Conciencia Tecnológica No. 34, Julio - Diciembre 2007

Control De Un Vehículo Guiado Automáticamente (AGV)

Investigación

M. C. Alejandro Pedroza Reyes[1], Dr. Carlos Sánchez López[2], M. C. Héctor Ulises Rodríguez Marmolejo[1].
(1) Departamento de Ingeniería Eléctrica – Electrónica, (2) Departamento de Metal – Mecánica.
Instituto Tecnológico de Aguascalientes. Av. Adolfo López Mateos No. 1801 Ote. Fracc. Bona Gens.
Aguascalientes, Ags. Tel. (449) 910-5002, Fax (449) 970-0423
Correo electrónico: alexpedroza_2000@yahoo.com, carlossl@ita.mx

Resumen

El presente trabajo tiene como objetivo principal el aporte de un algoritmo de cálculo de trayectorias para un vehículo autónomo guiado. El cálculo se hace partiendo de parámetros capturados en un computador, los cuales son enviados al sistema de control para calcular una ruta libre de colisiones, que aproveche el espacio disponible, para proveer de movimientos suaves al vehículo. El algoritmo desarrollado está basado en la construcción de curvas paramétricas tipo Hermite y su implementación se llevó a cabo en un microcontrolador dsPIC30F en lenguaje C modificado. Además se ha realizado un prototipo a escala, con el cual se realizaron pruebas que permitieron dar fe del buen funcionamiento del algoritmo, aún y cuando es susceptible de muchas mejoras, entre ellas, el desarrollo de un prototipo más eficiente o una comunicación inalámbrica, para mejorar las condiciones de simulación.

Palabras clave

Algoritmo, trayectoria, vehículo autónomo, control, Hermite.

Abstract

The intent of this work is to contribute with an algorithm to calculate an autonomous guided vehicle trajectory. The calculation method starts when the parameters are set in a software to be process and sent to a control system in order to calculate a collisions-free paths, also this algorithm should approach the available space and to provide vehicle softly movements.

The developed algorithm is based on the construction of Hermite curves and its implementation was carried out with a microcontroller dsPIC30F programmed in C language modified. In addition to this, a scale prototype was developed in order to test the algorithm performance; the result was satisfactory however some improvements are required as a wireless communication and a better prototype to up grade simulate condition.

Keywords

Algorithm, path, autonomous vehicle, control, Hermite.

Introducción

El trabajo que se presenta a continuación busca dar a conocer los resultados obtenidos dentro del Instituto Tecnológico de Aguascalientes por el grupo de investigación atendido por el cuerpo académico "Automatización, Robótica y Sistemas Mecatrónicos". Una de las problemáticas que se están abordando es el desarrollo de sistemas con desplazamiento autónomo, lo cual dio la pauta para la realización del presente trabajo.

En la actualidad los vehículos guiados automáticamente (AGV) son muy utilizados en la industria, principalmente en los países desarrollados, con la finalidad de sustituir a las personas en labores que representan algún riesgo o bien en lugares donde se requiere de precisión y continuidad en el acomodo dentro de almacenes. Entre las tareas más importantes que puede llevar a cabo un AGV se encuentran: navegación y guiado, cálculo de rutas, administración de tráfico, transferencia de carga, entre otras.

Fundamentos Teóricos

Cálculo de Trayectorias.

El cálculo de trayectorias se hace en base a la construcción de curvas suaves para facilitar el movimiento, las curvas son definidas mediante parámetros sencillos con la finalidad de que éstas puedan describir una trayectoria sin demasiadas restricciones.

En base a esto se investigó sobre las diferentes formas de generar curvas. Existen básicamente dos formas de construir una curva: como funciones de alguna variable como x, y y z o como funciones de un parámetro como lo puede ser t. La primera forma implica algunas dificultades como el caso de una

pendiente infinita para algún punto en particular. Sin embargo la representación paramétrica de curvas elimina este problema ya que reemplaza el uso de las pendientes por vectores tangentes (los cuales nunca pueden ser infinitos) [1].

En base a esto se eligió la segunda forma para la construcción de la trayectoria. El objetivo es construir una curva con una serie de segmentos de curva como se muestra en la figura 1. El punto en el que se juntan los dos segmentos éstos y sus tangentes son iguales

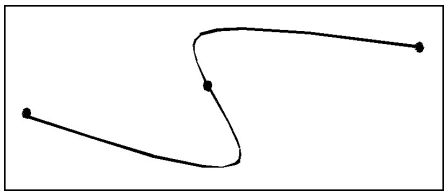

Figura 1. Curva paramétrica.

Existen muchas formas de definir una curva paramétrica, una de las más comunes es la Hermite [2], que requiere, para ser definida de los puntos que limitan el segmento de curva y dos vectores que parten del punto inicial y final, los cuales indican la dirección en la cual la curva debe iniciar.

El procesador digital de señales dsPIC30F4011

El procesamiento matemático para el cálculo de trayectorias involucra el manejo de matrices. Para el desarrollo del algoritmo de cálculo de la trayectoria se investigaron algunos tipos de microcontroladores y a partir de esta investigación se concluyó utilizar una de las más recientes creaciones de la empresa Microchip©, un dispositivo con una maquinaria DSP que tiene un costo muy bajo en comparación con otros de características similares. Otra de las ventajas que presentó fue que está diseñado para el control de motores, con hasta 6 canales de PWM y una interfaz de cuadratura de encoder.

Además Microchip provee de la utilería C30 dentro del ambiente de desarrollo MPLAB que permite realizar la programación en lenguaje C, lo cual facilitó el desarrollo. Además tiene una estructura DSP, el cual consiste de un multiplicador de alta velocidad de 17 bitios, un registro de corrimiento y un sumador/ restador de 40 bitios [3].

Software de desarrollo MPLAB C30

MPLAB C30 es un compilador optimizado que traduce los programas de código C en código ensamblador. Este compilador también soporta muchas opciones de líneas de comando y extensiones de lenguaje que permiten acceso total a las capacidades del dispositivo dsPIC.

El compilador utiliza un conjunto de pasos para generar código en C que incluyen optimizaciones de alto nivel que toman las ventajas de las características particulares de la arquitectura del dsPIC. MPLAB C30 contiene una librería estándar completa de ANSI C. Todas las funciones de la librería están validadas conforme a la librería estándar ANSI C [4].

Materiales y métodos

Generación de curvas paramétricas.

Estas curvas, como ya se mencionó tienen la ventaja de que son suaves y en el caso de las tipo Hermite parten de un punto y orientación inicial y terminan exactamente de la misma forma. Para comenzar a analizar lo conveniente de las curvas paramétricas se realizó un programa en MatLab© para generar algunas curvas Hermite. Se generaron 4 curvas (figura 2), donde el punto inicial tiene coordenadas (0, 0), el punto final (0, 30), el vector de inicio tiene un ángulo de 45° y el final de -90°. Sólo la longitud de los vectores (se indica con un número sobre cada curva) fue modificada para obtener diferentes curvas.

Como puede verse conforme el vector de salida y llegada, es más grande, la curva tiene una extensión más amplia, lo cual hace que sea más suave, sin embargo si el tamaño del vector es excesivo se producen oscilaciones.

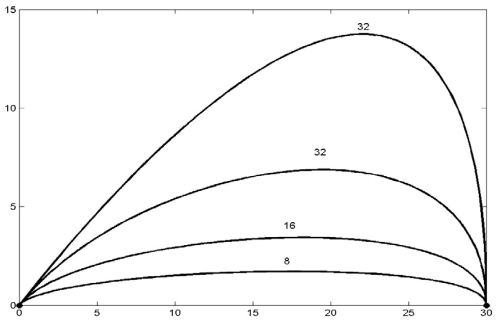

Figura 2. Conjunto de curvas hermite con diferente longitud de vectores.

Como la curva Hermite se encuentra en función del parámetro t que varía de 0 a 1, éste fue dividido en proporción directa con la distancia a recorrer, entonces la cantidad de puntos generados para la trayectoria es:

Puntos generados=100 x dist. Total (1)

O sea que habrá 100 puntos por cada metro, aproximadamente 1 cada cm. Ahora bien, el parámetro t estará dividido en relación inversa con la cantidad de puntos que se requieran calcular, o sea que tendrá incrementos en cada iteración de:

Incremento de t = 1/(puntos generados) (2)

Una característica importante de este tipo de curvas es que los puntos calculados están más juntos en tanto la curva sea más cerrada, por lo que estos puntos no estarán espaciados de manera uniforme. Esta característica ayuda a que el vehículo describa la trayectoria de una manera más perfecta, ya que en segmentos donde la curvatura sea mínima o nula no hay necesidad de tener demasiados puntos ya que la dirección se conserva.

El cálculo de la curva Hermite que ha de servir de trayectoria se calcula de la siguiente manera:

1. Almacenamiento de datos. Dentro de una matriz de 4 x 4 son almacenados los valores de la matriz Hermite (*A*). Las coordenadas de inicio y final de la trayectoria, y los ángulos de salida y llegada enviados por la interfaz humana son almacenados en un vector de datos de 6 elementos (*ent[6]*).

2. Cálculo de valores previos. Se calcula la distancia entre los puntos para obtener el factor de separación entre los puntos, se calcula además la magnitud de los vectores en función de la distancia que se va a recorrer. Se calcula y asignan los valores de las matrices columna *Dx* y *Dy*, las cuales resultan de la multiplicación de las matrices columna (*Hx* y *Hy*) que contienen las coordenadas de x y y separadamente de los puntos inicial y final y las coordenadas de los vectores y la matriz Hermite (*A*).

3. Cálculo de los puntos que conforman la curva. Para este procedimiento se realiza un ciclo que se repite tantas veces como puntos se requieran, es decir 100 veces la distancia que separa a los puntos medida en metros. Este ciclo empieza con $t = 0$; calculando los valores para x y y al sumar el valor previo de estas coordenadas con la multiplicación entre el vector de potencias de *t*, es decir *T* con las matrices columna *Dx* y *Dy*.

Creación del ambiente.

El ambiente en el que se desarrolla el vehículo está descrito mediante el primer cuadrante de un sistema de ejes coordenados, de esta manera tenemos que en ningún momento una coordenada puede tener valores negativos, de tenerlos entonces se tendría una incursión a un área no permitida o fuera del ambiente. Para definir el límite superior del área se ha establecido una valor máximo para la coordenada *y*. A su vez para definir el límite derecho está establecido un valor máximo para el eje *x*.

Ahora bien un obstáculo puede ser definido mediante un conjunto de puntos ordenados en el sentido de las manecillas de reloj que representan sus vértices, estos puntos al estar ordenados pueden formar segmentos de recta entre pares adyacentes; estos segmentos formarán un poliedro convexo que representará al obstáculo. Un ejemplo de esto se muestra en la figura 3.

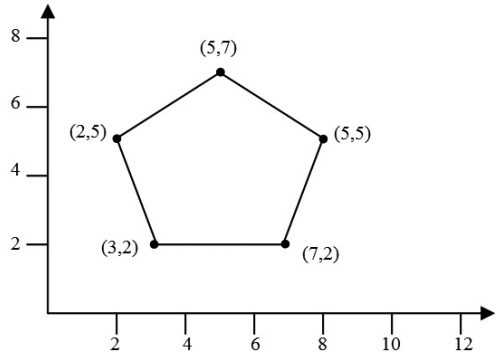

Figura 3. Definición de obstáculo.

Detección de incursión en áreas prohibidas.

Las áreas prohibidas son aquellas área que queden fuera del primer cuadrante del plano cartesiano o excedan los límites de los ejes *x* y *y*. El otro tipo de área prohibida son los obstáculos, aquellos poliedros que se encuentran dentro del área permitida pero que por sus características no pueden ser atravesados, llámese estantes, maquinaria o cualquier otro objeto con el cual pudiera colisionar el vehículo.

a) Fuera de ambiente.

En el primer caso se tiene una subrutina que se encarga de examinar si las coordenadas calculadas en cada iteración por la subrutina hermatrix se encuentran fuera del ambiente, para ello se revisa si son valores negativos o si son mayores del valor máximo para los ejes *x* y *y*. Esta subrutina entrega una bandera o variable tipo bit para indicar si los valores de *x* y *y* cumplen con las condiciones.

b) Colisión con obstáculos.

Para determinar si existen colisiones (figura 4) con obstáculos se ha elaborado una subrutina denominada "*test*", la cual toma los valores calculados en cada iteración por la subrutina "*hermatrix*" y los asigna al primer renglón de una matriz de 3 x 3 donde la tercer columna es unitaria.

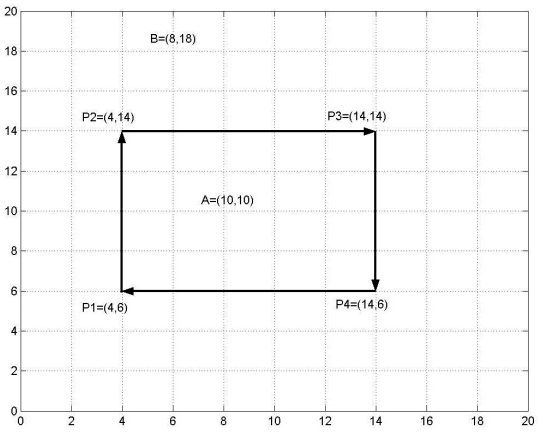

Figura 4. Determinación "dentro" o "fuera" de obstáculo.

Luego los dos primeros datos del segundo renglón están conformados por las coordenadas del primer punto del primer obstáculo y los dos primeros datos del tercer renglón corresponden a las coordenadas del segundo vértice del mismo obstáculo. El signo del determinante de esta matriz indica a que lado se encuentra el punto evaluado del vector. El punto se encontrará "dentro" si en todas las evaluaciones el determinante es negativo.

Evasión de obstáculos.

El primer cálculo de trayectoria se realiza suponiendo que se puede llegar al destino sin bordear obstáculos, sin embargo se evalúa si cada coordenada cruza algún obstáculo; a esta trayectoria se le denominar primaria. En caso de haber colisiones con obstáculos, se almacena la información acerca de los obstáculos que han sido cruzados para posteriormente buscar salvarlos.

Una vez que se ha determinado que la trayectoria primaria tiene colisiones con obstáculos, el siguiente paso consiste en bordear aquel primer obstáculo que se encuentre estorbando. Para ello se ha diseñado una subrutina (figura 5) que tiene por tarea buscar cual de las esquinas del poliedro de que se trate puede servir como punto auxiliar para descomponer la trayectoria de tal manera que el vehículo vaya primeramente a ese punto y luego busque completar su recorrido.

La primera forma de conocer si aquel vértice puede servir de punto de apoyo para bordear al obstáculo es determinar si para llegar a él no existe intersección con el mismo obstáculo o con otro.

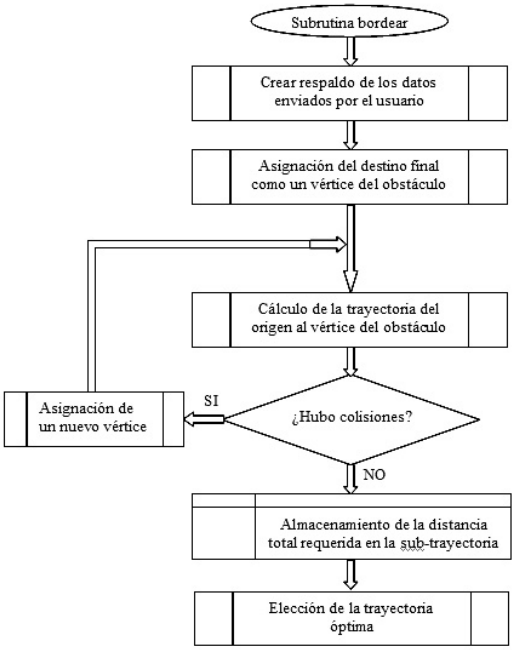

Figura 5. Subrutina Bordear.

En la figura 6 se muestra un caso de colisión en el que el vehículo parte del punto *(2,2)* y debe llegar al punto *(8,10)*. Al calcular la trayectoria primaria se detecta que el obstáculo 1 está estorbando por lo que hay que bordearlo, para lo cual se calculan sub-trayectorias hacia cada uno de los vértices del obstáculo con lo cual se obtienen las trayectorias a, b, c y d. De estas trayectorias sólo c colisiona con el obstáculo por lo cual es desechada instantáneamente; luego de las restantes se observa que aunque b y d no colisionan en su etapa inicial con el obstáculo, sí lo hacen en su etapa complementaria *b'* y *d'* por lo que también deben ser eliminadas.

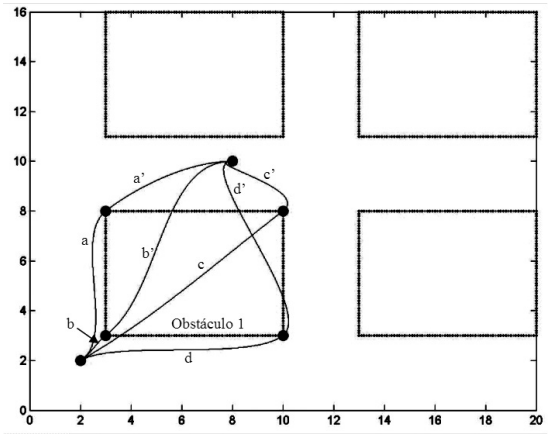

Figura 6. Posibles trayectorias para bordear un obstáculo

Así pues sólo queda la trayectoria *a-a'*. En caso de existir dos o más trayectorias factibles, en cada caso se calcula la distancia total que habrá de recorrerse y se guarda para su posterior análisis de distancia total. Luego se calcula el resto de la trayectoria, nuevamente evaluando si existen colisiones con algún obstáculo, de haberlas se vuelve a descomponer la trayectoria buscando bordear por los vértices del obstáculo que interfiere. Este procedimiento se realiza hasta que la trayectoria sea completada con éxito.

Implementación.

Para probar el funcionamiento del algoritmo y comprobar que la trayectoria elegida es realizable se construyó un vehículo de tres ruedas de 45 cm. de largo, 25 cm. de ancho por 20 cm. de alto. El par de ruedas traseras tienen un movimiento libre, mientras que en la rueda delantera se concentra el control de los movimientos con dos motores. Este diseño pretende sólo la implementación y en ningún momento se buscó que fuera un prototipo único o con un alto grado de eficiencia. Este prototipo tiene como única finalidad mostrar que el algoritmo funciona y el desarrollo de un prototipo mas especializado se deja como trabajo futuro.

La flexibilidad que se tiene en el cálculo de la trayectoria óptima es tal que podría utilizarse casi cualquier tipo de vehículo, ya que puede controlarse de manera muy simple la forma de la curva resultante, puede contener giros muy cerrados o muy amplios con solo cambiar el parámetro de magnitud de los vectores de arranque y llegada.

La fuente principal de energía para el funcionamiento de este prototipo consta de dos baterías de 12 Voltios conectadas en serie para proveer de 24 Voltios al circuito de control que a su vez proporciona este voltaje al motor de tracción, el cual está fabricado por la empresa Nisca®.

El motor de tracción tiene acoplado al eje un encoder para permitir conocer la posición del mismo a través de 3 sensores ópticos que envían señales al *dsPIC*. Asimismo la velocidad del motor de tracción es controlada mediante uno de los canales de *PWM* con que cuenta este dispositivo. La señal de *PWM* está conectada a un circuito *"puente H*" construido con transistores *MOSFET*.

En la figura 7 se muestra un diagrama a bloques de la conexión electrónica de los dispositivos del prototipo.

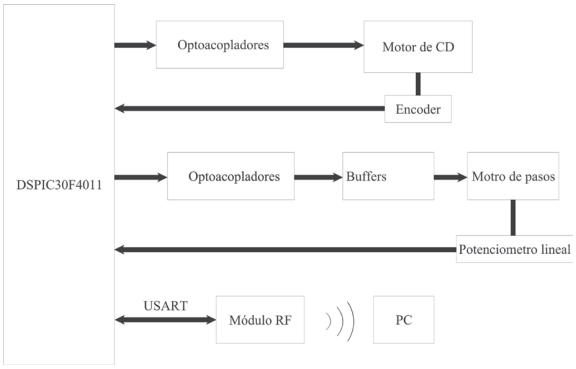

Figura 7 Diagrama a bloques del sistema.

Resultados y discusión

Se obtiene un algoritmo que permite crear trayectorias mediante el calculo de curvas Hermite (ver figura 1). Se identifica la necesidad de controlar los vectores de inicio y final de la curva para obtener trayectorias de diversas configuraciones (ver figura 2). Lo anterior se aprovecha para saber si la trayectoria interfecta con algún obstáculo y si así sucediera entonces cambiar a otra que lo evite. Vía programación, se implementó un espacio de trabajo en coordenadas positivas y se logra determinar la posibilidad de que el AGV incursione en áreas prohibidas, es decir fuera del espacio definido. Por otra parte, se establece un procedimiento para especificar obstáculos y se aplica un método para identificar el espacio dentro o fuera del obstáculo (ver figura 4). Lo anterior se implementó en un prototipo a escala, con tracción y dirección en una rueda delantera. Los resultados de las pruebas realizadas se consideran aceptables comparándolos con lo obtenido en teoría y en simulación. Se observó la gran flexibilidad del algoritmo ya que su funcionamiento no depende del diseño del AGV, lo cual a primer instancia permite que pueda implementarse en cualquier vehiculo de este tipo, potencialmente hablando, podría implementarse en cualquier sistema de desplazamiento donde se tenga tracción y dirección.

Conclusiones

El desarrollo de un proyecto de esta magnitud implica sin duda la incursión a una gran cantidad de campos de conocimiento, como el cálculo, la teoría de control, el desarrollo de software, la electrónica de potencia, entre muchas otras.

El presente trabajo pretende ser sólo una pequeña parte de un proyecto mucho más ambicioso para lo cual se requerirá una gran cantidad de tiempo adicional, e incluso ameritará una tesis doctoral.

Si bien se ha buscado llegar a la implementación mediante un prototipo físico, nunca se ha pensado en que sea un trabajo terminado, ya que el hecho de llevar a cabo la construcción de este modelo se ha realizado únicamente con la finalidad de probar que el algoritmo desarrollado funciona, sin embargo la construcción del AGV implicará un modelado matemático que permita conocer el desempeño del mismo bajo condiciones de perturbación, muy comunes en la industria, lo cual será un trabajo futuro muy importante.

La parte medular de este trabajo es el desarrollo del algoritmo en *MPLAB C30*© para un *dsPIC30F* capaz de calcular una trayectoria factible y óptima para ser desarrollada en un área limitada con obstáculos conocidos, lo cual se ha logrado con éxito, además se tiene como trabajo futuro el desarrollo de una mayor capacidad de flexibilidad en el ambiente de desenvolvimiento con la finalidad de que el AGV sea capaz de adaptarse a ambientes cambiantes, como por ejemplo un área en el que existan obstáculos móviles, los cuales bien podrían ser otros AGV's.

También va a ser necesario crear un ambiente virtual en el que el usuario pueda modificar una mayor cantidad de parámetros, como la velocidad máxima de ejecución, control del giro, entre otras.

Actualmente se trabaja en el análisis de los diferentes sistemas de localización que pudieran constituir una señal de realimentación y de esta manera garantizar el cumplimiento del recorrido planeado.

Referencias

[1] Foley, J. D. y Van Dam A. (1982) *Fundamentals of interactive computer graphics,* Addison Wesley. U. S.
[2] James L. Fuller. Robotics *Introduction, programming and projects,* Merrill, New York 1991.
[3] Microchip. dsPIC30F (2003) *Family Reference Manual*. www.microchip.com, acceso: abril de 2005.
[4] Microchip. *MPLAB® C30 C* (2005) *Compiler user's guide*. www.microchip.com, acceso: abril de 2005.

Artículo recibido: 4 de junio del 2007

Aceptado para publicación: 24 de octubre del 2007