Implementación y Validación del Algoritmo de Robótica de Enjambre *Particle Swarm Optimization* en Sistemas Físicos

Implementation and Validation of the Particle Swarm Optimization Robotics Algorithm in Physical Systems

Alex Daniel Maas Esquivel (maa17146@uvg.edu.gt)

Departamento de Ingeniería Electrónica, Mecatrónica y Biomédica, Facultad de Ingeniería, Universidad Del Valle de Guatemala

Resumen

La robótica de enjambre emplea patrones de comportamiento colectivo mediante interacciones entre robots y robots con su entorno con el fin de alcanzar objetivos o metas establecidas. Uno de los algoritmos de robótica de enjambre común mente usado es el *Particle Swarm Optimization* (PSO), el cual es utilizado como planificador de trayectorias. Para tener una trayectoria suave y controlada se desarrolla el *Modified Particle Swarm Optimization* (MPSO) el cual toma en cuenta las dimensiones físicas del robot e implementa una seria de controladores buscando un movimiento controlado, el cual fue desarrollado en webots como trabajo de graduación previo. Se busca implementar y validar este algoritmo que funciona bien a nivel de simulación en un sistema físico el cual sea capaz de obtener resultados similares.

Al no contar con una plataforma móvil operativa se desarrollan distintas técnicas para lograr la validación del algoritmo bajo ambientes controlados. Se desarrollaron dos sistemas de comunicación, el primero que permita la obtención de la pose de cada agente en la mesa de pruebas de la UVG en tiempo real usando un algoritmo de visión por computadora desarrollado como otro proyecto de tesis; el segundo que permita el intercambio de información valiosa entre agentes que ayudan al propio algoritmo PSO. Se llevan a cabo dos pruebas para validar la correcta implementación, la primera fue una comparación directa con los resultados obtenidos a nivel de simulación haciendo uso del software webots, la segunda se llevo acabo en la mesa de pruebas de la UVG donde el algoritmo indica las nuevas posiciones calculadas y de forma manual son seguidas por marcadores para tener una validación en tiempo real.

Los resultados obtenidos en la comparación directa contra los obtenidos a nivel de simulación son aceptados a las posiciones nuevas calculas muy similares, indicando los mismos puntos de convergencia esto queda demostrado al obtener porcentajes de error por debajo de 1%. Los resultados obtenidos en la mesa de pruebas son aceptados al obtener direcciones de movimiento esperadas según el conocimiento que se tiene de la meta.

Palabras clave: robótica de enjambre, PSO, iteraciones, punto de convergencia.

Abstract

Swarm robotics employs patterns of collective behavior through interactions between robots and robots with their environment to achieve set goals or objectives. One of the commonly used swarm robotics algorithms is Particle Swarm Optimization (PSO), which is used as a trajectory planner. To have a smooth and controlled trajectory, the Modified Particle Swarm Optimization (MPSO) is developed, which considers the physical dimensions of the robot and implements a series of controllers looking for a controlled movement, which was developed in webots as previous graduation work. It seeks to implement and validate this algorithm that works well at the simulation level in a physical system which can obtain similar results.

By not having an operating mobile platform, different techniques are developed to achieve algorithm validation under controlled environments. Two communication systems were developed, the first one that allows obtaining the pose of each agent on the UVG test table in real time using a computer vision algorithm developed as another thesis project; the second that allows the exchange of valuable information between agents that help the PSO algorithm itself. Two tests are carried out to validate the correct implementation, the first was a direct comparison with the results obtained at the simulation level using the webots software, the second was carried out at the UVG test table where the algorithm indicates the new positions calculated and manually are followed by markers for real-time validation.

The results obtained in the direct comparison against those obtained at the simulation level are accepted at the new positions, very similar calculations, indicating the same points of convergence, this is demonstrated by obtaining error percentages below 1%. The results obtained in the test table are accepted by obtaining expected directions of movement according to the knowledge of the goal.

Keywords: swarm robotics, PSO, iterations, convergence point.

Introducción

La robótica de enjambre o *swarm robotics* es una nueva aproximación a la coordinación de un gran número de robots relativamente simples. De forma que estos mismos robots sean capaces de llevar a cabo tareas colectivas que están fuera de las capacidades de un único robot (Tortos 2013). El algoritmo PSO pertenece a las técnicas denominadas optimización inteligente y se clasifica como un algoritmo estocástico de optimización basado en población.

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 (Duarte y Quiroga 2010). 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* (Grandi y Melchiorri 2012).

Con estos datos es posible calcular los dos primeros factores principales de la ecuación del PSO

$$F_{cognitivo} = P_{local} - P_i$$

$$F_{social} = P_{global} - P_i$$

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 función costo o *funcion fitness* el valor escalar que genera como resultado se le denomina costo y 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 (Duarte y Quiroga 2010).

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 más reducida, definidas como $C1$ y $C2$. El factor de uniformidad corresponde a dos números aleatorios que van entre 0 y 1; se definen como r $r1$ y $r2$. El factor de constricción φ el cual se encarga de controlar la longitud de los pasos que cada partícula puede dar en cada iteración. El factor de inercia ω el cual se encarga de controlar cuanta memoria puede almacenar cada partícula (Duarte y Quiroga 2010).

De forma que podemos armar la ecuación de velocidad brindada por el PSO.

$$V_{i+1} = \varphi\left[\omega V_i + C1r1\left(F_{cognitivo}\right) + C2r2(F_{social})\right]$$

Donde V_i representa la velocidad actual y V_{i+1} la nueva velocidad calculada para la partícula. Una vez actualizadas las velocidades de todas las partículas, se calcula las posiciones de cada una de estas

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

Donde X representa la posición actual y X_{i+1} la nueva posición calculada para la partícula, Δt es el tiempo que le toma al algoritmo realizar cada iteración.

Se tomó el modelo estándar y existen del PSO y se procedió a modificarlo para que este tome en consideración las dimensiones de robots físicos y la velocidad a la que estos pueden moverse, además se implementaron diferentes tipos de controladores para buscar el robot pueda llegar al punto de meta realizando una trayectoria suave y controlada, este fue denominado *Modified Particle Swarm Optimization* (MPSO). El algoritmo fue desarrollado en fases previas a este proyecto y funciona bastante bien a nivel de simulación, donde se empleó el software webots.

En este trabajo se presenta la implementación de este algoritmo en un sistema físico, la validación del sistema físico y entorno de programación que mejor se ajuste a las necesidades, la correcta migración del algoritmo y pruebas de validación. Se busca replicar de mejor forma los resultados obtenidos a nivel de

simulación (webots) mientras el PSO es ejecutado en un sistema fisco. Además, se realiza otra validación usando una mesa de pruebas en conjunto con un algoritmo de visión por computadora para lograr un paso mas hacia una futura aplicación con robots móviles con ruedas.

Materiales y métodos

Para la implementación del algoritmo MPSO se evaluaron distintos tipos de microcontroladores, sistemas embebidos y entornos de desarrollo. Se selecciona el lenguaje de programación y el tipo de robot móvil que mejor se adapte a las necesidades del algoritmo. Al tener distintas opciones de microcontroladores, plataformas de evaluación y ordenadores se realiza un *trade study*, el cual es un estudio que ayuda a identificar una solución entre una lista de soluciones calificadas, estas son: Raspberry Pi, Tiva C, Arduino Uno, PIC-8bits, PIC-16-bits, PIC-32bits.

Entre los criterios usados están: capacidad de memoria, frecuencia de operación, adaptabilidad a robots móviles, disponibilidad, costo y previo uso. Se le asigno una calificación de 1 a 6 (siendo 6 la mejor calificación) a todas las posibles soluciones, las calificaciones se basaron en la literatura (Valdés y Areny, 2007; M. T. Inc., 2013; T. Instruments, 2013). En la figura 1 se observan los resultados y la mejor solución, siendo esta el ordenar Raspberry Pi.

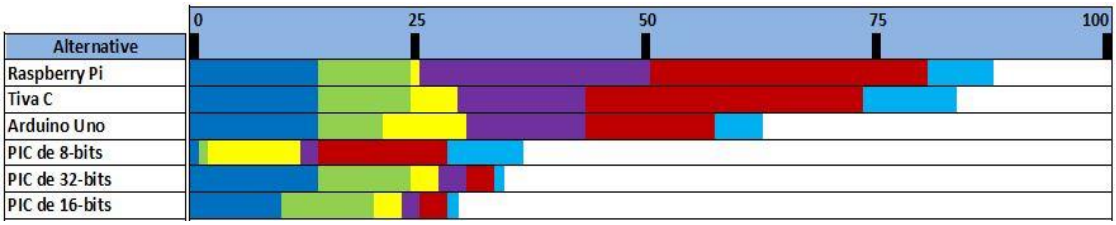

Figura 1. Trade Study sistema físico.

Se evaluaron distintos lenguajes de programación de alto nivel para realizar la implementación del algoritmo MPSO. Las soluciones calificadas son los lenguajes: C, C++, Python, Java y Matlab. Los criterios usados fueron: disponibilidad de librerías, adaptabilidad y previo uso. Se le asigno una calificación de 1 a 5 (siendo 5 la mejor calificación) a todas las posibles soluciones. En la figura 2 se observan los resultados y la mejor solución, siendo el lenguaje C.

Figura 2. Trade Study lenguaje de programación.

Como primera validación se implemento el algoritmo ordinario PSO en el ordenador Raspberry Pi para comparar su funcionamiento con el desarrollado en Matlab para fases anteriores. Enfocando la implementación a robots móviles se implementó el algoritmo MPSO en cada dispositivo a utilizar en donde cada agente será capaz de ejecutar su propio algoritmo. Cada agente tendrá la capacidad de recibir sus coordenadas y orientación actual, así como comunicarse con otros agentes para tomar una decisión en conjunto y proceder con el cálculo de una nueva posición. Para todas las demás pruebas se usaron los modelos Raspberry Pi 3B, 3B+ y 4, por lo que estos resultados son replicables en cualquiera de estos modelos.

Implementación física del PSO

Con la limitante de no contar con una plataforma física operativa (robot móvil con ruedas) se buscaron alternativas para lograr una comunicación entre todos los agentes (parte fundamental del algoritmo) y que cada agente sea capaz de obtener su orientación y coordenadas actuales como entradas para el algoritmo. Para esto se usó programación multihilos para que estas tareas pueden ser ejecutadas de forma simultanea y no afecten la ejecución del algoritmo principal.

Como parte de otro proyecto de tesis se continuo con el desarrollo de un algoritmo de visión por computadora para el reconocimiento de la pose de agentes colocados en una mesa de pruebas. Se uso una red local y un protocolo de comunicación UDP para lograr una vía de comunicación entre este algoritmo desarrollado en Matlab y el algoritmo PSO desarrollado en lenguaje C. Se estableció que los agentes fueran los servidores mientras que el algoritmo de visión por computadora la de cliente, este ultimo encargado de enviar a direcciones especificas las poses leídas.

Una parte fundamental del correcto funcionamiento del PSO es el poder comunicar la posición actual, así como el costo actual a los otros agentes para tomar una decisión sobre el recorrido a tomar. El intercambio de información debe permitir a todos los agentes enviar tanto como recibir datos, con esto en cuenta una comunicación del tipo cliente-servidor no sería eficiente entre agentes. Se uso un protocolo de comunicación UDP del tipo *broadcast* en la cual todos los agentes conectados a la red pueden enviar y recibir la misma información, en la figura 3 se ejemplifica este proceso donde una cantidad N de agentes pueden intercambian información entre sí.

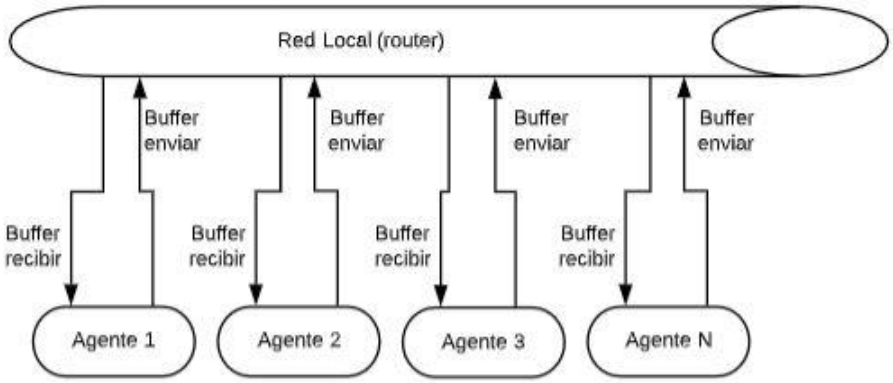

Figura 3. Representación del envío y recepción de datos entre agentes.

Ambientes Controlados

La validación del algoritmo PSO en un sistema físico se realizó de dos formas, la primera fue una comparación directa con los resultados del mismo algoritmo evaluado en webots en fases anteriores y la segunda fue una demostración usando la mesa de pruebas de la UVG.

Validación implementando Webots

Para la primera validación se usó webots para obtener las coordenadas y orientación de cada agente, se usó un total de siete agentes para las distintas pruebas bajo ambientes controlados. Para tener una comparación valida entre ambos algoritmos se trabajó bajo las mismas métricas, las cuales fueron: posición inicial, cantidad de agentes, función costo, tipo de controlador, velocidad de los motores, dimensiones físicas, tiempo de muestreo, inercia y demás parámetros internos del PSO.

La pose de los agentes con respecto a webots se almaceno en un archivo de texto el cual se ordena y posteriormente se envía a cada dirección especifica mediante matlab. Con esto se obtiene la trayectoria esperada que deben de realizar los agentes y permite realizar una comparación valida con los resultados obtenidos en fases anteriores.

Se probaron distintas combinaciones de funciones costo y parámetros de inercia, los cuales permiten verificar el costo actual de cada agente y las nuevas posiciones dada por el algoritmo PSO. Se probaron varios tipos de controladores para verificar el valor enviado a los motores (en una futura plataforma móvil) hagan sentido con relación a las nuevas posiciones previamente calculadas.

Validación implementando algoritmo visión por computadora

Para la segunda validación se usó el algoritmo de visión por computadora desarrollado como otro trabajo de graduación para obtener las coordenadas y orientación de cada gente. Por las dimensiones de la mesa de pruebas y el tamaño de los marcadores se usaron únicamente cuatro agentes, el algoritmo de visión por computadora toma tiempo en procesar cada imagen por lo que cada iteración del PSO se hace al recibir una nueva coordenada.

Resultados y discusión

Validación del dispositivo físico

Para tener una comparación valida ambos algoritmos PSO (ejecutado en la Raspberry y matlab) fueron ajustados con las mismas métricas, se probaron las funciones costo *Sphere* y *Himmelblau*, en combinación de un tipo de inercia constante y caótica. Se generaron 3 casos para cada combinación donde se variaron los agentes. En la tabla 1 se muestran los resultados de la función costo *Sphere* y una inercia constante.

Tabla 1. Descripción de la tabla 1.

Cantidad de Agentes	Matlab		Raspberry Pi		Porcentaje de Error	
	X	Y	X	Y	X	Y
10	0.0000050	-0.0000110	0.0000067	-0.0000066	34%	40%
50	-0.0000002	0.0000002	-0.0000001	0.0000003	40%	55%
100	0.0000016	-0.0000012	0.0000010	-0.0000016	39%	33%

Al no contar con dimensiones físicas, los agentes se modelan como partículas haciendo que las más leves variaciones en cuento a su posición final generen porcentaje de error se alto. En la tabla 2 se muestran los resultados de la función costo *Himmelblau y* una inercia caótica.

Tabla 2. Descripción de la tabla 1.

Cantidad de Agentes	Matlab		Raspberry Pi		Porcentaje de Error	
	X	Y	X	Y	X	Y
10	3.5844283	-1.8481265	3.5844283	-1.8481265	0%	0%
50	3.5844283	-1.8481265	3.5844283	-1.8481265	0%	0%
100	3.0000000	2.0000000	3.0000000	2.0000000	0%	0%

Con esta combinación de parámetros se obtuvo un mejor resultado obteniendo hasta un 0% de error, se continuaron haciendo pruebas de comparación para validar el dispositivo físico.

Validación de la implementación de algoritmo en sistemas físicos

Para la primera validación se realizaron cinco pruebas combinando funciones costo, controladores y tipo de inercia. En la tabla 3 se muestra las primeras iteraciones del algoritmo para la prueba número 5, en la cual se evaluó la función costo *Rosenbrock*, un tipo de inercia *random* y un controlador de pose. Estos resultados corresponden al agente No.7.

Tabla 3. Prueba 5 primeras iteraciones agente 7.

	Webots		Raspberry		% Error	
Nuevas posiciones						
iteración	X_w	Y_w	X_r	Y_r	X	Y
1	0.141206	0.218473	0.14119	0.21849	0.000%	0.000%
2	0.049871	0.163887	0.021058	0.146722	0.041%	0.025%
3	-0.032152	0.11489	0.027807	0.126922	0.086%	0.017%
4	-0.013096	0.126416	0.030789	0.109509	0.063%	0.024%

El error obtenido representa la resta del valor teórico (webots) menos el valor experimental (Raspberry) con relación a las dimensiones físicas del robot móvil, que para estas pruebas se toman las dimensiones del famoso robot móvil E-puck, estas en milímetros. Como se ve en la figura 4 el agente comienza en el mismo punto en ambas pruebas y si bien toma las nuevas posiciones varían ligeramente presentan el mismo comportamiento.

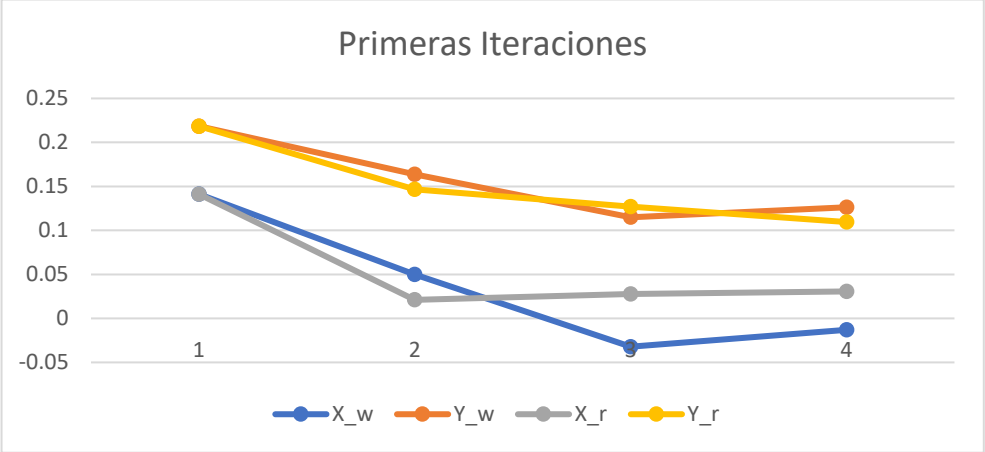

Figura 4. Gráfico de primeras iteraciones.

En la tabla 4 se muestran las iteraciones finales del algoritmo para la misma prueba y el mismo agente, esta prueba tomo 2041 iteraciones para detenerse.

Tabla 4. Prueba 5 ultimas iteraciones agentes 7.

	Webots		Raspberry		% Error	
Nuevas posiciones						
iteración	X	Y	X	Y	X	Y
2038	0.37453	0.163496	0.375238	0.163597	0.001%	0.000%
2039	0.374411	0.163480	0.374498	0.163492	0.000%	0.000%
2040	0.37428	0.163462	0.374293	0.163463	0.000%	0.000%
2041	0.374204	0.16345	0.374241	0.163452	0.000%	0.000%

Como se ve en la figura 5 el agente calcula las mismas nuevas posiciones en ambas implementaciones, esto en las iteraciones finales, lo que muestra una buena convergencia del agente 7 para esta prueba.

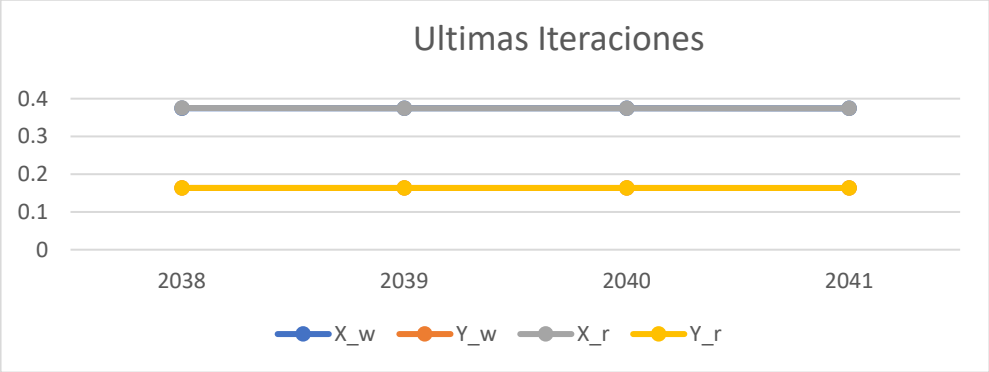

Figura 5. Gráfico de iteraciones finales.

Para la segunda validación se evaluó la función costo *Shpere*, un tipo de inercia exponencial y un controlador PID. En la figura 6 se muestran las posiciones iniciales de los 4 marcadores, el orden de los agentes fue de 1-4 comenzando de abajo hacia arriba.

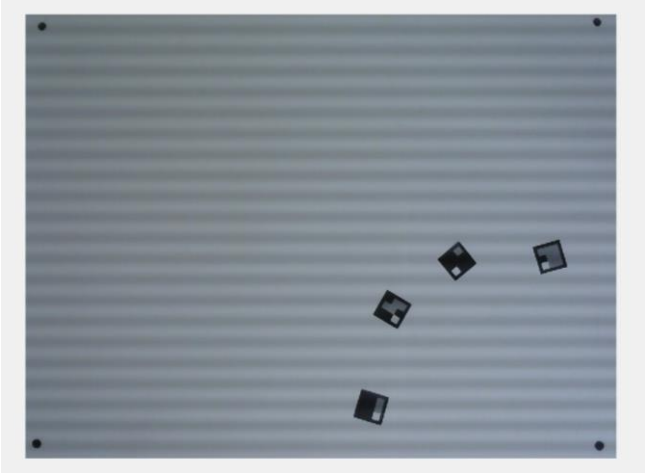

Figura 6. Posición inicial de los marcadores.

En la figura 7 se ven los resultados del algoritmo PSO ejecutado en 4 dispositivos, la nueva posición calculada para X para los agentes 1, 2 y 3 nos indica que se deben mover a la derecha mientras que la coordenada Y nos indica que deben de bajar, ambas velocidades están en su limite permitiendo, indicando un movimiento constante. Las nuevas posiciones del agente 4 presentan el mismo movimiento, pero en menor cantidad ya que este se encuentra mas pegado al limite de la mesa de pruebas y sus velocidades varían entre -2 y 2.

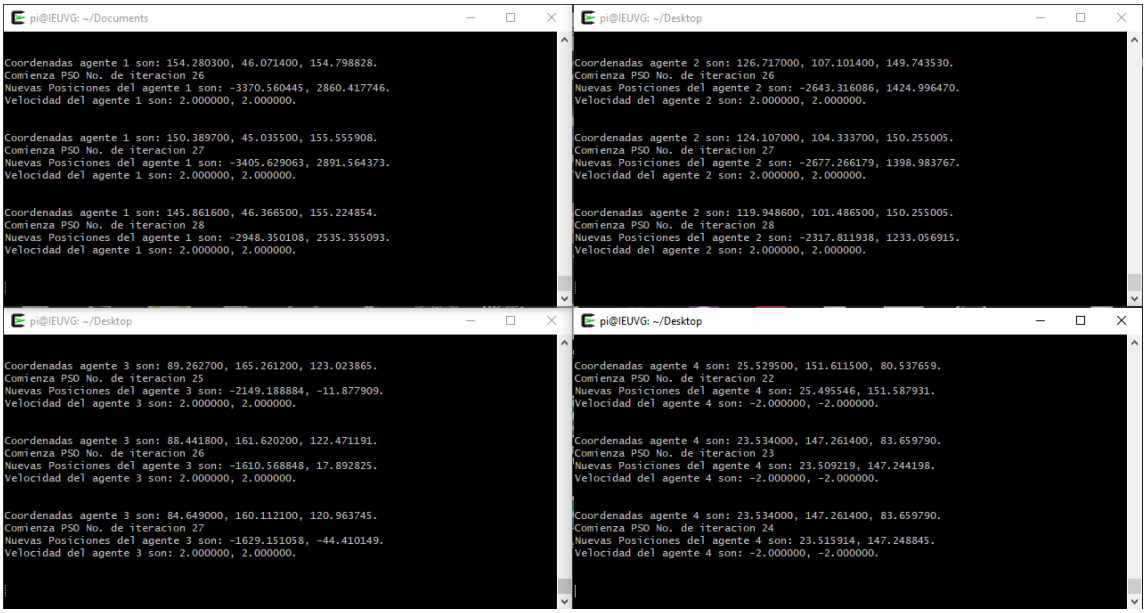

Figura 7. Proceso iterativo.

En la figura 8 se muestran las posiciones finales de los agentes aproximadamente en la interacción 35, siendo esta posición la ultima que el algoritmo de visión por computadora logra capturar, luego de esto deja de reconocer los marcadores.

Figura 8. Posición final de los marcadores.

Conclusiones

Los resultados obtenidos de la comparación directa entre webots y el ordenador Raspberry Pi en cuanto al cálculo de nuevas posiciones dadas por el PSO son aceptadas en base a los porcentajes de error y graficas generadas, por lo que se obtiene una implementación exitosa del algoritmo PSO en sistemas físicos.

Lo resultados obtenidos en la mesa de pruebas nos indican un correcto movimiento de los agentes colocados sobre la mesa, el algoritmo PSO en sistemas físicos se adapta a las posiciones iniciales de los marcadores y realiza un calculo de nuevas posiciones que hace sentido de acuerdo con la posición actual de los marcadores.

Al evaluar distintas plataformas y microcontroladores y de acuerdo con la validación del PSO ordinario y su comparación con su versión en Matlab el ordenador Raspberry Pi es la mejor opción para la implementación del PSO en un sistema físico.

Bibliografía

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

F. Valdes y R. Areny, Microcontroladores Fundamentos y Aplicaciones con PIC. España: Marcombo, 2007

L. S. Tortosa, "Ajentes y enjambres artificiales: modelado y comportamientos parasistemas de enjambre robóticos," phdthesis, Universidad de Alicante, España, 2013

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

M. T. Inc, General Purpose, 16-Bit Flash Microcontrollers with XLP Technology Data Sheet,Microchip Technology Inc,september2013.68

M. T. Inc, PIC16F87XAData Sheet, Microchip Technology Inc, september2013.

M. T. Inc, PIC32MZ Embedded Connectivity with Floating Point Unit (EF) Family, Micro-chip Technology Inc, september2013.

T. Instruments, Tiva C Series TM4C123G LaunchPad Evaluation Board, Texas Ins-truments, 2013.