$ que codifican una región dentro del espacio de entrada, centrada en w_i, además tiene un vector de pesos c_i que codifica un control de motor como la salida que produce la neurona. Con un patrón de entrada *u*, la neurona ganadora *s* es determinada de forma que el vector de pesos w_i es el mas cercano a *u*, esta neurona ganadora produce una salida de control de motor c_s que mueve el robot. Notar que cualquier patrón de entrada *u* que caiga dentro de la región que enmarca w_s produce el mismo control de motor c_s.

Según el trabajo en el cual se basa el presente informe el método que es dado a llamar indirecto, produce un movimiento de robot suave que no se puede lograr con el método directo.

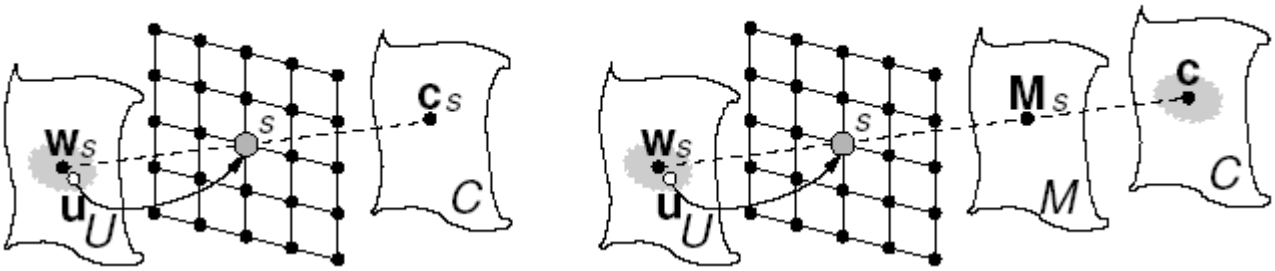
Si el control sensor-motor es un problema lineal, entonces el vector de control de motor *c* podría ser relacionado con el patrón de entrada *u* por medio de la ecuación lineal:

$$C=Mu$$

donde M es llamada matriz paramétrica de control de motor. El problema de control podría ser reducido entonces en determinar M a partir de los ejemplos de entrenamiento. Sin embargo en la práctica la coordinación sensor-motor es un típico problema no lineal debido a que un motor real toma un tiempo finito pero no cero para acelerar y desacelerar de forma de cambiar la velocidad.

Para resolver el problema no lineal descripto, el método indirecto realiza el entrenamiento particionando el espacio de entrada en regiones locales lineales, cada neurona *i* en el EKM indirecto tiene un vector de pesos w_i similar al de las neuronas en el método directo, pero a diferencia del mismo el vector de pesos de salida de la neurona *i* representa la matriz paramétrica de control de motor M_i en el espacio de parámetros en lugar del control de motor c_i. La matriz paramétrica de control de motor es mapeada en el control de motor c por medio de la ecuación $c=M_iu$.

En el desarrollo del método directo se mapean todos los patrones de entrada *u* dentro de una región del espacio de entrada, representada por la neurona *s*, al mismo punto discreto c_s dentro del espacio de controles de motor, esto significa que solamente un numero muy pequeño de puntos en el espacio de controles de motor son representados por la salida de las neuronas. En contraste el método indirecto mapea cada *u* dentro de una región del espacio de entrada en un punto *c* diferente dentro del espacio de control de motor, de esta forma el método es lineal y continuo, mapeando una región del espacio de entrada en una región en el espacio de salida (controles de motor). En el siguiente gráfico se puede apreciar esta idea.



Como se describe en párrafos anteriores al explicar de que forma es producida la Autoorganización, se detalla que para poder determinar la matriz paramétrica de control de motor M_i correspondiente a la neurona *i*, se utiliza el método del gradiente descendiente, lo que se quiere lograr con esto es actualizar M_i a partir de minimizar la función de error:

$$e = \frac{1}{2} G(k,i) \|\mathbf{c} - \mathbf{M}_i \mathbf{v}\|^2 .$$

con lo cual se quiere lograr que c se aproxime a M_iv, donde v es el desplazamiento que produjo el robot y con el cual se esta entrenando a la red.

Al estudiar la idea de Low en profundidad se observa que su planteo de mantener una linealidad en base a las matrices paramétricas de control de motor tiene el siguiente problema, tener un sistema de ecuaciones de esta forma, es decir c=Mv tiene infinitas soluciones, ya que las incógnitas son los valores de la matriz, con lo cual se esta resolviendo un sistema de dos ecuaciones y cuatro incógnitas (la matriz de control de motor es de 2x2). Con lo cual si bien se garantiza un continuidad lineal dentro del región que comprende a la neurona ganadora, esta continuidad lineal se pierde en el límite entre las regiones comprendidas por el área de la neurona ganadora y cualquiera de sus vecinas, cuestión que podría llegar a resolverse agregando dos ecuaciones más que restrinjan dos grados de libertad exigiendo la continuidad entre regiones contiguas.

En lugar de calcular una matriz paramétrica de control, en este trabajo se decidió seguir la idea general pero sin utilizar dicha matriz, el problema entonces esta en como utilizar un mapeo directo entre patrón de entrada y control de motor para producir un movimiento suave, la solución que se decidió implementar es realizar un cambio sobre el algoritmo de mapeo directo al calcular los controles de motor, calculando la salida como un promedio ponderado de los controles de motor de la neurona ganadora y sus vecinas en función de la distancia entre sus pesos y el patrón de entrada, con lo cual en lugar de tener un mapeo directo de control de motor se obtendrá un resultado continuo de los controles de motor que producen tanto la ganadora como sus vecinas.

Suponiendo un patrón de entrada *u*, su correspondiente neurona ganadora *s* y su vecindario *N(s)*, el control de motor que realizará el robot será determinado por la siguiente ecuación:

$$c = \frac{\sum_{i \in N(s)} \left(C_i * G\left(D_{Low}(i), D_{Low}(s) \right) \right)}{\sum_{i \in N(s)} G\left(D_{Low}(i), D_{Low}(s) \right)}$$

donde G(distancia_neurona(i) , desviación) es una función de gauss centrada en el patrón de entrada *u*.

Luego para el entrenamiento o Autoorganización, en lugar de mapear el control de motor ejecutado *c*, a la neurona ganadora, se deberá mapear un control de motor que mantenga la idea de producir controles de motor similares y continuos entre neuronas cercanas obteniendo en consecuencia movimientos suaves. Con lo cual se actualizara tanto el control de motor de la neurona ganadora como así también un vecindario de ella en la proporción que determine una función de gauss centrada en dicha neurona y tomando en cuenta el vecindario entre las neuronas en el mapa autoorganizado. De esta forma se ajustarán los controles de motor de la neurona ganadora y de sus vecinas en proporción a la diferencia entre sus pesos de salida y el control de motor *c* .

Es decir durante la Autoorganización en lugar de asignar el control de motor producido a la neurona ganadora como lo hace el método directo u obtener una matriz de parámetros de control de motor como lo hace el método indirecto, se computará el peso de salida de las neuronas como se muestra a continuación, donde *N(s)* indica el vecindario de la neurona ganadora *s*.

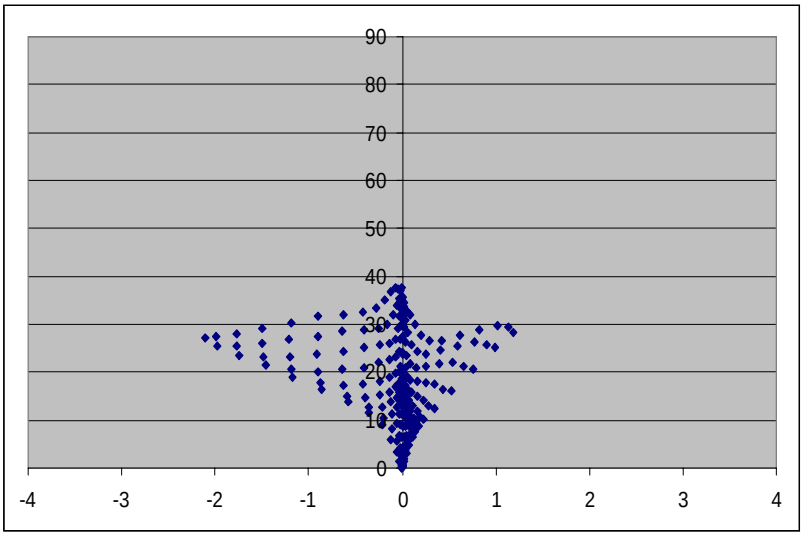
$$\Delta C_i = (C - C_i) * G(i,s) * \eta \qquad \forall i \in N(s)$$

$$C_i = C_i + \Delta C_i$$

Entrenamiento con Exploración

Al entrenar con el método propuesto, se comprueba que especificando el parámetro η con valores pequeños y campanas de gauss angostas no se consigue un buen ordenamiento de los controles de motor de las neuronas, si en lugar de dichos valores se utilizan valores mayores de η y campanas de gauss más anchas la red colapsa, esto se debe al alto grado de corrección que aplica al algoritmo sobre los controles de motor, provocando la perdida de la diversidad de controles de motor iniciales, y por ende no se pueden conseguir desplazamientos variados, por consiguiente luego de unas pocos pasos de entrenamiento los pesos de la red se aglutinan alrededor de un único centro.

En el siguiente gráfico se puede apreciar la distribución de las neuronas tras haber colapsado.



Para solucionar este problema en el presente trabajo se propone utilizar la técnica conocida como *Exploración*. Este método es utilizado para que un robot visite una mayor cantidad de pares situación/acción, que puedan no haber sido aprendidas si se tiene en cuenta solo los movimientos producidos durante el entrenamiento. Para lograr dicha exploración durante el entrenamiento on-line se producen cada cierta cantidad de pasos controles de motor aleatorios de forma de que la red pueda aprender los desplazamientos que estos producen.

Experimentos y Análisis

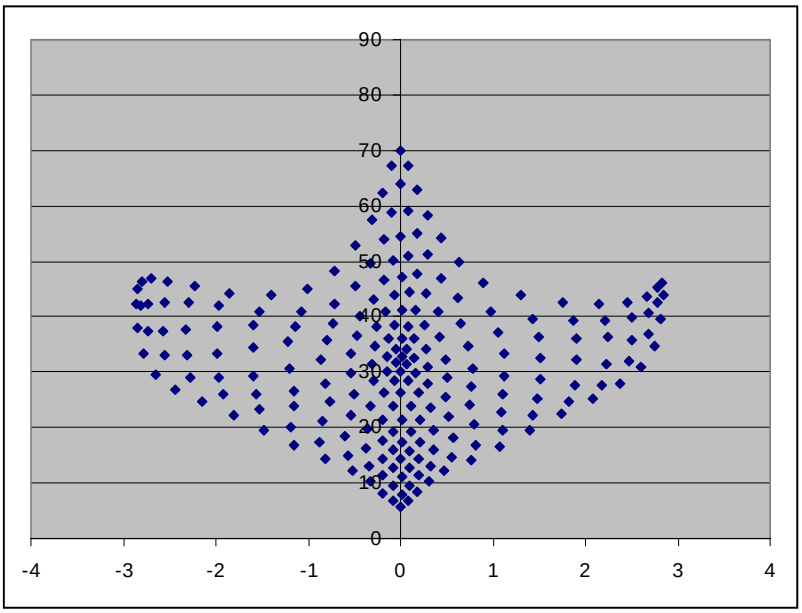
Esta sección presenta una evaluación de las distintas alternativas analizadas, utilizando un entrenamiento on-line sobre el movimiento de un robot que se mueve a un objetivo determinado en un ambiente libre de obstáculos.

Para la realización de los experimentos se utilizo un simulador de un robot *khepera* desarrollado por Johan Carlsson (http://r2d2.ida.his.se/). El robot utilizado por medio de este simulador posee una velocidad máxima en sus ruedas de 10 unidades (rango desde –10 hasta 10), siguiendo esta velocidad puede moverse en el entorno establecido a 76 unidades por segundo.

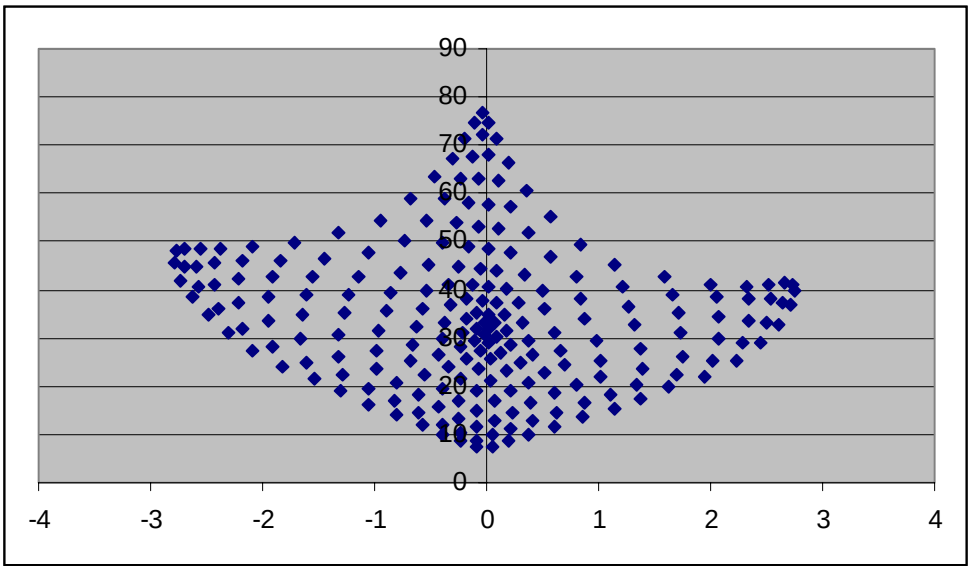
Para la realización de los experimentos se le presentan al robot objetivos hacia los cuales debe moverse, esto se prolonga durante una determinada cantidad de paso, entrenando a la red durante los mismos; cada paso de entrenamiento se produce una vez cada dos segundos. Luego del entrenamiento en forma on-line, se testea el movimiento obtenido por el robot tras haber aprendido sus desplazamientos, en este intervalo no se entrena a la red. Se imprimen también la distribución de las neuronas resultante del aprendizaje.

El mapa autoorganizado utilizado es de tamaño 15x15 neuronas, la red se inicializa con pesos ubicados en el rango de ángulo $[-\pi,\pi]$ y distancia cero, los parámetros de η iniciales utilizados para el entrenamiento de los pesos y de los controles de motor es de 0.1 y los finales de 0.0001, el vecindario de actualización inicial es de distancia 30 y la final es de 0, la desviación de la función gauss utilizada es proporcional a dicha distancia.

En primer lugar se entreno a la red por medio del método directo, a continuación se muestra la distribución de las neuronas resultante del entrenamiento.



Luego se entreno a la red con el método propuesto en este trabajo, consiguiendo una distribución de las neuronas según se detalla en el siguiente gráfico.



Como se puede observar ambos gráficos presentan características similares, esto se debe a que el robot obtiene en cada paso que realiza desplazamientos muy semejantes en ambos métodos, produciendo entonces mapas autoorganizados bastante similares.

Para poder observar el resultado de la ejecución de los controles de motor obtenidos según los métodos estudiados, en los siguientes gráficos se observan las trayectorias producidas por los mismos. Los gráficos fueron obtenidos tras haber indicado al robot la posición del objetivo, para lo cual el robot se mueve hacia él en base a los controles de motor y desplazamientos aprendidos, una vez que el robot llega al mismo se le indica un nuevo objetivo y así sucesivamente luego de una cantidad predefinida de pasos, se obtienen estos gráficos.

Gráfico utilizando el método Directo.

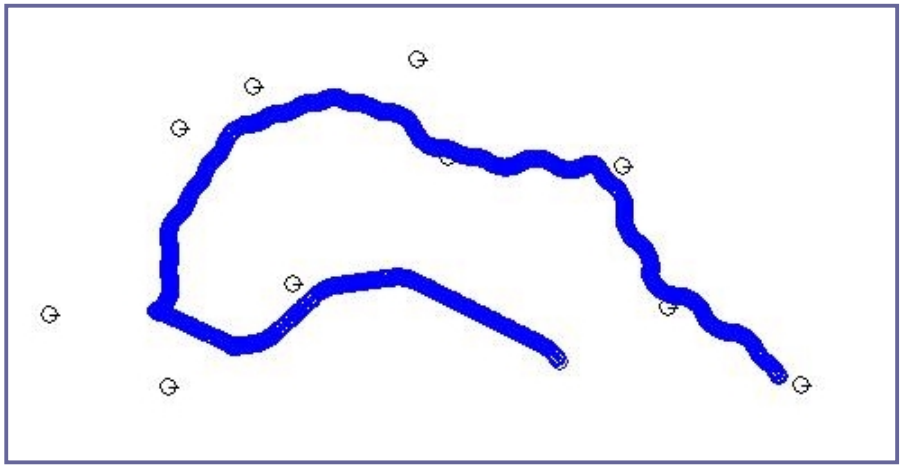
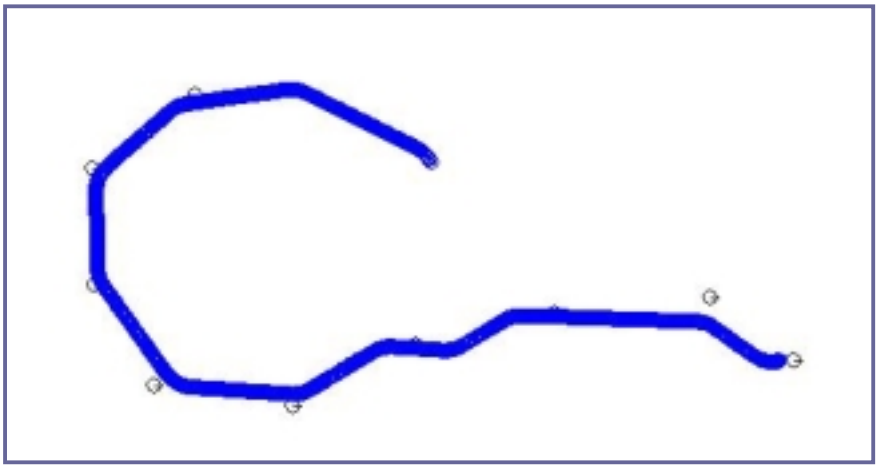


Gráfico método utilizando el promedio ponderado.



En los gráficos se puede apreciar la diferencia entre ambos métodos en cuanto a la suavidad de las trayectorias, tanto en los giros producidos como en su avance hacia el objetivo, notándose de esta forma la mejora que se ha logrado con el método propuesto.

Es preciso mencionar que el tiempo que transcurre entre cada paso de robot es un factor muy importante en el entrenamiento, pues los desplazamientos que consigue el robot serán mas variados a mayor tiempo. En tiempos cortos solo se pueden obtener desplazamientos de ángulos muy cercanos a cero o π, esto se debe a que en el preciso instante de ejecución del control de motor el robot toma movimientos exactamente hacia adelante o hacia atrás, pues comienza a moverse según su orientación, teniendo presente que el desplazamiento es corto entonces solo puede desplazarse en dichos valores de ángulo. Al no poder realizar ciertos desplazamientos estos no van a poder ser aprendidos y por consiguiente tampoco podrán ser utilizados por el robot, provocando que sus trayectorias no sean adecuadas.

Otro factor afectado por la utilización de tiempos cortos entre cada paso del robot es que en el avance hacia el objetivo solo son seleccionadas como neuronas ganadoras aquellas que se encuentran en el limite superior del mapa autoorganizado, según el gráfico, es decir aquellas que se corresponden con controles de motor que producen mayor velocidad, por lo cual aún cuando el robot se encuentre cerca del objetivo se siguen activando dichas neuronas, resultando que el robot no disminuya su velocidad a medida que se acerca al objetivo. Esto sucede porque la distancia máxima de los desplazamientos obtenidos esta restringida a la longitud del recorrido del robot en el tiempo especificado, es decir a menor lapso de tiempo, menor distancia de recorrido, pudiendo inclusive no superar esta distancia el diámetro del robot utilizado, y por consiguiente solo podría activarse una neurona interna si el objetivo se encontrase dentro del espacio que ocupa el robot.

Utilizar valores de tiempo grandes también acarrea problemas, ya que dado un desplazamiento, si el tiempo es elevado, para llegar desde la posición inicial a la final existen gran cantidad de trayectorias con distintos valores de control de motor.

Como se puede apreciar el método que se ha propuesto introduciendo los valores apropiados de tiempo y los parámetros correctos para el entrenamiento produce un movimiento de robot más suave y preciso que los otros métodos mencionados.

Conclusiones

En este trabajo se han mencionado distintas alternativas utilizando mapas autoorganizados de Kohonen para su aplicación en robot móviles que se mueven hacia un objetivo determinado. Existen diversos enfoques para resolver este problema (Ej: Aprendizaje por refuerzo), cada uno de los cuales presenta sus ventajas y desventajas, los mapas autoorganizados de Kohonen y su extensión (EKM's) resultan una opción seria para resolver el problema, que conjugándolo con el método propuesto por Kian Hsiang Low logra obtener un aprendizaje on-line.

Las ideas de Low son un aporte interesante a esta problemática, el método que él propone consiste en un ensamble de módulos de EKMs de alcance de objetivo y de obstáculos que cooperan entre si para determinar el camino a seguir de forma de alcanzar un objetivo. Se pudo constatar que esta propuesta presenta algunas dificultades, como lo es la forma en la cual se calcula los desplazamientos producidos por el robot y más importante aún el problema de discontinuidad que existe al utilizar las matrices de control de motor, por tal motivo la propuesta presentada en el presente trabajo logra hacer uso de estas ideas corrigiendo dichos inconvenientes.

Al utilizar mapas autoorganizados existen ciertos parámetros, como los son la velocidad de aprendizaje η y la desviación de la función de vecindario (función de gauss) los cuales son difíciles de determinar, de hecho existe gran cantidad de trabajos realizados sobre el tema. Este problema se complica aun mas debido a las características del problema, como son, no contar con un conjunto de entrenamiento independiente, pues los desplazamientos a aprender se obtienen a su vez de la misma red de Kohonen. Se quisiera que el robot continúe aprendiendo mientras dure su actividad, para lo cual habría que dejar un valor fijo para el parámetro η, hacerlo puede producir el ordenamiento de la red durante un tiempo, pero por ejemplo al aprender un desplazamiento no conveniente para el movimiento del robot, puede desaprender los desplazamientos anteriores. Este problema ocasiona que los parámetros utilizados para el aprendizaje sean variables, graduándolos a medida que transcurre el tiempo, lo cual también como se ha mencionado, puede provocar que la red no produzca los resultados deseados, por este motivo la utilización del método de exploración resulta conveniente y es determinante para producir resultados satisfactorios.

Otro parámetro importante que debe tenerse en cuenta es el tiempo que transcurre entre cada paso de robot, el valor asignado a el mismo hace variar gran parte de los parámetros utilizados en el ensamble de EKMs. El tipo de robot utilizado también es un factor que impacta de igual forma en los parámetros que deben ser seleccionados.

Si bien la propuesta de Low es interesante respecto de cómo obtener un movimiento suave y preciso, la misma, presenta una discontinuidad de los valores de controles de motor entre neuronas, este problema ha sido resuelto en este trabajo, realizando una modificación al método directo que toma en cuenta un promedio ponderado entre los controles de motor de neuronas cercanas, provocando de esta forma una solución efectiva a el problema, haciendo que el robot puede desplazarse hacia el objetivo en forma suave, correcta y precisa.

Trabajo Futuro

Esta sección revela las limitaciones del presente trabajo, las posibles mejoras o aplicaciones del método propuesto.

Como se ha mencionado reiteradamente, la selección de los parámetros de entrenamiento correctos es un trabajo complejo, para lo cual una posible mejora del método consistiría en aplicar algún algoritmo para la selección automática de estos parámetros. La elección del camino que realiza el robot esta determinada por un conjunto de funciones de gauss, que codifican en forma rígida la trayectoria a seguir, por lo cual seria interesante encontrar alguna alternativa basada en la misma idea pero que sea flexible.

En el presente trabajo se ha mencionado la utilización de módulos de evasión de obstáculos y como ellos interrelacionan con el método propuesto, la evaluación e investigación de la utilización de estos no ha sido verificada, seria por consiguiente de gran importancia agregar esta funcionalidad para ser utilizada en el movimiento de un robot que avanza hacia un objetivo en entornos complejos e impredecibles. El agregar esta funcionalidad traje aparejado la complejidad de la utilización de los sensores que debería poseer un robot para utilizar estos módulos.

Una extensión de este método es agregar la noción de movimiento al objetivo que debe alcanzar el robot, a fin de poder llegar al objetivo evitando obstáculos y realizando movimientos suaves y precisos. Este aporte sería de utilidad por ejemplo en las aplicaciones de Fútbol de Robots y exploración de entornos desconocidos con el fin de seguir a un objetivo, podría incluso aplicarse en controles de automotores para el manejo automático del mismo con el fin de seguir a otros vehículos.

El robot al dirigirse al objetivo, en ciertas circunstancias, utiliza controles de motor que no toman en cuenta la noción de cercanía para bajar la velocidad acorde a la distancia.

Queda pendiente la verificación de la implementación del presente trabajo sobre robots reales, en los cuales se presentaran otro tipo de problemáticas relacionadas con fallas mecánicas, ruido en los sensores, etc.. , o la aplicación sobre robots que no posean dos ruedas, por ejemplo arañas, vehículos motorizados para terrenos complejos.

Como se ha analizado, el tiempo entre cada paso de robot es un factor relevante en el entrenamiento, se podría verificar si la utilización de tiempos grandes para el aprendizaje y tiempos mas cortos en la ejecución producen mejores resultados, dado que con tiempos mas cortos entre cada paso de robot se producirían mayor cantidad de controles de motor provocando trayectorias mas precisas.

Bibliografía

[L04]Kian Hsiang Low, Wee Khleng, Marcelo H. Ang, Jr. *An Ensemble of Cooperative Extended Kohonen Maps for Complex Robot Motion Tasks*. MsC Tesis, 2004.

Kian Hsiang Low, Wee Khleng, Marcelo H. Ang, Jr. *Continuos-Spacecd Action Selection fro Single and Multi-Robot Tasks Using Cooperative Extended Kohonen Maps*. IEEE International Conference on Networking, Sensing and Control (ICNSC'04), p. 198-203.

Kian Hsiang Low, Wee Khleng, Marcelo H. Ang, Jr. *Action Selection for Single and Multi-Robot Tasks Using Cooperative Extended Kohonen Maps*. 18th International Joint Conference on Artificial Intelligence (IJCAI-03), 2003, p. 1505-1506.

K. H. Low, W. K. Leow, and M. H. Ang, Jr. *Action selection in continuous state and action spaces by cooperation and competition of extended Kohonen maps.* Int. Joint Conf. on Autonomous Agent & Multi Agent Systems, 2003, p. 1056-1057.

[VG95] C. Versino, L.M. Gambardella. *Learning the visuomotor coordination of a mobile robot by using the invertible kohonen map*. Proceedings of IWANN95, International Workhop on Neural Networks, Malaga, Spain, June 1995.

C. Versino, L.M. Gambardella. *Learning Fine Motion in Robotics: Desing and Experimets*. Recent Advances in Artificial Neural Networks, Desing and Application, CRC press, pp. 127-153, 2000.

[O04] Elena Lazkano Ortega. *Pautas para el desarrollo incremental de una arquitectura de control basada en el comportamiento para la navegación de robots en entornos semi-estructurados*. Universidad del Pais Vasco. Tesis de Doctorado. Sep. 2004, pag. 18-25.