Evaluación de trayectorias para un robot móvil autónomo aplicable en pequeñas y medianas producciones agrícolas

Evaluation of paths for a Mobile Robot in Small and Medium-applied Agricultural productions

Pedro Martín G.[1], Oscar Hernández M.[2], Anazíbio Batista de F.[3], Enio Bandarra F.[4], Rogério Sales G.[5]
Ingeniería en Automatización, Universidad de La Salle, Bogotá, D.C. Colombia
pmartin@unisalle.edu.co
Engenharia Mecânica, Universidade Federal de Uberlândia, Uberlândia, MG, Brasil.
oscarshm@hotmail.com
neto.6666@hotmail.com
bandarra@mecanica.ufu.br
rsgoncalves@mecanica.ufu.br

***Resumen*— Se resume una aplicación de un robot móvil desarrollado para pequeñas y medianas plantaciones, con navegación autónoma dentro del cultivo, detectando y/o evitando obstáculos y haciendo dosificación controlada de fertilizantes según la demanda localizada. Definidas las condiciones del cultivo y su distribución; fueron probados algunos algoritmos de planeación de trayectorias: Dijkstra, Floyd y A*, para seleccionar el que mejor se adapta a la aplicación. También se presenta una aplicación virtual desarrollada para la simulación de trayectorias aplicando los algoritmos, y finalmente, los resultados de pruebas experimentales sobre una maqueta como representación del cultivo usando un robot a escala.**

***Palabras clave*— Agricultura de precisión, robot móvil, control, automatización, planeación de trayectorias.**

***Abstract*— It summarizes an application of a mobile robot developed for small and medium-sized plantations, with autonomous navigation within the crop, detecting and/or avoiding obstacles and making controlled release fertilizers according to punctual demand of dosage. Defined the culture conditions and their distribution were tested some path planning algorithms: Dijkstra, Floyd and A *, to select the one best suited to the application. It also presents a virtual application developed for path simulation using these algorithms, and finally, the results of experimental tests on a model as a representation of crop using a robot to scale..**

***Key Word* — Precision agriculture, mobile robot, control, automation, path planning.**

I. INTRODUCCIÓN

Con el aumento creciente de la demanda de productos alimenticios en el mundo entero, es absolutamente necesario el aumento significativo de la producción agrícola. Por lo tanto, es necesario aumentar cada vez más la eficiencia de los procesos de plantación. De esta forma, la aplicación de sistemas autónomos en los cultivos ha sido estudiada, principalmente, debido a la escasez de mano de obra calificada y producción mecanizada en el sector agrícola; especialmente en Colombia, donde la producción agrícola es en gran porcentaje a nivel familiar, trasmitida por herencia en forma experimental y en forma manual en pequeñas áreas de producción de tipo rural.

En cualquier tipo de producción, es común obtener el máximo beneficio con mínimo costo a partir del mejor uso de los recursos, como estrategia fundamental para obtener la máxima rentabilidad con la mejor calidad de los productos; para mayor productividad y competencia en los mercados. Por esta razón, la agricultura toma mayor relevancia cada día, aumentando la investigación en los diferentes campos relacionados, con el objeto de obtener un alto grado de producción interviniendo el suelo y el medio ambiente lo mínimo posible. Para eso, se requiere el desarrollo y aplicación de alta tecnología para planeación, producción y gestión automática de los cultivos. Eso es llamado hoy como "Agricultura de Precisión", según

[1] Ingeniero Mecánico, Ph.D(C), Profesor asistente III. Universidad de La Salle, Bogotá. Colombia.
[2] Engenheiro Mecânico. Ph. D. Universidade Federal de Uberlândia, MG. Brasil.
[3] Engenheiro Mecânico. Universidade Federal de Uberlândia, MG. Brasil.
[4] Engenheiro Mecânico, Ph. D. Universidade Federal de Uberlândia, MG. Brasil.
[5] Engenheiro Mecânico, Ph. D. Universidade Federal de Uberlândia, MG. Brasil.

Fecha de Recepción: 14 de Septiembre de 2012.
Fecha de Aceptación: 29 de Diciembre de 2012

Blackmore [1]. Esto permite obtener el máximo potencial del suelo por hectárea con menor uso de abonos, menor degradación del suelo y mayor rendimiento de la producción con menor costo; usando equipos automáticos que usan sensores [2] y detección [3], sistemas de posicionamiento global (GPS) [4-8], sistemas de información geográfica (GIS) [9,10] y aplicaciones robóticas para siembra [11,12], dosificación y/o aplicación de fertilizantes [13,14], recolección [15,16], irrigación [17] y desmalezado [18,19] entre otros. Estos sistemas están basados en tecnologías capaces de hacer una distribución apropiada de las plantas según las condiciones físicas, químicas y geográficas; estableciendo las trayectorias más cortas generadas por un sistema computarizado [20-23]; para un mejor aprovechamiento del área, lograr mayor productividad y con dosificación variable de fertilizantes en forma localizada. Considerando esta necesidad, fue realizado un estudio de desarrollo de un robot que pudiese moverse en forma autónoma en un ambiente que simula la plantación de papa, evitando posibles obstáculos que puedan existir, y siguiendo la trayectoria óptima (menor posible) entre el punto de partida y su destino.

II. CONTENIDO

2.1. **Ambiente de simulación.**

Fue utilizado como espacio de trabajo un ambiente que simula una plantación de papa, construido en MDF con dimensiones de 2,00 x 2,00 metros como se muestra en la Fig. 1.

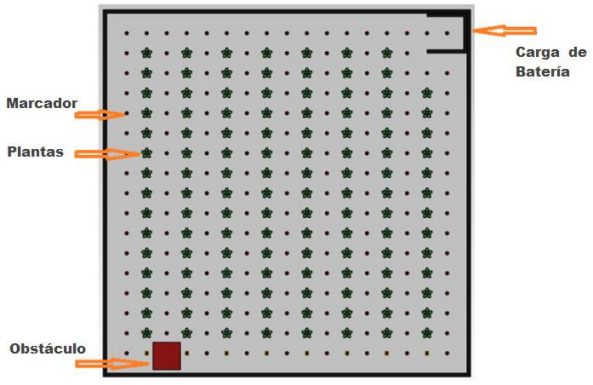

Figura 1. Distribución de las plantas en el espacio virtual y marcadores y obstáculos para la simulación a escala.

Los marcadores fueron adoptados para ayudar en la locomoción del robot por el escenario. Los marcadores y las supuestas plantas de papa fueron posicionados de tal manera que formaran una matriz espaciada con distancia patrón de 109 milímetros. Este escenario es una representación de una plantación real de papa a escala. Además de los marcadores y plantas de papa, existe en el escenario un local destinado a la estación de recarga de batería y otras operaciones como: recarga de fertilizante. En el escenario pueden existir obstáculos para el paso del robot como el mostrado en la Fig. 1.

2.2. **Simulación del sistema y el escenario.**

Cada uno de los puntos del cultivo (puntos de posicionamiento y de las plantas) es considerado como un objeto con información guardada en una base de datos incluyendo la información de composición del suelo, coordenadas de posición y cantidades de fertilizante requerida y aplicada. El sistema comienza leyendo los datos de las coordenadas de posición de los puntos y a partir de los datos de posición de punto actual y punto destino, aplica un algoritmo de generación de trayectorias (Dijkstra, Floyd o A*), para establecer el menor camino a recorrer por el vehículo entre los dos puntos, optimizando el recorrido del robot. Según estas características, fue programada en JAVA una aplicación en computador; para simular el movimiento del robot en el escenario. Los obstáculos pueden ser colocados en cualquier punto del escenario dentro del área de simulación mediante el ratón del computador. En la Fig. 2, fue capturada una imagen del computador, donde se muestra el área de simulación en la parte izquierda de la ventana, con la estación de recarga (home) en la parte superior izquierda y de color verde. En la parte inferior izquierda está la zona de simulación, con las filas de cultivo en color café, los puntos de posicionamiento en color rojo y el obstáculo representado por el cubo rojo. El vehículo representado en color naranja, muestra en la parte inferior entre paréntesis, la dirección de cada punto alcanzado. En el recuadro azul, se resaltan el puerto de comunicación remota usado y los botones de conexión y desconexión. En la parte derecha de la ventana, está el área de transmisión de la información para la verificación de las condiciones de simulación. El recuadro rojo muestra la posición actual, las coordenadas de la estación de reposo (home), las coordenadas del punto de destino solicitado y la pestaña de selección del algoritmo a utilizar. En el recuadro verde, se muestran en la parte superior, la secuencia de puntos de trayectoria a seguir (fondo gris) obtenidos al aplicar el algoritmo correspondiente; y en la parte inferior (fondo color naranja), los puntos alcanzados cuando el vehículo se desplaza según la simulación o los avisos de solicitud de nueva trayectoria de desplazamiento. En el recuadro naranja, se muestran las condiciones de simulación de arcos y posiciones del grafo aplicado, historial, tamaño del arreglo aplicado y tipo de comunicación para enviar los datos obtenidos.

Fue probada la capacidad de simulación de la aplicación, midiendo el consumo de recursos de cómputo según el tiempo de procesamiento del volumen de información: primero, fue medido el tiempo de carga y almacenamiento de los datos para todos los puntos de una hectárea (47619 puntos) según la distribución del cultivo y el consumo de recursos de computador en porcentaje. Después, fue probada la influencia de la precisión de posicionamiento entre 0 y 25 cm de error radial alrededor de cada punto de trayectoria. Finalmente, fueron desarrolladas pruebas con los tres algoritmos, con obstáculos y sin ellos, como evaluación de comportamiento del sistema completo.

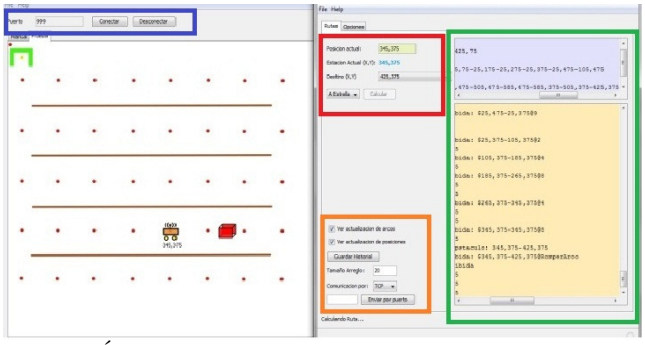

Figura 2. Área de simulación de trayectorias (izquierda) y área de transferencia de datos (derecha) en la ventana de simulación por computador.

2.3. **Algoritmos de planeación de trayectorias**.

El *algoritmo A**[23] a partir de un mapa de puntos o nodos clasificados en alcanzables o inalcanzables, se parte de un punto actual donde está el robot y un punto destino a donde debe ir, de acuerdo con la Fig. 3. El algoritmo A* calcula la sucesión de puntos o nodos que componen la ruta de menor costo (menor distancia recorrida) entre todas las rutas compuestas por los nodos.

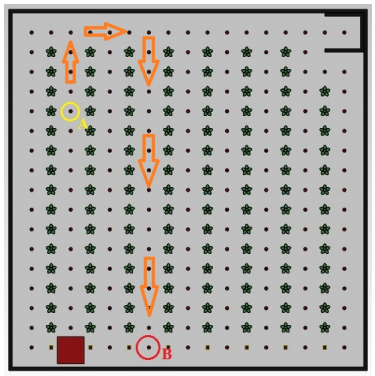

Figura 3. Situación inicial, donde los nodos libres son los puntos rojos, los nodos no disponibles por las plantas y el obstáculo cuadrado rojo, el punto inicial en amarillo (A) y el punto final en rojo (B) para determinar la trayectoria mas corta representada por la flechas en color naranja.

Para generar la trayectoria se necesitan dos listas: una abierta y una cerrada y una función heurística que estima los méritos de cada nodo que se genere. Por trabajar bajo la estructura de árbol de expansión mínima, la *lista de Abiertos* contiene los nodos que han sido generados pero que no han generado sus sucesores. En cambio, la *lista de Cerrados* contiene los nodos que ya han sido examinados.
Se considera que el vehículo parte desde el nodo *s* hasta un nodo destino *t*. Mediante una función de evaluación *f(n)* se decide heurísticamente qué probabilidad tiene el nodo *n* de ser expandido. El algoritmo A* realiza básicamente las siguientes operaciones:

Paso 1. Se marca el nodo inicial *s* como *Abierto*, se calcula *f(s)* y se introduce en la *lista de Abiertos*.
Paso 2. Se selecciona el nodo *n* de la lista cuya función de evaluación *f(n)* sea menor.
Paso 3. Si *n = t,* se marca *n* como *cerrado* y termina el algoritmo.
Paso 4. En otro caso, se marca *s* como *Cerrado*. Se calcula el nuevo costo para cada sucesor *m*, *f(m)* , y en caso de ser menor, se actualiza el costo y se inserta en la lista. Se Repite el paso 2.
Se define la función de evaluación *f(n)* como la suma de dos funciones:

$$f(n) = g(n) + h(n) \qquad (1)$$

donde *g(n)* es el costo actual del camino óptimo desde el origen hasta el nodo *n*, y *h(n)* es el costo de un camino óptimo desde el nodo *n* hasta el punto destino. Como la función de evaluación *f(n)* no es conocida a priori, se utiliza una función estimada *f*(n)* que se puede representar como en la ecuación anterior, pero teniendo en cuenta los costos estimados según la ecuación (2).

$$f^*(n) = g^*(n) + h^*(n) \qquad (2)$$

donde *g*(n)* es el costo estimado del camino desde el origen hasta el nodo *n* con el mínimo costo encontrado hasta el momento y *h*(n)* es una estimación del costo de un camino óptimo desde *n* hasta el destino. La función de costo estimada *g(n)** debe cumplir la condición:

$$g^*(n) \geq g(n) \geq 0 \qquad (3)$$

es decir, que la estimación de la función de costo debe ser mayor o igual que el costo real. Se dice que A* es admisible cuando encuentra siempre una solución óptima. Para ello es necesario que la función heurística *h*(n)* no sobreestime el valor costo real *h(n):*

$$0 \leq h^*(n) \leq h(n) \qquad (4)$$

Además, la función heurística h(n)* es localmente consistente si, para cada par de celdas adyacentes (a, b) se cumple la condición:

$$0 \leq h^*(a) \leq h^*(b) + c(a, b) \qquad (5)$$

donde c(a, b) es el costo del arco entre las celdas a y b. La elección de la función heurística *h(n)* = 0* es trivialmente admisible y localmente consistente, pero realizaría una búsqueda sin información equivalente a una búsqueda en anchura.

El *algoritmo de Dijkstra* primero marca todos los vértices como no utilizados. El algoritmo parte de un vértice origen que será ingresado, a partir de ese vértice se evaluan sus adyacentes, buscando el que esté más cerca del punto origen, se toma como punto intermedio y se verifica si es posible llegar más rápido a través de este vértice a los demás. Después se selecciona al siguiente más cercano (con las distancias ya actualizadas) y se repite el proceso. Esto se hace hasta que el vértice no utilizado más cercano sea el destino. Al proceso de actualizar las

distancias, tomando como punto intermedio al nuevo vértice se le conoce como relajación (relaxation).
Dijkstra usa una Cola de Prioridad o "Heap" que debe tener la propiedad de "Min-Heap" es decir cada vez que extraiga un elemento del Heap debe devolver el de menor valor, en nuestro caso dicho valor será el peso acumulado en los nodos.

El *algoritmo de Floyd* [24], calcula el camino mínimo entre todos los pares de vértices del grafo ponderado y no negativo mediante técnicas de programación dinámica.
Para ello se calculan una serie de matrices $D_k[i,j]$ que deben cumplir:

$$D_k[i,j] = D_{k-1}[i,k] + D_{k-1}[k,j] \quad \text{(pasando por k)} \qquad (6)$$

ó $$D_k[i,j] = D_{k-1}[i,k] \quad \text{(no pasando por k)} \qquad (7)$$

Donde:

$$D_k[i,j] = \text{mínimo}(Dk\text{-}1[i,k] + Dk\text{-}1[k,j]), D[i][j]) \qquad (8)$$

Longitud del camino más corto para ir desde el vértice "*i*" al vértice "*j*" pudiendo pasar por los vértices *1,2,...,* hasta *k.* Básicamente la expresión anterior nos dice: "si para ir desde el vértice "i" al "j" mejoramos pasando por el vértice "k", este se añade al camino. Además, para reconstruir el camino se hace uso de una matriz de trayectorias donde en cada iteración si se mejora el camino desde el vértice "*i*" al vértice "*j*" pasado por "k", este se anota en la matriz de trayectorias.

2.4. **Resultados de la simulación**.

Los resultados de generación de trayectorias con los tres algoritmos: Dijkstra, Floyd y A* se muestran en la Figs. 4 a 6. La Fig. 4 muestra el resultado de simulación del algoritmo de Dijkstra sin obstáculos a la izquierda, con la trayectoria obtenida en color azul claro y el mismo algoritmo con obstáculos (cubos rojos) a la derecha. Se muestra en azul la trayectoria obtenida, con la generación de una nueva ruta complementaria, una vez encontrado un obstáculo inesperado en el camino definido inicialmente.

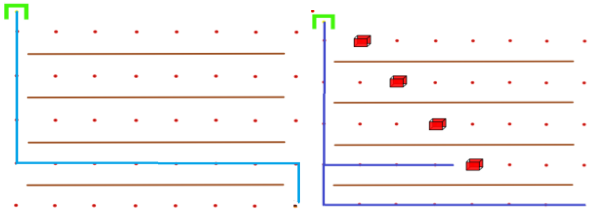

Figura 4. Ruta obtenida y simulada con algoritmo de Dijkstra, siguiendo las filas de cultivo cafés y evitando los obstáculos representados por los cubos rojos.

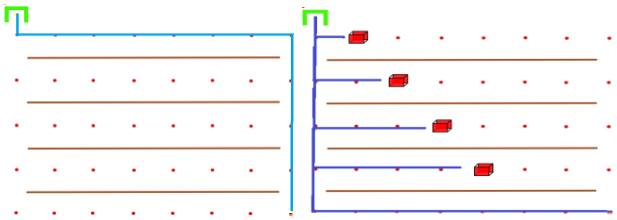

Figura 5. Ruta obtenida con algoritmo de Floyd sin obstáculos en color azul claro y con obstáculos inesperados (cubos rojos), en color azul oscuro.

La Fig. 5, a la izquierda muestra la ruta simulada con algoritmo de Floyd sin obstáculos en color azul claro nuevamente y a la derecha, con obstáculos en color azul oscuro. Cuando encuentra un obstáculo, el sistema pide una nueva ruta y elimina el punto de la lista de puntos disponibles y en la Fig. 6 es mostrada la simulación del algoritmo A* con la ruta resultante en color naranja.

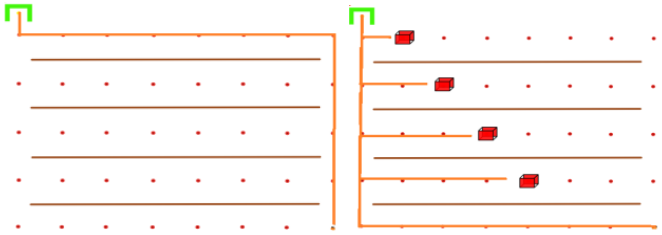

Figura 6. Ruta obtenida con algoritmo A* sin obstáculos (izquierda) y con obstáculos (derecha).

2.4.1. **Consumo de recursos de computador**.

Las pruebas de carga de datos en el computador, se hicieron para todos los puntos de una hectárea (47619) en dos equipos diferentes: el primero, con procesador INTEL XEON 2.67 GHz y el segundo, con procesador ATHLON X3 2.9 GHz, los dos con sistema operativo WINDOWS. En todas las pruebas fueron medidos los tiempos con el reloj del computador a través de los recursos del sistema.

La Tabla I muestra los resultados obtenidos en la carga de datos con los tres algoritmos en el primer computador. Se puede ver que el algoritmo A* consume menor cantidad de recursos en menor tiempo de carga. El algoritmo de Floyd presentó interrupción por la gran cantidad de datos del cultivo, no llegando a la convergencia.

Los resultados obtenidos con el segundo computador, se muestran en la Tabla II. En él fueron medidos los tiempos de carga y de almacenamiento de los datos y las líneas por separado. Se puede ver que los algoritmos tienen desempeño similar, pero el algoritmo A* tiene tiempo de ejecución menor.

TABLA I.
PRUEBAS DE ALGORITMOS CON EL PRIMER COMPUTADOR

PARÁMETRO	DIJKSTRA	A*	FLOYD
Uso de CPU (%)	26	26	98
Uso de Memoria (Mb)	970	949	977
Tiempo (s)	75	50	--

Tabla 1. Uso de recursos y tiempo tardado en la carga y almacenamiento de los datos; para el primer computador utilizado con cada uno de los tres algoritmos probados.

TABLA II.
PRUEBAS DE ALGORITMOS CON EL SEGUNDO COMPUTADOR

	A*		DIJKSTRA	
	TIEMPO (s)	**Uso de CPU** (%)	**TIEMPO** (s)	**Uso de CPU** (%)
Parte 1 carga datos	29	60	25	57
Parte 2 carga líneas	120	32.5	124	34,5
Duración algoritmo	145	28.5	155	28,5

Tabla 2. Resultados de consumo de recursos en el segundo computador de prueba, para los dos algoritmos: A* y Dijkstra.

2.4.2. **Simulación de precisión del sensor de posición**.

Se hicieron pruebas de precisión de posicionamiento aplicando una función aleatoria que cambia la posición real del robot generando un error entre 0 y 25 cm alrededor de cada punto. El porcentaje de error fue calculado en comparación al error cero o posición ideal. El objetivo era medir el tiempo en alcanzar el punto según la magnitud del error. La Tabla III muestra el promedio de los resultados de cinco mediciones para cada prueba.

La Tabla III muestra el retardo en tiempo para alcanzar un punto destino a partir de la oscilación alrededor del punto, según el error radial de posiciónamiento. Además, el tiempo aumenta bastante para un error mayor de 10 cm. Es decir, entre 0 y 10 cm de error de posicionamiento, el tiempo de retardo para encontrar el punto buscado solo implica alrededor de 7 segundos adicionales como máximo, mientras entre 10 y 25 cm de error de posicionamiento, significan retardos entre los 8 y los 40 segundos que se convierten en un error apreciable en cada punto. Eso permite ver que el error aceptable no puede ser mayor a 10 cm en la práctica; para corregir este error mediante la precisión dada por el sensor de posicionamiento de manera suficientemente confiable.

En la Tabla III, el rendimiento en cada caso, está calculado tomando como base el 100% cuando hay cero error de posicionamiento; teniendo en cuenta la disminución de eficiencia obtenida por el retardo de tiempo para encontrar el punto buscado, oscilando el carro alrededor de él; según el error radial de posicionamiento.

TABLA III.
SIMULACION DE PRECISION DE POSICIONAMIENTO MEDIANTE EL ERROR RADIAL.

ERROR (cm)	DIJKSTRA (s)	Rend. (%)	FLOYD (s)	Rend. (%)	A* (s)	Rend. (%)
0	27,56	100	27,51	100	27,44	100
5	27,93	98,67	27,88	98,67	27,83	98,59
10	35,36	77,94	33,75	81,51	34,48	79,58
15	51,8	53,20	44,47	61,86	38,23	71,77
20	52,28	52,71	55,46	49,60	50,23	54,62
25	73,85	37,31	79,37	34,66	77,89	35,22

Tabla 3. Tiempos de ejecución y rendimiento de los algoritmos variando el error radial de posicionamiento en cm, para medir el tiempo de retardo en llegar a la posición, según el grado de precisión.

2.5. **Prototipo de Robot a escala**

Para la validación funcional fue construido un robot a escala con recursos LEGO que fuese compacto, eficiente y requiriese la menor cantidad de sensores posible. Fue construido a partir del kit educacional LEGO Mindstorms NXT 2.0 con un sensor brújula Hitechnic (sensor de compás magnético) y una cámara NXT Cam-v3. [25].

La Fig. 7 muestra El sensor tipo brújula que mide el campo magnético de la tierra y devuelve un valor entre 0 y 359 (izquierda) y la cámara NXTCam-v3, que posee un subsistema de visión con procesamiento de la imagen en tiempo real, con capacidad de detectar y seguir líneas u objetos de hasta ocho colores diferentes.

2.6. **Navegación.**

Para la navegación por el escenario, se utilizaron dos sensores para orientar el robot: un sensor brújula y una cámara. La brújula es utilizada para hacer giros de 90º. Cuando se conoce la posición inicial, basta con girar los motores hasta que la lectura instantánea coincida con la final deseada. Para eso fue utilizado un control Proporcional (P) para minimizar los errores de posición.

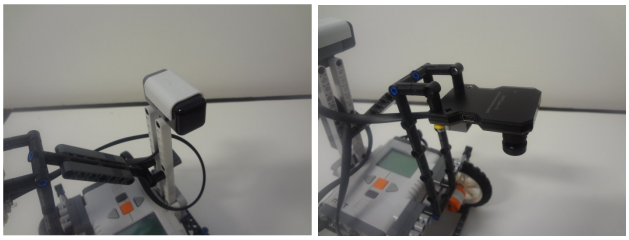

Figura 7. Sensor tipo brújula Hitechnic (izquierda) y la Cámara NXTCam-v3 (derecha).

La Fig. 8 muestra el robot completo, con el sistema de "rueda loca" utilizado como tercer y cuarto punto de apoyo del robot.

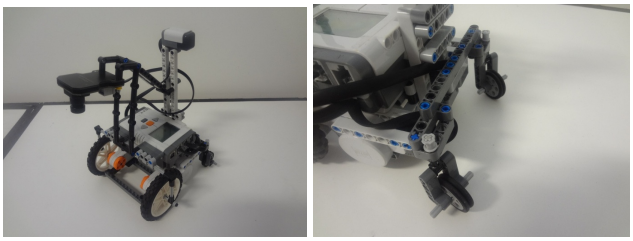

Figura 8. Robot LEGO utilizado con el sistema de rueda loca.

Este controlador fue desarrollado usando la lectura del sensor compas. El error es la diferencia entre la posición buscada y la lectura actual; estableciendo la fórmula de corrección del controlador como:

$$CORRECCIÓN=K_P*ERROR$$

El valor de la corrección es sumado o restado de la potencia de los motores para corregir el posicionamiento, con una constante K_P de 0.3333. Los motores giran en sentidos opuestos para que el robot realice la curva girando sobre sí mismo. La cámara es usada para guiar el robot mediante los marcadores existentes en el escenario. La cámara distingue el marcador como un rectángulo y retorna las coordenadas A,B,C,D respecto a dos bordes básicos de los límites del terrenos en direcciones longitudinal y transversal; para dimensionar el recuadro de posicionamiento de la cámara conforme se muestra en la Fig. 9.

Por medio de las distancias mostradas por la cámara, es posible calcular las distancias al centro del rectángulo hasta el borde de referencia de la izquierda. Siendo así, es posible comparar la posición instantánea del centro del rectángulo con la posición ideal de ese cuadrado. La posición ideal es cuando el robot está posicionado perfectamente alineado con la plantación, o sea, andando en línea recta. En ese código fue usado un controlador PID (Proporcional-Integral-Derivativo) para minimizar los errores de trayectoria, el cual fue desarrollado usando la lectura de la cámara (NXT CAM). La cámara retorna los valores de "A", "B", "C" y "D" mostrados en la Fig. 9, con los cuales es calculado el centro del cuadrado de la Fig. 9. El error en el eje X es la diferencia entre la posición buscada (centro del recuadro de la figura) y la posición actual del cuadrado dentro del recuadro, aplicando la corrrección mediante la fórmula:

$$CORR=K_P*ERROR+K_I*INTEGRAL+K_D*DERIVATIVO$$

Donde:

$$INTEGRAL = INTEGRAL + ERROR$$

y $$DERIVATIVO = ERROR - LASTERROR$$

El valor de la corrección es sumado o restado de la potencia de los motores con las siguientes constantes: $K_P=5.0$; $K_I=0.1$ y $K_D=10.0$. Además de guiar el robot por el escenario, la cámara también es utilizada para contar cuantos marcadores fueron recorridos durante el desplazamiento.

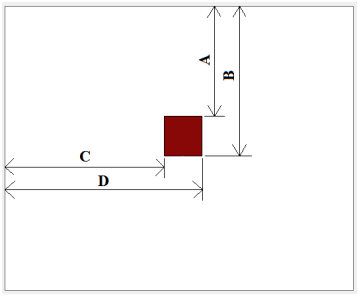

Figura 9. Ubicación del marcador por la cámara.

Para que el robot pueda seguir una trayectoria especifica, fue desarrollado un código que, sabiendo la ruta que el robot debe seguir en el escenario, y sus posiciones y orientaciones inicial y final, haga que el robot ande cuantos marcadores sean necesarios y siga la cantidad de curvas designadas para que el robot llegue al destino en el escenario. De esta forma, para aplicación real, es representado el "layout" de la plantación, el punto de partida y de llegada y la presencia de obstáculos. Con estas informaciones se calcula la ruta que el robot deberá seguir. Las Figs. 11 a 13 muestran paso a paso las etapas de la prueba experimental que el robot debe seguir para ir de la posición inicial y llegar al destino en función de la Fig. 10.

Para seguir la ruta presentada en la Fig. 10, el robot deberá andar a partir de la posición inicial, cuatro marcadores en línea recta, girar 90º a la derecha, andar otros cuatro marcadores en línea recta, girar nuevamente 90º a la derecha y andar los 16 marcadores restantes en línea recta para llegar a su destino. En la aplicación real en plantaciones de papa en lugar de los marcadores será utilizada la propia planta de papa.

Para desarrollar el código, se dividió el escenario en una matriz donde cada posición tiene dos coordenadas, y el robot puede estar orientado de cuatro modos: Norte, Sur, Este u Oeste. Así, sabiendo las coordenadas y orientaciones de donde está el robot y donde desea ir; basta seguir las etapas conforme a la sección 2.6.

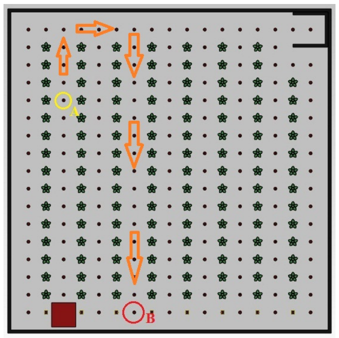
Figura 10. Ejemplo de ruta que el robot deberá seguir.

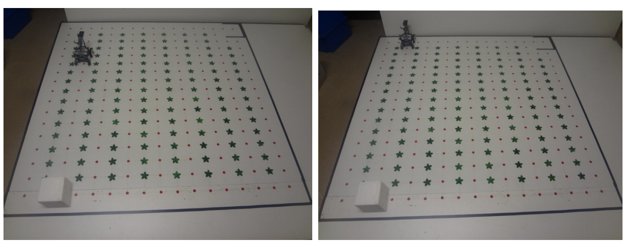
Figura 11. Robot en la posición inicial (izquierda) y avanza cuatro marcadores (derecha).

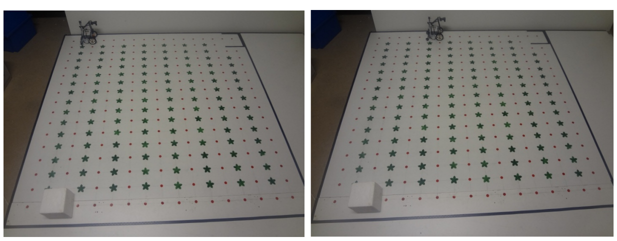
Figura 12. El robot gira 90º a la derecha (izquierda) y anda cuatro marcadores nuevamente (derecha).

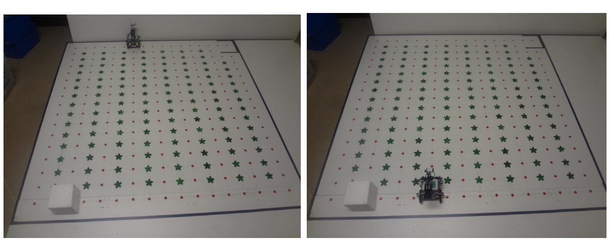
Figura 13. El robot gira 90º a la derecha nuevamente (izquierda) y avanza otros 16 marcadores para llegar al destino (derecha).

2.6. **Generación de trayectoria – Algoritmo A***

Ya fue mostrado como el robot hace para llegar a un punto destino siguiendo una ruta en las Figs10 a 13. Sin embargo, se debe determinar cómo es calculada esta ruta, sabiendo donde se encuentra el robot y a dónde quiere llegar. Para calcular el camino óptimo, o sea, el menor camino entre los dos puntos, fue utilizada una adaptación del algoritmo A* [26].

El algoritmo A* fue desarrollado con base en el algoritmo de Dijkstra. Como su antecesor, es basado en el uso de un gráfico ponderado, en el cual, a través del uso de etiquetas –o "frames"- en los nodos, es posible encontrar la ruta con menor costo desde un punto de partida al punto final. Este algoritmo, sin embargo, incorpora información heurística para encontrar de forma eficiente el camino deseado. La Fig. 14, muestra un diagrama de flujo de cómo funciona el algoritmo A*.

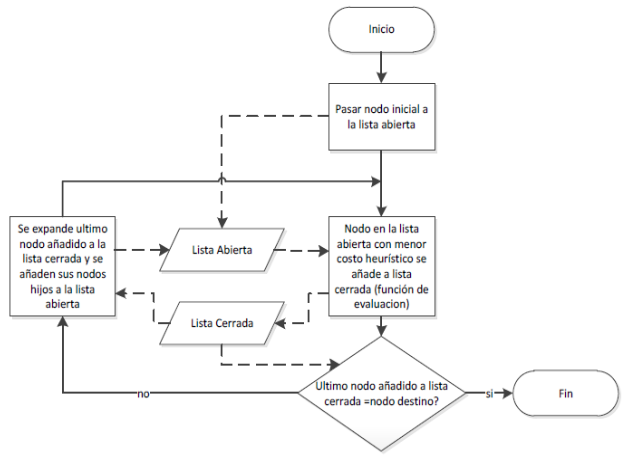

Figura 14. Diagrama de flujo del algoritmo A*.

Para el caso de la plantación de papa, el escenario fue tratado como una matriz de 17 filas por 17 columnas, de acuerdo con la Fig. 15. En la matriz de la Fig. 15, el número 1 representa un punto donde está una planta de papa, el número 0 representa los marcadores o camino por donde se puede circular, el número 6 representa el obstáculo que puede ser colocado en cualquier posición que se desee apenas alterando su posición en el programa, el número 2 representa la pared del local de recarga de batería del robot, el número 4 indica la posición inicial del robot y el número 5 indica el destino.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	2	2
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	0	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	4	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0
0	0	6	0	0	0	0	0	0	0	5	0	0	0	0	0	0

Figura 15. Matriz que representa el escenario.

Con la ejecución del algoritmo A*, se obtiene la secuencia de puntos o camino de menor costo para ir de 4 a 5; como el representado en la Fig. 16. En esta figura, el número 3 representa los marcadores por donde el robot debe pasar. O sea, el camino óptimo. Es posible observar en la figura que, en caso

de que no hubiese obstáculo, la menor ruta sería ir por la parte inferior de la matriz, pero como el camino de abajo está obstruido, la ruta de menor costo es la de la Fig. 16.

0	0	3	3	3	3	3	3	3	3	3	0	0	0	0	0	0
0	1	3	1	0	1	0	1	0	1	3	1	0	1	0	2	2
0	1	3	1	0	1	0	1	0	1	3	1	0	1	0	0	0
0	1	3	1	0	1	0	1	0	1	3	1	0	1	0	1	0
0	1	4	1	0	1	0	1	0	1	3	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	3	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	3	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	3	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	3	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	3	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	3	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	3	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	3	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	3	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	3	1	0	1	0	1	0
0	1	0	1	0	1	0	1	0	1	3	1	0	1	0	1	0
0	0	6	0	0	0	0	0	0	0	5	0	0	0	0	0	0

Figura 16. Matriz con la ruta calculada.

III. CONCLUSIONES

El posicionamiento del robot en el espacio es importante según la distribución de las plantas. Eso depende de las características específicas del cultivo, con dimensiones variables entre plantas y entre filas, requiriendo de una precisión menor de 10 cm de posicionamiento. Esta precisión se puede obtener con sensórica y/o tecnología GPS.

Los algoritmos probados, entregan la ruta más corta entre dos puntos. Sin embargo, el algoritmo de Floyd, no consigue calcular rutas para gran cantidad de puntos siendo el máximo para un área de 10m por 10 m aproximadamente.

El algoritmo A* mostró ser mejor que el algoritmo de Dijkstra con mayor diferencia de tiempo especialmente, cuando son muchos puntos. Sin embargo, el algoritmo de Dijkstra puede calcular diferentes rutas a diferentes puntos, desde una posición inicial; siendo mejor para rutas multidestino.

El proceso de carga de datos tarda mucho tiempo para grandes cantidades de puntos o cultivos de mayor tamaño; siendo mejor para aplicación en cultivos pequeños y medianos de cómo se pretendía en este estudio.

Para cultivos mayores se puede aplicar el algoritmo A* disminuyendo la cantidad de puntos de posición por estaciones de dosificación con múltiples posiciones por parada. Es decir, para múltiples plantas en cada parada.

La simulación mostró la viabilidad de uso del algoritmo A* con y sin obstáculos, en la trayectoria según la distribución del cultivo.

No existe una diferencia importante entre los tres algoritmos con condiciones similares, cuando se trata de pequeña cantidad de puntos de posicionamiento para áreas menores a 10 x 10 m.

En este trabajo es presentado un prototipo de robot para ser aplicado en plantaciones de papa. Con los resultados obtenidos en el desarrollo de este robot, se hace evidente que es perfectamente posible construir un robot para navegar en forma apropiada y autónoma en una plantación sin dañar el cultivo, sea de papa o de cualquier otro cultivo, sea para aplicar fertilizantes, para irrigación, recolección o cualquier otra actividad que pueda desempeñar un robot móvil autónomo.

RECOMENDACIONES

En un ambiente simulado, los resultados fueron convincentes de la viabilidad de funcionamiento de los códigos de programación. Como actividades futuras están previstas la utilización de un GPS para navegación, la construcción de un prototipo real y ensayos en plantaciones reales de papa.

Los autores agradecen a los ingenieros Antonio Tumialán, Alex García y Christian López por su ayuda con la aplicación programada en JAVA.

REFERENCIAS

[1]. S. Blackmore. “Agricultura de precisión – AP”. *Revista Nacional de Agricultura.* Colombia. No. 949, pp. 20-28. Junio 2007.

[2]. C.D Christy. “Real-time measurement of soils attributes using on-the-go near infrared reflectance spectroscopy”. *Computers and Electronics in Agriculture.*Science Direct. No. 61. pp.10-19. 2008.

[3]. T. Bakker,.H. Wouters, K. van Asselt, J. Bontsema, L. Tang, J. Muller and G. van Straten. “A Vision based row detection system for sugar beet”. *Computers and Electronics in Agriculture.* Science Direct. No. 60, pp. 87-95. 2008.

[4]. L. García-Pérez, M. García-Alegre and A. Ribeiro. “An agent of behavior architecture for unmanned control of a farming vehicle”. *Computers and Electronics in Agriculture.* Science Direct. No. 60. 2008. pp. 39-48.

[5]. S. Gan-Mor, R. L. Clark, B.L. Upchurch. “Implement lateral position accuracy under RTK-GPS tractor guidance”. *Computers and Electronics in Agriculture.* Science Direct. No. 59. p.p. 31-38. 2007.

[6]. S. De Bruin, G.B.M Heuvelink, and J.D. Brown. “Propagation of positional measurement errors to agricultural field boundaries and associated costs”. *Computers and Electronics in Agriculture.* Science Direct. No. 63. p.p. 245-256. 2008.

[7]. D.C. Slaughter, D. K. Giles and D. Downey “Autonomous robotic weed control systems: A review”. *Computers and Electronics in Agriculture.* Science Direct. No. 61. p.p. 63-78. 2008.

[8]. F. Kumhála, M. Kroulík and V. Prosek. "Development and evaluation of forage yield measure sensors in a moving-conditioning machine". *Computers and Electronics in Agriculture*. Science Direct. No. 58. pp. 154-163. 2007.

[9]. I. Santé-Rivera, R. Crecente-Maseda and D. Mirnada_Barrós. "GIS-based planning support system for rural land-use allocation". *Computers and Electronics in Agriculture*. Science Direct. No. 63. pp. 257-273. 2008.

[10]. M.I. Ramos, A.J. Gil, F.R. Feito, and A. García-Ferrer. "Using GPS and GIS tools to monitor olive tree movements". *Computers and Electronics in Agriculture*. Science Direct. No. 57. pp. 135-148. 2007.

[11]. V. Leemans and M. Destain, "A computer-vision based precision seed drill guidance assistance". *Computers and Electronics in Agriculture*. Science Direct. No.59. pp. 1-12. 2007.

[12]. R. Jorgensen, M. Norremark, C.G. Sorensen and N.A. Andersen. "Utilising scripting language for unmanned and automated guided vehicles operating within row crops". *Computers and Electronics in Agriculture*. Science Direct. No. 62. pp. 190-203. 2008.

[13]. H. Yong M. Huang, A. García, A. Hernández and H. Song. "Prediction of soil macronutrients content using near-infrared spectroscopy". *Computers and Electronics in Agriculture*. Science Direct. No. 58. pp. 144-153. 2007.

[14]. P. Martín, O.S. Hernández, N.M. Arias, W. Amaya, y J.E. Rangel. "Diseño de un sistema de dosificación para distribución inteligente de fertilizantes". Proceedings of the 2nd International Symposium on Innovation and Technology ISIT 2011. International Institute of Innovation and Technology E.I.R.L. (IIITEC), pp.113 - 119. 2011.

[15]. G. Vellidis, M. Tucker, C. Perry, C. Kvien and C. Bednarz. "A real time wireless smart sensor array for scheduling irrigation". *Computers and Electronics in Agriculture*. Science Direct. No 61. pp.44-50. 2008.

[16]. D. M. Bulanon, H. Okamoto and S. Hata. "Feedback control of manipulator using machine vision for robotic apple harvesting". ASABE. Annual International Meeting, 17 – 20. Tampa, Florida. Paper No. 053114. July, 2005.

[17]. C. Amiama, J. Bueno, C.J. Alvarez and J.M. Pereira. "Design and field test on automatic data acquisition system in a self-propelled forage harvester". *Computers and Electronics in Agriculture*. Science Direct. No. 61. pp. 192-200. 2008.

[18]. M. Loghavi, and B.Behzadi M. "Development of a target oriented weed control system". *Computers and Electronics in Agriculture*. Science Direct. No. 63. pp.112-118. 2008.

[19]. B. Âstrand, and A. Baerveldt. "An Agricultural Mobile Robot with Vision-based Perception for Mechanical Weed Control". *Autonomous Robots*. Kluwer Academic Publishers. Netherlands. No. 13. pp. 21-35. 2002.

[20]. O. I. Caldas, D.J. Monroy, D.M. Espitia, M.A. García and R.A. Castillo. "Aproximación de una Planeación Óptima de la Trayectoria para un Robot Móvil". Colombia, 4th IEEE Colombiano Workshop on Robotics and Automation Ponencia: Aproximación de una planeación óptima de la trayectoria para un robot móvil. 2008.

[21]. M. O'Connor, T. Bell, G. Elkaim and Dr. B. Parkinson. "Automatic Steering of Farm Vehicles Using GPS" [On Line]. Stanford University. Available: http://biblioteca.universia.net/html_bura/ficha/params/title/automatic-steering-of-farm-vehicles-using-gps/id/45826085.html.pdf. 1996.

[22]. G. Pajares, J.J Ruz, P. Lanillos, M. Guijarro, J.M de la Cruz and M. Santos. "Generación de trayectorias y toma de decisiones para UAV's". Revista iberoamericana de automática e informática Industrial. Vol. 5. No. 1 pp. 83-92. 2008.

[23]. J. A. Marín, M.A. Zamora and H. Martínez. "Planificación de trayectorias en un mapa de celdillas difusas". Universidad de Murcia. 2003.

[24]. R. Borrego, D. Recio. "Manual de Algorítmica". Escuela Técnica superior de Ingeniería Informática. Depto. De Matemáticas Aplicadas I. Proyecto de fin de carrera. User. [Online]. Available: http://forja.rediris.es/frs/download.php/87/AlgoritmosDeGrafos.pdf

[25]. NXTCam-v3 User Guide, Mindsensors. [Online]. Available: http://www.mindsensors.com/index.php?module=documents&JAS_DocumentManager_op=downloadFile&JAS_File_id=988

[26]. P. Lester "A* Pathfinding para Iniciantes". [Online].
Available:
http://www.policyalmanac.org/games/aStarTutorial_port.htm