SISTEMA DE NAVEGACIÓN BASADO EN SENSOR DE VISIÓN DINÁMICA PARA UN ROBOT HEXÁPODO

De la cruz Sustaita Luis David (1), Rostro González Horacio (2), Pérez Trujillo Abraham (2), Espinal Jiménez Andrés (2), Sotelo Figueroa Marco Aurelio (2)

1 [Licenciatura en sistemas de información administrativa, Universidad de Guanajuato] | [ld.delacruzsustaita@ugto.mx]

2 [Departamento de electrónica, División de ingenierías, Irapuato-Salamanca, Universidad de Guanajuato] | [hrostrog@ugto.mx]

Resumen

El presente trabajo, propone un sistema de navegación para un robot hexápodo; el cual le permite desplazarse en ambientes tanto estáticos como dinámicos. El sistema está soportado por ePyNN, una tarjeta Raspberry Pi 3 con el simulador para Redes Neuronales Pulsantes PyNN embebido, el cual se encarga de dos tareas principalmente: la ejecución del sistema de locomoción basado en un Generador Centrales de Patrones y el análisis del ambiente mediante un Sensor de Visión Dinámica; ambos componentes inspirados en seres vivos. La propuesta fue puesta a prueba y validada en diferentes ambientes.

Abstract

The present research, proposes a navigation system for a hexapod robot; which permits to displace it in either static ambient or dynamic ones. The system is supported by ePyNN, a Raspberry Pi 3 board with embedded PyNN for simulating spiking Neural Networks, which mainly carries out two tasks: the execution of the Central Pattern Generator-based locomotion system and the analyze of the ambient by means of a Dynamic Vision Sensor; both are inspired in living being's characteristics. The proposal was tested and validated on the hexapod robot in different ambient.

Palabras Clave

Robot hexápodo; Sensor de visión dinámica; Raspberry Pi; ePyNN; Generador de patrones centrales.

INTRODUCCIÓN

La robótica es una rama de la inteligencia artificial que ha tenido cabida en ámbitos industriales, ingenieriles, de estudios teóricos, entre otros. Hoy en día, existen una gran variedad de tipos de robots, algunos con diseños bioinspirados como lo son aquellos con patas, aletas o alas, etc. [1]. Actualmente, los sistemas de locomoción de los robots han sido basados en métodos con diferentes fundamentos como modelos matemáticos, leyes heurísticas, circuitos neuronales, etc. [2].

Recientemente, los sistemas de locomoción bioinspirados, circuitos neuronales conocidos como Generadores centrales de patrones (GCPs), han sido exitosamente usados para la locomoción de robots bípedos [3], cuadrúpedos [4], hexápodos [5,6,7,8], entre otros con diseños inspirados en seres vivos debido a diferentes propiedades que proporcionan estos [1]. Sin embargo, para lograr que los robots naveguen en diversos ambientes, es necesario complementar la acción de los GCPs con información adicional, ya que en general, los GCPs son modulados o intercambiados por información sensorial externa [9].

En [8], Pérez et. al. Propusieron una plataforma de bajo costo que permite ejecutar la librería PyNN para la simulación de redes neuronales pulsantes; como aplicación de su propuesta, dotaron de un sistema de locomoción basado en GCPs a un robot hexápodo. El presente trabajo, tiene como objetivo retomar el trabajo realizado en [8] y complementarlo para crear un sistema de navegación basado en visión mediante un Sensor de Visión Dinámica (DVS, por sus siglas en inglés), el cual permita al robot hexápodo evitar objetos estáticos o ambulantes en el ambiente que se encuentre.

MATERIALES Y MÉTODOS

ePyNN

En [8], se propone una plataforma de bajo costo para simular Redes Neuronales Pulsantes. Los componentes se describen a continuación:

- *PyNN:* es una librería desarrollada en Python, para construir modelos de redes neuronales que pueden ser probados en diferentes simuladores (Brian, NEST, NEURON) o incluso en sistemas de hardware neuromórficos como el *SpiNNeaker* y BrainScales sin tener que adaptarlo a cada entorno diferente [8]. En este caso, se usó el simulador Brian, pero puede ser usado cualquier otro de los que soporta PyNN.
- *Raspberry Pi 3 modelo B:* es un modelo que incluye las siguientes características: 1 GB de memoria RAM, 4 puertos USB, procesador Quad Core de 1.2GHz Broadcom BCM2837 de 64bit, conexión wifi y bluetooth, entrada micro-USB de 2.5A. Es un ordenador de tamaño y características reducidas (aproximadamente el de una tarjeta de crédito).

Generador Central de Patrones

El Generador Central de Patrones (GCP), es una Red Neuronal Artificial que permite imitar comportamientos de circuitos neuronales biológicos; los cuales contribuyen a la locomoción mediante la generación periódica de patrones rítmicos [10]. Dichos patrones rítmicos son generados sin necesidad de entradas sensoriales externas. Los patrones rítmicos, sirven para controlar y coordinar actividades motoras repetitivas como el andar, masticar, o nadar, entre otros [6]. Para el robot hexápodo, se requiere de una Red con 12 neuronas; cada una controla un servomotor del robot. Los parámetros de la Red Neuronal fueron tomados de [8].

Sensor de Visión Dinámica

El Sensor de Visión Dinámica (DVS), a diferencia de las cámaras estándar que toman fotos del escenario completo con información probablemente irrelevante para la navegación del robot, permite procesar la información en cada pixel asíncronamente; cada pixel censa eventos de cambios de intensidad de luz en el tiempo. Dando como resultado una serie de eventos capturados con resolución de microsegundos, ya que solo se

envían eventos cuando se detecta un cambio, esto conlleva a sistemas que pueden ser procesar información a una velocidad mayor que cuando se utiliza otro tipo de cámaras convencionales [11]. La imagen 1, muestra un disco con un punto negro que gira, en la parte superior se muestra la salida de una cámara estándar tomando continuamente fotos con las diferentes posiciones del punto negro, mientras que, en la inferior, se muestra la salida del DVS, donde únicamente se ve la información del movimiento del punto negro sin capturar el escenario completo.

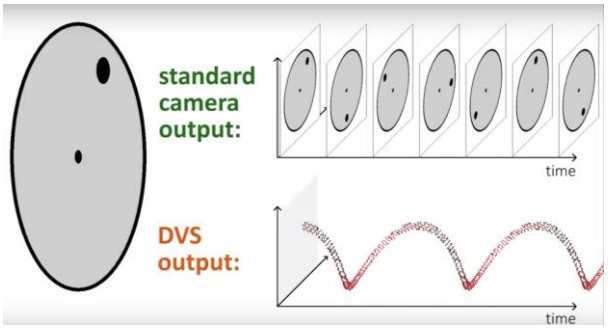

IMAGEN 1: Ejemplo de la salida de cámara convencional y salida de un DVS

Sistema de Navegación

El DVS resulta ser bastante útil para prevenir obstáculos sin realizar costosos procesamientos de la información censada. De acuerdo con los resultados obtenidos en experimentos realizados con un robot de ruedas su potencia se incrementa al incorporarlo con redes neuronales neuromórficos [12].

En la imagen 2 se puede observar una vista del robot hexápodo (izquierda) en la que se puede apreciar la Raspberry Pi y las 6 patas, también observamos una vista frontal donde se logra distinguir el DVS.

La Raspberry Pi 3 es la encargada de ejecutar el GCP con el que se generan los patrones del hexápodo, además, a esta se conecta al DVS, y se encargara de procesar el algoritmo de navegación basado en los eventos registrados.

El DVS produce una matriz, o ventana, de 128x128 pixeles, se conecta por USB a cualquier ordenador. Su consumo de energía y almacenamiento es bastante bajo, y por protocolo AER la transmisión de datos es prácticamente al instante. Para este trabajo, se desarrolló una interfaz en Python puesto es el lenguaje en el que se desarrolló PyNN y por consecuencia el GCP que le da movimiento a las patas del hexápodo.

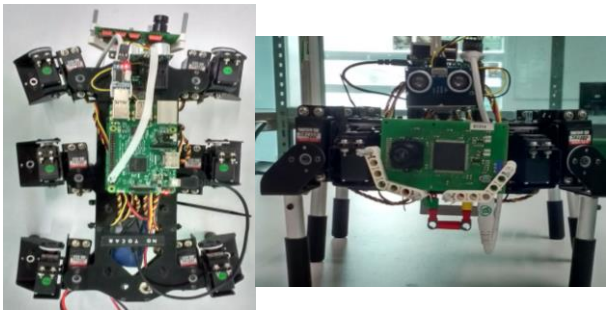

IMAGEN 2: DVS instalado sobre el robot hexápodo.

El sistema de navegación tiene la tarea de lograr el desplazamiento del robot (mediante el sistema de locomoción basado en GCP) y a su vez, la tarea de evitar obstáculos; ya que esta, es una de las más importantes para la seguridad tanto del robot como de los objetos y personas que lo rodean. Con el SVD se puede evitar obstáculos con gran rapidez dependiendo de la cantidad de eventos registrados en cierto tiempo y dependiendo de la región donde sucedan estos eventos. Para lograr ambas tareas, se propone la evasión de obstáculos estáticos y ambulantes haciendo que el robot hexápdo gire al momento de detectar eventos en una zona de riesgo asignada en su campo de visión.

Uno de los retos encontrados en la investigación fue reducir la cantidad de eventos ruidosos que se generan al momento de poner el sensor en movimiento. En un ambiente no controlado podemos encontrar gran cantidad de objetos a lo lejos que al no ser detectados al 100% por su lejanía, para el DVS, significan eventos de poca importancia, incluso cuando no existe actividad se llegan a detectar eventos; por esta razón se propuso diseñar y utilizar un filtro para reducir la cantidad de eventos ruidosos detectados basado en la diferencia de tiempo que existe entre dos

eventos consecutivos capturados en la misma posición. Existen otros filtros basados en vecindades de eventos y el tiempo [12] como lo es el *Background Activity Filter*, desarrollado en JAVA para el Adress-Event representation (AER).

El sistema de navegación propuesto cuenta con un filtro para reducir el ruido de fondo, el algoritmo implementado fue el siguiente:

Para cada evento:

1- Guardar el tiempo en el que fue capturado (tmap).

2- Si la diferencia del tiempo anterior y el tiempo actual sobre el mismo pixel es mayor a 0.8 seg. el evento no es tomado en cuenta, de lo contrario se toma como evento valido.

3- Actualizar tmap con el tiempo actual.

El método de detección de objetos propuestos toma en cuenta solo una región de la matriz proporcionada por el DVS, esta *región de riesgo* está situada en la parte baja central, que es donde existe mayor probabilidad de encontrar un objeto de peligro para el andar del hexápodo. (ver imagen 3).

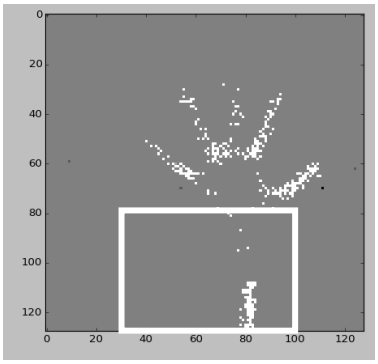

IMAGEN 3: Captura de pantalla de los eventos detectados cuando una mano pasa por el DVS. El recuadro blanco es la zona de riesgo.

Este método consta de los siguientes pasos:

1- Guardar el tiempo del primer evento registrado en la zona de riesgo.

2- Cada 500 eventos que estén dentro de la zona de riesgo reiniciar el contador.

3- Si la diferencia de tiempo entre el primer evento registrado y el ultimo (500) es menor a 0.5 segundos se manda una alerta de peligro, se guarda el tiempo en que sucedió la alerta y se le manda un patrón de giro al hexápodo, de otra forma no hacer nada para continuar con el andar.

4- Si se detectan más alertas en menos de 4 segundos no se toman en cuenta hasta después de los 4 segundos.

RESULTADOS Y DISCUSIÓN

En el método de detección de objetos, el último paso, fue propuesto debido a que, si se detecta una alerta y el hexápodo gira, 4 segundos es lo que tarda en terminar de realizar el giro. Si no se implementara ese paso se podrían mandar varias alertas consecutivas por aun tener el objeto de peligro en la zona de riesgo provocando que el robot haga los ajustes necesarios de acuerdo con el número de alertas detectadas.

El filtro propuesto logra eliminar considerablemente la cantidad de eventos ruidosos, esto también reduce la cantidad de eventos detectados en contrastes bajos. En la imagen 4 se muestra una gráfica en 3D de eventos capturados por el DVS.

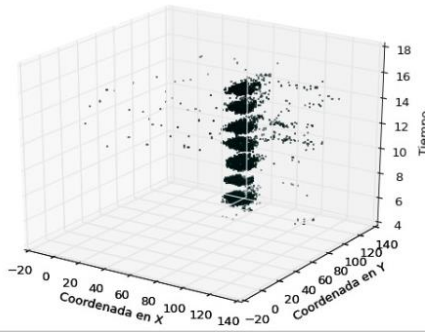

IMAGEN 4: Grafica en 3D de los eventos detectados por el DVS.

Debido al filtro, en un ambiente con obstáculos fijos, el algoritmo tarda en detectar algunos objetos de bajo contraste que no generan la cantidad de eventos necesarios para evitarlos oportunamente; además, obstáculos como paredes u objetos grandes con áreas lisas o sin cambios de color son

difíciles de detectar puesto que solo se logra identificar sus bordes. En un ambiente con obstáculos dinámicos es más fácil detectarlos como objetos de riesgo ya que al estar en movimiento generan más eventos.

El reto para las siguientes investigaciones será combinar el DVS con un sensor ultrasónico para resolver el problema de las paredes u objetos grandes, el sensor ultrasónico emite señales que miden la distancia que existe entre el sensor y algún objeto frente a él.

Se hizo un repositorio de videos en los que se puede apreciar el hexápodo caminando y evadiendo algunos obstáculos. Link de carpeta: https://www.dropbox.com/sh/8tcy8rx9y18gy90/AADi__FBAwUEpxrPSZHTv5ZNa?dl=0

CONCLUSIONES

En este trabajo se obtuvieron resultados aceptables aplicando el filtro de ruido para el DVS reduciendo considerablemente la cantidad de eventos basura detectados.

Se logró que el hexápodo evitara obstáculos fijos y dinámicos realizando un giro para desplazarse en otra dirección y continuar con el andar.

Implementar el algoritmo de detección de objetos representa bajar el rendimiento de la Raspberry Pi destinado al procesamiento de la red neuronal y el andar del hexápodo, por lo que, al correr ambos procesos, el robot se vuelve un poco más lento.

AGRADECIMIENTOS

Gracias a la Universidad de Guanajuato y al programa *Veranos UG* por el apoyo brindado para poder llevar a cabo esta investigación. A los colaboradores que aportaron su tiempo y conocimiento para lograr culminar con éxito el estudio.

REFERENCIAS

[1] Yu, J., Tan, M., Chen, J., & Zhang, J., (2014). A Survey On CPG-Inspired Control Models and System Implementation, in IEEE Transactions on Neural Networks and Learning Systems, 25 (3), pp. 441-456. doi: 10.1109/TNNLS.2013.2280596.

[2] Ijspeert, A. J., (2008). Central Pattern Generators for Locomotion Control In Animals And Robots: A Review, Neural Networks. 21 (4), pp. 642-653, doi: http://dx.doi.org/10.1016/j.neunet.2008.03.014.

[3] Guerra Hernandez, E. I., Espinal, A., Batres Mendoza, P., Garcia Capulin, C. H., Romero Troncoso, R. De J., & Rostro-Gonzalez, H., (2017). A FPGA-Based Neuromorphic Locomotion System for Multi-Legged Robots, in IEEE Access, 5, pp.8301-8312, doi: 10.1109/ACCESS.2017.2696985.

[4] Espinal, A., Rostro Gonzalez, H., Carpio, M, Guerra Hernandez, E. I., Ornelas Rodriguez, M., Puga Soberanes, H. J., Sotelo Figueroa, M. A. & Melin, P., (2016). Quadrupedal Robot Locomotion: A Biologically Inspired Approach and Its Hardware Implementation, Computational Intelligence and Neuroscience, 2016 (2016). Doi: 10.1155/2016/5615618

[5] Rostro Gonzalez, H., Cerna Garcia, P. A., Trejo Caballero, G., Garcia Capulin, C. H., Ibarra Manzano, M. A., Avina Cervantes, J. G., & Torres Huitzil, C., (2015). A CPG System based On Spiking Neurons For Hexapod Robot Locomotion, Neurocomputing, 170, pp. 47-54. Doi: http://dx.doi.org/10.1016/j.neucom.2015.03.090.

[6] Espinal, A., Rostro Gonzalez, H., Carpio, M., Guerra Hernandez, E. I., Ornelas Rodriguez, M., & Sotelo Figueroa, M., (2016). Design of Spiking Central Pattern Generators for Multiple Locomotion Gaits in Hexapod Robots by Christiansen Grammar Evolution. Frontiers in Neurorobotics, 10. Doi: 10.3389/fnbot.2016.00006.

[7] Cuevas Arteaga, B., Domínguez Morales, J. P., Rostro González, H., Espinal, A., Jiménez Fernández, A. F, Gómez Rodríguez, F., & Linares Barranco, A., (2017). A SpiNNaker Application: Design, Implementation and Validation of SCPGs.

[8] Pérez Trujillo, A., Espinal, A., Sotelo Figueroa, M. A., & Rostro González, H., (2017). ePyNN: a low cost embedded system for simulating Spiking Neural Networks. (Aceptado en la 26th Annual Computational Neuroscience Meeting)

[9] Arena, P., (2000). The central pattern generator: a paradigm for artificial locomotion, Soft Computing, 4 (4), pp. 251-266. Doi: https://doi.org/10.1007/s005000000051

[10] Drazen, D., Lichtsteiner, P., & Häfliger, P., (2011). Toward real-time particle tracking using an event-based dynamic vision sensor. Springer-Verlag, Doi: https://doi.org/10.1007/s00348-011-1207-y.

[11] Delbruck, T., (2008). Frame-free Dynamic digital vision. Proceedings of the International Symposium on Secure-Life Electronics, Advanced Electronics for Quality Life and Society, pp. 21-26. Doi: 10.5167/uzh-17620.

[12] Milde, M., Blum, H., Dietmüller, A., Sumislawska, D., Conradt, J., Indiveri, G., & Sandamirskaya, Y., (2017). Obstacle avoidance and target acquisition for robot navigation using a mixed signal analog/digital neuromorphic processing system, 11, pp. 28. Doi: 10.3389/fnbot.2017.00028.