10º Congreso Nacional de Mecatrónica
Noviembre 3 y 4, 2011. Puerto Vallarta, Jalisco.

Desarrollo de un Simulador para Prototipo de Vehículo Evasor de Obstáculos Usando Lógica Borrosa

Madrid Amarillas Germán[1], Luna Acosta Noé[2], Jiménez López Eusebio [3], Reyes Ávila Luis[4], Eduardo Núñez Perez[5] y Fernando Orduña Cabrera[6]

[1,2] Centro de Investigación y Aplicación en Automatización y Mecatrónica de la Universidad Tecnológica del Sur de Sonora, otrogama@hotmail.com, nluna@uts.edu.mx
[3,5]CINNTRA UTS-IIMM-ULSA Noroeste, ejimenezl@msn.com
[4]IMT-IIMM.
[1,5]Universidad La Salle Noroeste.
[6]Instituto Tecnológico Superior de Cajeme

Resumen

La Mecatrónica se auxilia de otros campos del conocimiento, como es el caso de la Inteligencia Artificial (AI), para cumplir objetivos didácticos o industriales. Para motivar la enseñanza de la mecatrónica, es necesario generar prototipos o simuladores de sistemas que pueden ser controlados usando algoritmos de la Inteligencia artificial, como es el caso de la lógica borrosa. En la construcción de prototipos se requiere implementar una fase orientada a la programación y simulación de la realidad que se quiere trabajar. La simulación computacional es una etapa fundamental en el desarrollo de máquinas y sistemas mecatrónicos. En este artículo, se desarrolla una simulación de un dispositivo evasor de obstáculos usando lógica borrosa. El modelo matemático, así como el algoritmo inteligente desarrollado, fue implementado en un programa codificado en el lenguaje de programación DEV C++ mediante las bibliotecas gráficas "Winbgim". Dicho programa simula un sistema constituido por cinco sensores como datos de entrada y dos motores como datos de salida y un sistema de control basado en lógica borrosa de tipo mamdani.

Palabras clave: Inteligencia Artificial, Lógica Borrosa, Evasión de Obstáculos.

1 Introducción

La Mecatrónica se auxilia de otros campos secundarios del conocimiento, como es el caso de la Inteligencia Artificial, para lograr objetivos educativos, esto es la transferencia de conocimientos y el desarrollo de prototipos didácticos. Por otro lado, para motivar a los alumnos de ingeniería al estudio sistemático de la Mecatrónica y de sus áreas principales (la Mecánica, la Electrónica y la Computación [1]) y secundarias, es necesario generar prototipos por medio de los cuales sea posible probar teorías, técnicas o métodos. Los prototipos son recursos didácticos que atraen y motivan no solo al desarrollo de productos si no que también le dan sentido aplicativo a campos de conocimientos secundarios como las Matemáticas y la Lógica.

En el caso de la Mecatrónica, al ser una ingeniería especializada, se requiere que todo prototipo, aunque sea didáctico, debe pasar por un proceso de análisis, modelación, simulación y construcción. Por ejemplo, el diseño de un prototipo mecatrónico debe pasar primero por una etapa de simulación computacional antes de ser construido en la realidad. La simulación computacional, de acuerdo con [2], es una herramienta poderosa que utilizan las industrias y los centros de estudio para el diseño y la modificación de los sistemas productivos y para el desarrollo de prototipos didácticos. Simular es sinónimo de imitar el comportamiento de un sistema con algún propósito específico. Para el diseño de sistemas mecatrónicos la simulación computacional juega un papel muy importante, pues permite imitar y probar los comportamientos

de los sistemas y subsistemas antes de ser implementados en la realidad.

Por otro lado, una de las áreas de la Ingeniería que auxilia a la mecatrónica en el desarrollo de prototipos didácticos especializados, es la Inteligencia Artificial. Esta área proporciona teorías, métodos y algoritmos que hacen posible el diseño, la simulación y en control de máquinas mecatrónicas inteligentes. La Inteligencia Artificial es cada vez más común en los productos comercializados actualmente. Industrias como la automovilística, teléfonos celulares, manufactureras de alto volumen de producción, videojuegos, de alimentos, etc. se han inclinado por estas nuevas tecnologías por su alta eficiencia en trabajos complejos y rápidos. Sin embargo, a nivel didáctico, la Inteligencia Artificial, de acuerdo con [3], tiene algunos problemas como: la información es un poco confusa y carece de la difusión adecuada, sobre todo a nivel de la enseñanza de la ingeniería. Una amplia porción del material didáctico disponible trata el tema más a nivel técnico que al nivel ingenieril, haciendo de la inteligencia artificial una herramienta que hace más caso a la forma que al fondo. Cuando este no es el caso, el material tiende a ser elevadamente teórico, dificultando así que el estudiante aterrice dichos conocimientos. El ingeniero que desee implementar una solución donde se aplique la inteligencia artificial debe de ser capaz de desarrollar a la medida del problema, siguiendo una metodología que le permita hacer esto sin correr el riesgo de lograr un resultado pobre, tardado o inclusive obsoleto. Además, algunas de las principales razones por las que se dificulta el aprendizaje en estos campos del conocimiento son, por ejemplo, la poca o nula relación que el alumno encuentra entre estas materias y el campo laboral en el cual buscan desarrollarse, así como la apariencia tan abstracta con las que se presentan dichas materias en los salones de clase.

En este contexto, es necesario desarrollar proyectos en donde los alumnos puedan aplicar la mayoría de los conocimientos que aprenden en clases, en particular, conocimientos de la Inteligencia artificial, y con ello reforzar su interés y su motivación en aplicaciones tecnológicas. En un trabajo reciente se desarrolló un prototipo físico de un vehículo evasor de obstáculos [3] el cual fue controlado usando lógica borrosa [4,5]. Para el desarrollo de dicho prototipo no se implementó una etapa de simulación. Para continuar con el desarrollo completo del prototipo es necesario construir un simulador. En este artículo se desarrolla un simulador que imita el comportamiento *de un vehículo evasor de* obstáculos. El simulador es hecho en el lenguaje C++, usando el software libre DEV C++, mediante las bibliotecas gráficas “Winbgim” y es utilizada la lógica borrosa para generar un sistema de control para el vehículo.

2. Desarrollo

En esta sección se presenta el desarrollo del simulador. En primer lugar se presentan algunas consideraciones sobre el dispositivo real desarrollado en [3], y posteriormente se describe el modelo del simulador.

2.1 Vehículo evasor de obstáculos.

La figura 1 muestra la configuración del dispositivo evasor de obstáculos, el cual consta de dos motores y 5 sensores localizados en posiciones estrategias, según se muestra en la figura 1.

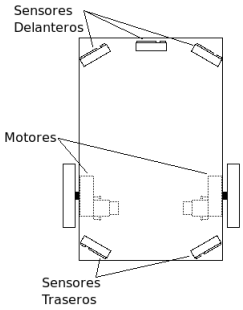

Fig. 1. Configuración física de las entradas y salidas del dispositivo

El dispositivo fue desarrollado en forma física y fue controlado con un algoritmo de lógica borrosa. El método usado para construir el modelo difuso fue el de Mamdami. Para controlar los movimientos, se diseñó un circuito electrónico usado dos microprocesadores [3]. Cabe mencionar que el desarrollo del dispositivo no incluyó el diseño de un simulador computacional. Tomando la experiencia real del prototipo, en este artículo se describe el desarrollo de un simulador computacional de dicho dispositivo.

2.2 Composición del sistema para el desarrollo del simulador.

La composición del sistema de divide, en primer lugar, en los sistemas de coordenadas, los cuales darán referencia a cada uno de los movimientos del dispositivo y, en segundo lugar,

las variables del sistema, las cuales son los datos manipulables a lo largo del algoritmo.

2.2.1 Sistema de coordenadas

Para el desarrollo del sistema en cuestión, es necesario tomar en cuenta dos sistemas de coordenadas, las cuales se describen a continuación. En primer lugar se tiene el sistema de coordenadas principal, el cual es utilizado por las bibliotecas gráficas "Winbgim". Estas bibliotecas consideran la esquina superior izquierda de la ventana generada en el programa como el origen (coordenadas (0, 0)) tomando en cuenta el eje trazado a partir del mismo hacia la derecha como el de las abscisas o eje 'X' positivo y el eje trazado desde el origen hacia abajo como el de las ordenadas o eje 'Y' positivo. En segundo lugar se tiene el sistema de coordenadas **objeto**. Este es un sistema de referencia auxiliar diseñado por el programador para localizar de una forma más natural las coordenadas del dispositivo. En nuestro caso, este sistema de coordenadas tiene su origen en un punto de la recta imaginaria que pasa por el centro de los dos motores. En la figura 2 se muestran los dos sistemas de coordenadas juntos, donde el dibujado en color amarillo, representa el sistema de coordenadas objeto.

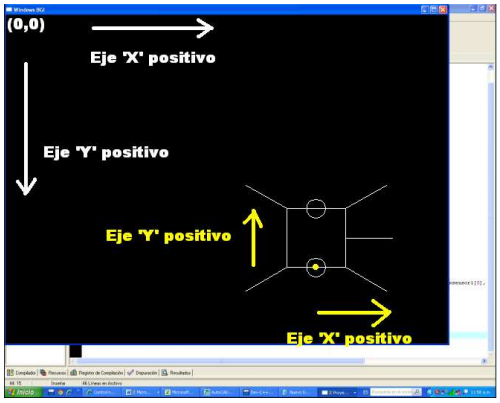

Fig. 2. Sistemas de coordenadas

2.2.2 Variables del sistema

Las variables del sistema se utilizan ya sea para obtener información del sistema o para modificar el estado del mismo. La composición gráfica principal del dispositivo se muestra en la figura 3.

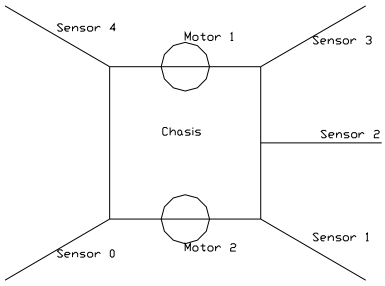

Fig. 3. Composición gráfica principal del dispositivo

De acuerdo con la figura 3, el dispositivo está constituido por cinco sensores, dos motores y un chasis. En el caso de los sensores, estos simulan la distancia a la que empezarán a detectar la presencia de obstáculos, misma presencia que será representada por una distancia en pixeles. El movimiento de los dos motores mostrados en la figura anterior dará pie al comportamiento cinemático del dispositivo. El chasis representa el cuerpo del dispositivo, el cual contendrá a los sensores y motores. Por otro lado, para lograr la simulación del movimiento del dispositivo, es necesaria la manipulación de los puntos que describen al mismo, para de esta manera generar las líneas necesarias para dibujarlo. En nuestro caso, son necesarios un total de 12 puntos, los cuales son la base del dibujo completo (ver figura 4).

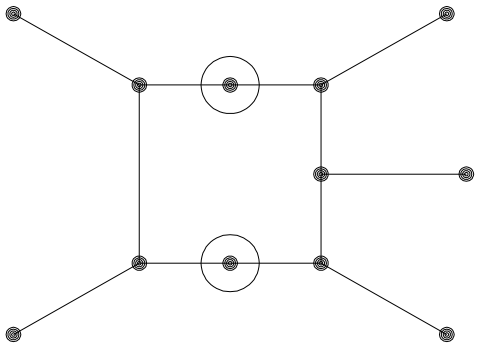

Fig. 4. Puntos base del dispositivo

2.2.3. Centro de giro

Para lograr la animación del dispositivo, fue necesario generalizar el movimiento, de tal manera que no existierán movimientos lineales, sino puros giros, de tal manera que cuando el dispositivo parece describir un movimiento rectilíneo, realmente describe el perímetro de un círculo grande. Para describir dicha trayectoria, es necesaria la localización del llamado "centro de giro", el cual es el punto de referencia sobre el

cual cada uno de los puntos del dispositivo girará. El centro de giro depende de la velocidad de cada motor. En la figura 5 se muestra un ejemplo de centro de giro aplicado a cinco de los doce puntos mencionados, generado a partir de las velocidades 75% y 50% de los motores 1 y 2 respectivamente.

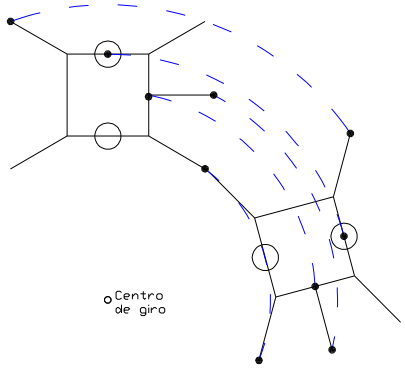

Fig. 5. Centro de giro

2.2.4. Vectorización

Para lograr la simulación descrita en la figura 5, es necesario vectorizar cada uno de los puntos del dispositivo, pudiendo de esta manera girar el vector con referencia al centro de giro. En la figura 6 se muestra la vectorización de los cinco puntos descritos en la imagen mencionada.

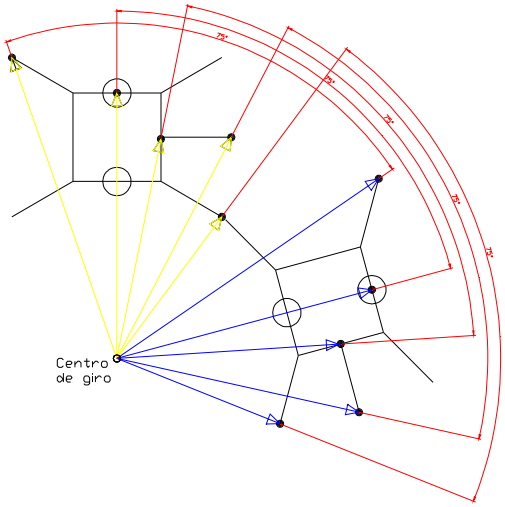

Fig. 6. Vectorización

2.2.5. Captación de los sensores

En el dispositivo se representa el alcance de cinco sensores instalados alrededor del mismo, los cuales captarán la distancia en pixeles del próximo obstáculo, misma que podrá ser usada por el programador para el comportamiento del sistema. Al igual que un vector, cada sensor tiene una dirección y una magnitud. La dirección representa simplemente la orientación física de cada sensor respecto al dispositivo. La magnitud indica el alcance máximo de cada sensor en dicha orientación.

2.2.5. Velocidad de los motores

La velocidad de los motores será representada por dos variables, en las cuales, el valor obtenido indicará la velocidad lineal de cada motor en pixeles/ms.

2.3 Desarrollo del Algoritmo

El algoritmo general del sistema se divide en ocho fases, como se muestra en la siguiente figura 7.

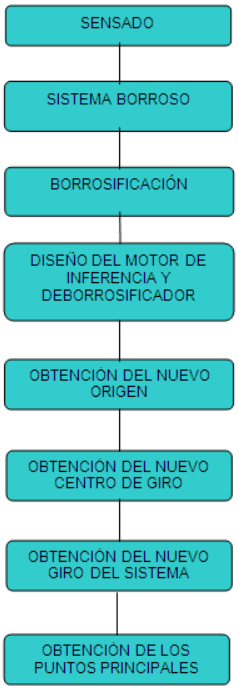

Fig. 7. Algoritmo general del sistema

Cabe mencionar que por razones de espacio no se describirán todas las fases, por lo que se expondrá un resumen de las dos primeras. Esto es:

Etapa de sensado

Para obtener la distancia en pixeles del objeto más próximo al dispositivo, fue necesario diseñar la función "sensado". Dicha función modifica los valores del arreglo "distancia", el cual contienen cinco variables enteras, las cuales a su vez representan la distancia a la que actualmente está el próximo obstáculo de cada sensor. Cabe mencionar que la distancia dependerá de la posición del sensor en el dispositivo, de la orientación del mismo y de la aproximación del obstáculo más cercano. La función "sensado" se auxilia de las funciones "getpixel" de la biblioteca Winbgim y "coor" declarada internamente en el programa. En el caso de la función getpixel, sus valores de entrada son las coordenadas de un pixel en pantalla y da como resultado el valor correspondiente al color del mismo. En el caso de la función coor, recibe como valor de entrada las coordenadas de un vector de posición y el tipo de coordenada que se desea obtener (ya sea en 'x' o en 'y') y obtiene como salida la coordenada correspondiente.

Etapa de sistema borroso

El sistema borroso es el sistema de control encargado de recibir información a través de sensores, interpretarla y tomar una decisión que se traduce en movimiento. El sistema difuso utilizado cuenta con cinco entradas y dos salidas, las cuales representan a los sensores y motores respectivamente como se muestra en la figura 8.

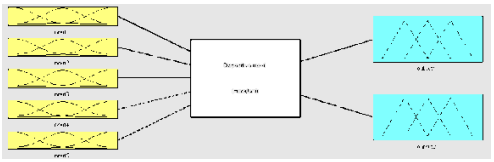

Fig. 8. Sistema borroso

Cada uno de los cinco sensores y dos motores colocados en el dispositivo tiene características diferentes por su posición. A cada una de las variables utilizadas se le asigna una función de membrecía que a su vez contiene determinada cantidad de funciones de transferencia. En el caso del sistema borroso utilizado, a las salidas (motores) se le asignaron funciones de membrecía idénticas (ver figura 9.), mientras que las entradas (sensores) obtuvieron dos funciones de membrecía diferentes (ver Fig. 9 y 10) a causa de las características posicionales de los sensores (delanteros y traseros).

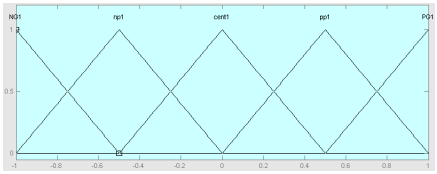

Fig. 8. Función de membrecía del motor

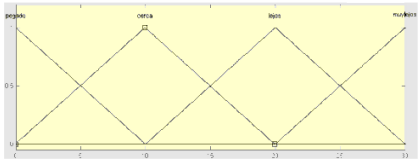

Fig. 9. Función de membrecía de un sensor delantero

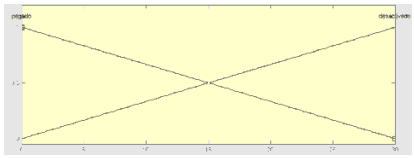

Fig. 10. Función de membrecía de un sensor trasero

Cabe mencionar que todo el proceso de desarrollo del algoritmo puede revisarse en [3].

2.4 Representación del Simulador

Una vez desarrollado y programado el algoritmo se procedió a visualizar los movimientos del dispositivo mediante una interfaz gráfica desarrollada en el lenguaje de programación DEV C++. Las figuras 11 y 12 muestran el dispositivo en un recorrido por el ambiente virtual.

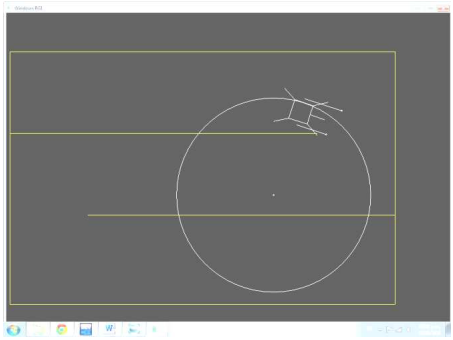

Fig. 11. Salida gráfica del simulador

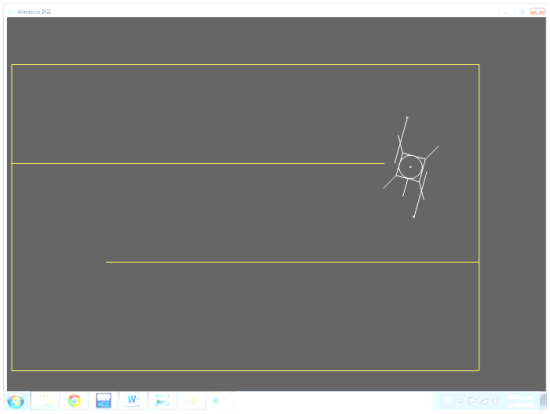

Fig. 12. Dispositivo recorriendo la ruta de trabajo

Finalmente, cabe mencionar que existe la posibilidad de agregar accesorios para comprender más lo que sucede durante la simulación. En este caso, están implementados tres accesorios como se muestran en las figuras anteriores.

1) Trayectoria: muestra la trayectoria del actuador más lejano al centro de giro.
2) Centro de giro: muestra el centro de giro alrededor del cual se describe la trayectoria actual.
3) Motores comparativos: mediante dos líneas, una para cada motor, se muestra gráficamente la velocidad de cada motor.

4. Conclusiones

En este artículo se ha descrito el desarrollo de un simulador de un dispositivo evasor de obstáculos usando lógica borrosa. Las conclusiones derivadas de este trabajo se resumen a continuación:

- Se mostró la aplicación de dos áreas principales de la ingeniería moderna como son las Matemáticas y la Inteligencia artificial para el desarrollo de un simulador cinemático de un dispositivo inteligente destinado a reaccionar ante la aparición de obstáculos virtuales.

- El simulador desarrollado puede ser utilizado para la generación de prototipos físicos, obteniendo de esta manera conclusiones exactas y manipulando una infinidad de variables en poco tiempo y con solo el acceso a una computadora, de maneras que físicamente llevaría tiempo y esfuerzo en demasía.

- Los modelos matemáticos muestran una aplicación fácil de comprender y sobre todo con una demostración visible como lo es el simulador mismo.

- El algoritmo basado en lógica borrosa de tipo mamdani [3] es un claro ejemplo de aplicación de la IA, mismo que al combinarlo con el simulador, dan una demostración del alcance del método.

Referencias

[1] Bishop, Robet H. The Mechatronic Handbook. Crc Press Washington D.C. 2002.

[2] JIMÉNEZ E.: Simulación de un proceso de manufactura con obstáculo en la línea de producción. *México. (1998).* (Tesis de Maestría en Ingeniería Mecánica). UNAM, Facultad de Ingeniería, División de Estudios de Postgrado, Sección Mecánica.

[3] Madrid G., JIMÉNEZ E., Reyes L., E., Luna Noé. Diseño e implementación del sistema de control de un dispositivo móvil evasor de obstáculos usando lógica borrosa. 9º Congreso Nacional de Mecatrónica, Octubre 13 al 15, 2010. Puebla, Puebla.

[4] Hao Ying, Yongsherng Ding, Shaokuan Li, Shihuang Shao. Typical Takagi-Sugeno and Mamdani fuzzy systems as universal approximators: necessary conditions and comparison.Fuzzy Systems Proceedings, 1998, IEEE World Congress in Computational Intelligence.

[5] MADRID G., BUENTELLO M. Diseño e implementación del sistema de control de un dispositivo móvil evasor de obstáculos usando lógica borrosa. (Tesis de mecatrónica de Licenciatura). Universidad La Salle Noroeste. Diciembre del 2008.