UNIVERSIDAD DEL VALLE DE GUATEMALA
Facultad de Ingeniería

Implementación y mejora de distintos algoritmos de robótica de enjambre a robots físicos presentes en la UVG

Protocolo de trabajo de graduación presentado por Alex Daniel Maas Esquivel, estudiante de Ingeniería Mecatronica

Guatemala,

2021

Resumen

Antecedentes

0.1. PSO

En la tercera tesis, desarrollada por Eduardo Santizo () se mejoro el desempeño del algoritmo de optimización de partículas PSO asiendo uso del método denominado PSO Tunner, el cual consiste de una red neuronal recurrente la cual es capaz de tomar diferentes métricas de las partículas PSO y tornarlas atravesé de su procesamiento por medio de una red LSTM, GRU o BiLSTM y busca realizar una predicción de los hiper parámetros que debería emplear el algoritmo. Esta predicción es de carácter dinámico, lo que hace que en cada iteración se generan las métricas que describen al enjambre y use estos resultados para la siguen te iteración buscando así reducir el tiempo de convergencia y susceptibilidad a mínimos locales del PSO original [1].

Además elabora una alternativa al algoritmo de navegación alrededor de un ambiente conocido por medio de programación dinámica basándose en el ejemplo de programación dinámica Gridworld. Este nos presenta el caso donde un agente se mueve dentro de un espacio de estados representado en la forma de una cuadrícula y únicamente puede realizar 4 movimientos: Moverse hacia arriba, abajo, izquierda o derecha. De acuerdo a su estado actual y ultimo movimiento es capaz de moverse a un nuevo estado y obtiene una recompensa, el agente busca obtener el máximo de recompensas posibles generando así una ruta óptima desde cada estado hasta la meta [1].

Estos avances fueron compactados en un grupo de herramientas llamado Swarm Robotics Toolbox para ser usado en futuras fases.

0.2. Ant Colony

En la tercera tesis, desarrollada por Gabriela Iriarte () se busco crear una alternativa al Modified Particle Swarm Optimization (MPSO) para su futuro uso en los Bitbots de la UVG, desarrollando así el algoritmo de planificador de trayectorias Ant Colony. Se implemento el algoritmo Simple Ant Colony, y después el algoritmo Ant System [2].

Se emplearon diversos controladores y se modifico el camino encontrado interpolándolo para crear más metas y lograr una trayectoria suave. Los parámetros encontrados se validaron por medio de simulaciones computarizadas permitiendo visualizar el comportamiento de colonia y adaptar los modelos del movimiento y de la cinemática a los Bitbots [2].

0.3. Robotarium de Georgia Tech

El proyecto de Robotarium proporciona una plataforma de investigación robótica de enjambre de acceso remoto. En esta se permite a personas de todo el mundo hacer diversas

pruebas con sus algoritmos de robótica de enjambre a sus propios robots con fines de apoyar la investigación y seguir buscando formas de mejorar las trayectorias definidas. Para utilizar la plataforma se debe descargar el simulador que se encuentra en la pagina del Robotarium el cual presenta la opción de descargarlo en Matlab o Python, se debe registrar en la página del Robotarium y esperar a ser aprobado para crear el experimento [3].

Se presenta una base plana con bordes que limitan el movimiento de los robots a utilizar, estas cumplen la función de fronteras del espacio de búsqueda. La base es de un color blanco para facilitar el procesamiento de imágenes y se cuenta con una cámara ubicada en el techo apuntando a toda la base, con esta lograr capturar las posiciones de los diferentes robots de pruebas [3].

Figura 1: E-puck [3]

Justificación

Objetivos

Objetivo General

Realizar una correcta implementación de los algoritmos de robótica de enjambre ya existentes para su uso en robots físicos comerciales.

Objetivos Específicos

- Evaluar distintas opciones de robots y seleccionar la mas adecuada para su uso en aplicaciones de robótica swarm.
- Reestructuración de algoritmo de robótica de enjambre ya existente hacia una plataforma comercial para su implementación en robots físicos.

- Validar la correcta implementación del algoritmo de robótica de enjambre mediante tiempos de reacción y localización de mínimos y máximos conocidos dentro de un ambiente controlado.

Marco teórico

0.4. Particle Swarm Optimization PSO

El algoritmo de Optimización del enjambre de partículas (PSO) fue creado por por el Dr. Russell Eberhart y el Dr. James Kennedy en el año de 1995. Este tiene origen la simulación de de comportamientos sociales utilizando herramientas e ideas tomadas por gráficos computa rizados e investigación sobre psicología social; además de una clara inspiración en el comportamiento social de las crías de aves y la educación de los peces [4] .

El algoritmo PSO pertenece a las técnicas denominadas optimización inteligente y se clasifica como un algoritmo estocástico de optimización basado en población. A esta clasificación igualmente pertenecen los Algoritmos Genéticos (AG). PSO es intrínsecamente paralelo. La mayoría de algoritmos clásicos operan secuencialmente y pueden explorar el espacio de solución solamente en una dirección a la vez [4].

Este algoritmo es iniciado y simula un grupo aleatorio de partículas a las cuales se les asigna una posición y velocidad inicial, a estas partículas se les conoce como "soluciones", para luego proceder a actualizar las generaciones de estas para encontrar la solución óptima. En cada iteración cada partícula es actualizada por los siguientes 2 mejores resultados [5].

El primero de estos se le conoce como la mejor solución lograda hasta ahora por cada partícula y recibe el nombre de **local best**. El segundo mejor valor es rastreado por el PSO y este proceso se repite hasta que se cumpla con un número de iteraciones especifica o se logre la convergencia del algoritmo. La convergencia se alcanza cuando todas las partículas son atraídas a la partícula con la mejor solución la cual recibe el nombre de **global best** [5]. Con estos datos es posible calcular los dos primeros factores principales de la ecuación de PSO para cada partícula, los cuales son:

- Factor Cognitivo: $P_{local} - P_i$
- Factor Social: $P_{global} - P_i$

Donde P_i representa cada una de las soluciones/posiciones actuales. Para verificar que tan buena es la posición actual para cada partícula se usa la denominada **funcion fitness** que se da por $f(P)$ [6], el valor escalar que genera como resultado se le denomina costo el objetivo de las partículas es encontrar un conjunto de coordenadas que generen el valor de costo más pequeño posible dentro de una región dada.

Se toma en cuenta una diversidad de factores, entre ellos, el factor de escalamiento el

cual consiste en hacer que las partículas tengan una mayor área de búsqueda o hacer que se concentren en un área mas reducida, definidas como $C1$ y $C2$. El factor de uniformidad son 2 numeros aleatorios que van entre 0 y 1; se definen como $r1$ y $r2$. El factor de constricción φ el cual se encarga de controlar la velocidad de cada partícula [6]. El factor de inercia w el cual se encarga de controlar cuanta memoria puede almacenar cada partícula. De forma que podemos armar la ecuación de velocidad brindada por el PSO.

$$V_{i+1} = \varphi[wV_i + C1r1(P_{local} - P_i) + C2r2(P_{global} - P_i)] \tag{1}$$

Una vez actualizadas las velocidades de todas las partículas,se calcula las posiciones de cada una de estas:

$$X_{i+1} = X_i + V_{i+1} * \Delta t \tag{2}$$

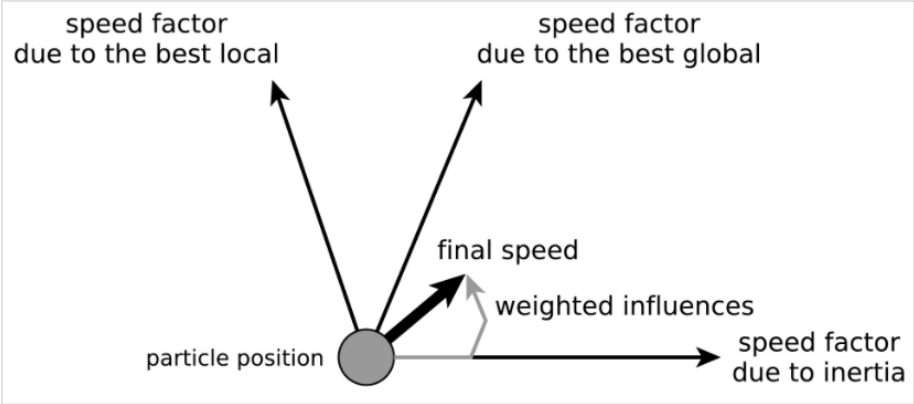

Figura 2: Efectos sobre la partícula [4]

0.5. Ant Colony Optimization

Este fue creado por Marco Dorigo en 1992 tomando como base el comportamiento de las hormigas las cuales a pesar de las limitadas capacidades de trabajo de cada hormiga individual, la colonia es capaz de realizar tareas de gran complejidad a través de la coordinación entre los individuos. La principal forma de comunicarse de las hormigas es modificar su ambiente esto debido a su escasa visión por lo que estas liberan feromonas para marcar rastros en el suelo e indicar el camino que deben seguir las demás [7].

0.5.1. Simple Ant Colony Optimization

El proceso de tomado para el caso mas simple del Ant Colony Optimization trata del problema general de la búsqueda del camino mas corto entre nodos presentes en un grafo $G = (V, E)$ donde E es la matriz que representa el conjunto de todas aristas del grafo y V el conjunto de todos los nodos del grafo. Cada posición dentro del grafo viene dado por un par de coordenadas (i, j) y en cada arista se tiene presenta cierta concentración de la

feromona, denotado $\tau_{i,j}$. La concentración inicial de feronomas en cada arista se calcula de manera aleatorio, cabe resaltar que esto no ocurre en la vida real ya que no habría pasado ninguna hormiga aun por esos lugares [7].

Las hormigas son colocadas en una posición inicial denominado nodo fuente y estas tienen libertad de escoger hacia que arista deciden dirigirse inicialmente, esto se realiza aleatoriamente. Casa hormiga es representada por el símbolo K y el largo del camino recorrido viene dado por L^K. Se construye iterativamente una solución para el problema en cuestión, las soluciones intermedias se denominan estados solución y en cada iteración del algoritmo, cada hormiga se mueve de un estado i a un estado j correspondiendo a una solución intermedia más completa. La de terminación del movimiento de las hormigas esta basado en la probabilidad $p_{i,j}^k$ [2].

$$p_{i,j}^k(t) = \frac{(\tau_{i,j}(t))^\alpha}{\sum_{k\epsilon j_k}(\tau_{i,j}(t))^\alpha} \tag{3}$$

Esto siempre que $j \ \epsilon \ N_i^k$. Donde N representa al conjunto de nodos viables conectados al nodo i. Además α es una constante usada para modular la influencia de la concentración de feromona. Como se menciona previamente, luego de encontrar comida en un nodo denominado nodo destino estas vuelven al nodo fuente esparciendo feromonas en cada arista que pasaron, este despliegue de feromonas esta dada por:

$$\tau_{i,j}(t+1) = (1-\rho)\tau_{i,j}(t) + \sum_{k=1}^{m} \Delta\tau_{i,j}^k(t) \tag{4}$$

Donde ρ es el factor de evaporación de las feromonas y m es el numero de hormigas, $\Delta\tau_{i,j}^k(t)$ es la cantidad de feromonas depositadas por la kth hormiga y esta dado por Q/L_k donde Q es una constante y L_k es el costo de la ruta de la kth hormiga [2].

0.5.2. Ant System (AS)

Este presenta ciertas mejoras al algoritmo descrito previamente, siendo este mas complejo. Una de las mejoras es la inclusión de la llamada información heurística en la ecuación de probabilidad y agregando capacidad de memoria con una lista tabú.

$$p_{i,j}^k(t) = \frac{(\tau_{i,j}(t))^\alpha(\eta_{i,j}(t))^\beta}{\sum_{k\epsilon j_k}(\tau_{i,j}(t))^\alpha(\eta_{i,j}(t))^\beta} \tag{5}$$

Donde η representa el inverso del costo de la arista [2].

0.6. Propuesta de Robots

0.6.1. Kilobot

El Kilobot es un sistema robótico de bajo costo, estos enjambres están inspirados en insectos sociales, como colonias de hormigas, que pueden buscar y encontrar eficientemente

fuentes de alimento en grandes ambientes complejos, transportar colectivamente grandes objetos y coordinar la construcción de nidos y puentes en tales ambientes [8].

El Kilobot está diseñado para proporcionar a los científicos un banco de pruebas físicas para avanzar en la comprensión del comportamiento colectivo y realizar su potencial para ofrecer soluciones para una amplia gama de desafíos. Este tiene un máximo de 33mm de diámetro [8].

Figura 3: Kilobot [8]

0.6.2. E-puck

Este robot fue desarrollado por la Escuela Politécnica Federal de Lausana (EPFL) el cual fue creado para fines educativos y de investigación. Este pequeño robot es cilíndrico con ruedas de 7 cm de diámetro, equipado con una variedad de sensores cuya movilidad está garantizada por un sistema de accionamiento diferencial. Tanto su diseño como librerías y manuales de uso son open source y todo se encuentra disponible en su pagina oficial [9].

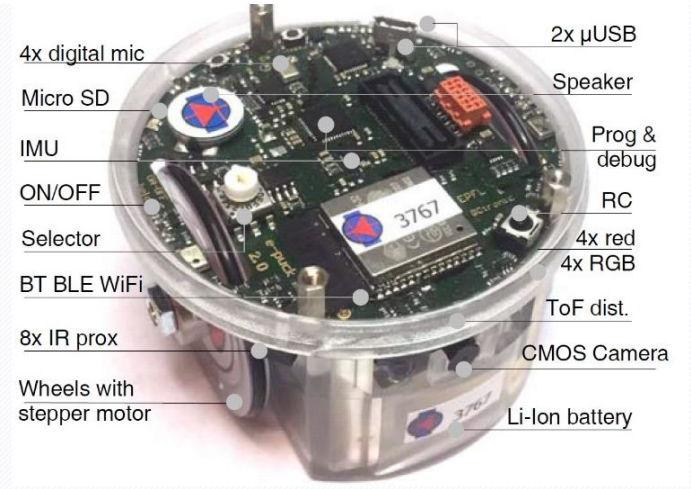

Figura 4: E-puck [9]

Metodología

Cronograma de actividades

Índice preliminar

Referencias

[1] E. A. S. Olivet, "Aprendizaje Reforzado y Aprendizaje Profundo en Aplicaciones de Robótica de Enjambre," Tesis doct., Universidad del Valle de Guatemala, Guatemala, Guatemala, 2020.

[2] G. I. Colmenares, "Aprendizaje Automático, Computación Evolutiva e Inteligencia de Enjambre para Aplicaciones de Robótica," Tesis doct., Universidad del Valle de Guatemala, Guatemala, Guatemala, 2020.

[3] Jason Maderer, *Robotarium: A Robotics Lab Accessible to All*, `https://www.news.gatech.edu/features/robotarium-robotics-lab-accessible-all`, Accessed: 2021-03-27, 2017.

[4] R. F. R. Grandi y C. Melchiorri, *A Navigation Strategy for Multi-Robot Systems Based on Particle Swarm Optimization Techniques*. Dubrovnik, Croatia: Dubrovnik, 2012.

[5] C. Duarte y C. J. Quiroga, *PSO algorithm*. Ciudad Universitaria, Santander, Colombia: Santander, 2010.

[6] A. S. A. Nadalini, "Algoritmo Modificado de Optimización de Enjambre de Partículas (MPSO)," Tesis doct., Universidad del Valle de Guatemala, Guatemala, Guatemala, 2019.

[7] M. Pedemonte, "Ant Colony Optimization," Tesis doct., Instituto de Computación, Universidad de la República Montevideo, Uruguay, Montevideo, Uruguay, 2015.

[8] K-Team, *KILOBOT*, `https://www.k-team.com/mobile-robotics-products/kilobot`, Accessed: 2021-02-25, 2017.

[9] GCtronic, *e-puck education robot*, `http://www.e-puck.org`, Accessed: 2021-03-27, 2018.