TecnoLógicas

Navegación Robótica Basada en Patrones Estáticos Utilizando el Sistema Embebido CMUcam3

Robotic Navigation based in Statics Patterns using the CMUcam3 Embedded System

Víctor M. Jurado-Gutiérrez [1]
Juan S. Botero-Valencia[2]
Sergio I. Serna-Garcés[3]
Carlos A. Madrigal-González[4]

1 Instituto Tecnológico Metropolitano, Medellín-Colombia
victormjg17@hotmail.com
2 Instituto Tecnológico Metropolitano, Medellín-Colombia
juanbotero@itm.edu.co
3 Instituto Tecnológico Metropolitano, Medellín-Colombia
sergioserna@itm.edu.co
4 Instituto Tecnológico Metropolitano, Medellín-Colombia
carlosmadrigal@itm.edu.co

Tecno. Lógicas., ISSN 0123-7799, Edición Especial, octubre de 2013, pp. 617-629

Resumen

Este artículo presenta un método de navegación Robótico basado en patrones estáticos usando técnicas de visión artificial implementadas sobre el sistema embebido CMUcam3, además presenta un análisis de procesamiento de cómputo para este sistema embebido. El patrón a determinar es una línea negra con cruces, guiando al agente robótico a través de ésta. Luego de la adquisición de las imágenes mediante la CMUcam3, se utiliza un algoritmo de segmentación basado en umbral, se determina el esqueleto de los patrones y luego se aplica la transformada de Hough lineal para determinar, las líneas, los ángulos y los tipos de cruces. Debido a que la transformada de Hough lineal es un método que requiere alto procesamiento, en este trabajo, se limita el rango de los ángulos y se normaliza su espacio de acumulaciones. En los resultados obtenidos se pudo determinar que el método desarrollado para la navegación es preciso y confiable, ya que en un 87% de los segmentos de decisión se pudo determinar correctamente el tipo de cruce y el ángulo de corrección. Él sistema embebido CMUcam3 logró procesar una imagen entre 0.15 y 0.28 segundos, dependiendo del tipo de cruce que encuentre.

Palabras clave

Visión Artificial, CMUcam3, Transformada de Hough, Sistema embebido.

Abstract

This paper presents a method based on static patterns for robotic navigation using computer vision techniques implemented on CMUcam3 embedded system, also presents an analysis of computational complexity for this embedded system. The pattern to be determined is a black line with intersections, driving the robotic agent over the line. A segmentation algorithm based in threshold is used after the images acquisition by the CMUcam3, then defines the thinning patterns and then the linear Hough transform is applied to determine the lines, the angles and the type of the intersections. Due to the linear Hough transform is a method that requires high processing, in this project, the range of the angles are limited and the accumulation space is normalized. The results showthat the method developed for the navigation is accurate and reliable, because in 87% of the decision segments it could determine correctly the type of intersection and the correction angle. The CMUcam3 embedded system succeeded to process an image between 0.15 and 0.28 seconds, depending of the type of the intersection.

Keywords

Computer vision, CMUcam3, linear Hough transform, embedded system.

1. INTRODUCCIÓN

Una de las tareas más importantes en la robótica de servicios es la navegación autónoma, que permite a un robot desplazarse por sí mismo en un área, reconociendo patrones y ejecutando labores específicas, situación por la cual, la investigación en esta área ha sido ampliamente abordada debido a la versatilidad de las implementaciones que se pueden desarrollar en pro de la robótica. El problema de la navegación autónoma radica en la capacidad del sistema robótico en reconocer su posición actual y su capacidad de orientación, por lo que el uso de diferentes tipos de sensores como los odométricos, los de imágenes y los de ultrasonido han sido utilizados para recoger este tipo de información, implementando funciones como la localización del robot, la construcción de su entorno, la toma de decisiones para seguir la mejor ruta a su destino y la capacidad de evadir obstáculos.

Un sistema robótico de navegación puede adquirir diversas formas de movimiento siendo muy comunes el uso de ruedas como el sistema propuesto por (Bhave et al., 2003) o extremidades como es presentado en (Ikeda, 2010), dependiendo de las condiciones del terreno donde se va a desplazar y al tipo de tarea que enfrentará.

La navegación robótica implementada a partir de métodos de visión artificial ha alcanzado grandes resultados y ha sido mejorada considerablemente, gracias a la inteligencia concedida para desarrollar sus tareas. En el trabajo hecho por (Tall et al., 2009) se usó un método de rastreo de color y un algoritmo de navegación llamado ojo de camaleón, dándoles la capacidad a dos CMUcam3 de tomar una imagen desde muchos ángulos y así poder guiar un pequeño bote remolcador entre una serie de boyas ejecutando ciertas tareas mientras recorre su camino. Aproximadamente el 25% de los intentos resultaron exitosos para el caso en que se usaron todos los pares de boyas, mientras que un 75% de los intentos resultaron exitosos usando la mitad de los pares las boyas. En el proyecto de (Huang, 2008) se propone una técnica para la detección y reconocimiento automático de los signos circulares viales, el cual es muy importante en la investigación de los sistemas de asistencia al conductor en China. Éste usa procedimientos como la segmentación y la transformada de Hough para la detección del

signo y la distancia Hausdorff para el reconocimiento de dicha figura, obteniendo un 94.2% de eficacia para signos indicativos y un 85.4% para signos de prohibición. También está el desarrollo de una estrategia de control que es presentado por (Carelli, 2008), que permite a un robot seguidor rastrear un vehículo moviéndose por una trayectoria desconocida con una velocidad desconocida, usando solo la visión artificial para establecer la posición y la orientación relativa del robot al objetivo.

Los sistemas de procesamiento embebido en la robótica móvil han sido muy variados y dependen de la aplicación. En algunos casos se utilizan sistemas de procesamiento sobredimensionados que afectan los costos y la complejidad de los proyectos. Por tal motivo en este artículo se presenta el desarrollo de un sistema robótico de bajo costo implementando técnicas avanzadas de procesamiento de imágenes mediante una CMUcam3 (http://www.cmucam.org), además de un análisis de procesamiento para este sistema.

En la sección II se describe la metodología usada, dando a conocer el diseño de la plataforma robótica, el procesamiento de las imágenes con la CMUcam3 y el método de navegación. En la sección III se presentan los resultados obtenidos y un análisis al procesamiento de cómputo para las imágenes obtenidas. Y por último se presentan las conclusiones.

2. METODOLOGÍA

2.1 Plataforma Robótica

El sistema de locomoción del robot está compuesto por ruedas, debido a que el terreno para el cual fue diseñado es plano y su estructura está basada en el chasis para robot RRCC04A de Pololu. Usa dos ruedas con encoders y otras dos como apoyo. Se diseñó una tarjeta especial para este tipo de base con un microcontrolador Freescale de 32 bits de la familia ColdFire V1 MCF51JM128VLH, el cual proporciona los medios necesarios para mover los motores y mantener una comunicación serial con la CMUcam3 y comunica-

ción inalámbrica ZigBee. En la Fig. 1 se muestra la plataforma que fue diseñada.

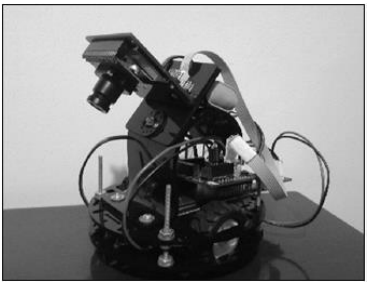

Fig. 1. Plataforma Robótica. Fuente: Autores

2.2 Adquisición de la Imagen

Usando las librerías CC3 que proporciona el fabricante en su sitio web, se inicializan algunos parámetros necesarios para la obtención de las imágenes con la CMUcam3. La resolución de las imágenes capturadas por la CMUcam3 puede ser de baja resolución o alta resolución y se pueden escoger entre los espacios de color RGB, HSV y YUV.

Para este proyecto se capturan las imágenes en 3 canales con una resolución baja de $143 * 88$, en formato YUV, mediante la función *cc3_pixbuf_load()*. Al realizar la captura en este formato, hay un ahorro de coste computacional ya que el canal Y obtiene la imagen en escala de grises, evitando así la operación en la imagen para pasar del formato RGB a escala de grises, Fig. 2a.

2.3 Segmentación

Después de obtener la imagen en escala de grises, se aplica un operador umbral t con valor de 60, el cual fue determinado mediante experimentación, tomando 100 imágenes en diferentes posiciones dentro de la ruta, la Fig. 2b muestra el resultado. Con la línea definida por la umbralización, se utiliza el algoritmo propuesto por (Zhang&Suen, 1984) para obtener el esqueleto del patrón detectado, este fue utilizado ya que tiene un coste computacional más bajo que los filtros para detectar bordes que común-

mente se usan en este tipo de aplicaciones. El algoritmo trabaja en paralelo entre dos sub-iteraciones, una va dirigida a la supresión de puntos limítrofes del sureste y los puntos de la esquina del noreste, mientras que la otra va dirigida a la supresión de puntos limítrofes del noroeste y los puntos de la esquina del sureste, Fig. 2c. Luego de esto, la imagen del patrón está lista para someterse a un algoritmo y extraer sus características que puedan comprobar que efectivamente hay una o dos líneas detectadas, es importante resaltar que no se trabajó con las 4 filas y 4 columnas iniciales y finales debido a que existen pixeles que sobran y no pertenecen a la línea.

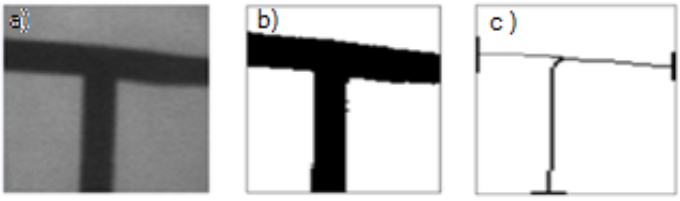

Fig. 2, a) componente Y de la imagen capturada. b) imagen umbralizada. c) imagen esqueletizada. Fuente: Autores

El método utilizado para la localización de líneas está basado en la transformada de Hough lineal (Duda et al., 1972), la cual también es usada para detectar diferentes formas en una imagen, haciendo posible la detección de círculos y elipses con modificaciones en el algoritmo. El objetivo del algoritmo es encontrar puntos alineados que puedan existir en la imagen, haciendo una transformación entre el plano de la imagen (x, y) y el espacio de parámetros (ρ, θ), usando la ecuación de la recta en forma polar (1).

$$\rho = x\,(\cos\theta) + y\,(\sin\theta) \tag{1}$$

Donde ρ es la distancia perpendicular entre el origen de coordenadas y la recta detectada, θ es el ángulo de ρ respecto al eje x. Ya teniendo la imagen se debe conocer el $\rho_{\max}$ y ρ_{min}, al igual que el $\theta_{\max}$ y θ_{min}, para limitar el espacio de acumulaciones. Este espacio, es una matriz que se va llenando según los valores que resulten de la operación de dicha ecuación. Para eso se opera la posición (x, y) de cada pixel perteneciente al patrón que hay en la imagen

por cada ángulo que hay entre $(-\pi, \pi)$, acumulando de a uno en cada celda de espacio (ρ, θ).

Al terminar de operar, se busca el valor más alto, o sea el de mayor acumulación, obteniendo así las coordenadas polares de la recta más predominante sobre la imagen, si existen 2 o más rectas en la imagen en el plano de acumulación se pueden encontrar también otros valores mayores diferentes al encontrado. Para este trabajo el espacio de acumulaciones se redujo hasta un tercio para ganar tiempo de procesamiento.

2.4 Navegación

El sistema robótico fue diseñado para que navegue siguiendo la línea negra y una ruta preestablecida, la cual es enviada mediante comunicación Zigbee desde el PC. Por esto los datos obtenidos de la transformada de Hough lineal se deben analizar e interpretar para que el robot actúe de manera autónoma frente a la imagen presente.

Para esto se debe tener en cuenta los posibles casos que se pueden presentar en el momento que haya detección de una línea en la imagen, tomando la recta predominante como la línea que ha obtenido la mayor acumulación y la recta secundaria como la línea que ha obtenido la segunda mayor acumulación.

Primer caso: *No hay líneas detectadas*, momento en el cual no hay valores determinados para seguir una línea, así que el sistema robótico girará de manera indeterminada sobre su eje hasta que encuentre una línea a seguir.

Segundo caso: *Hay una recta detectada con un ángulo igual a 90 grados*, momento en cual se debe pasar la información por un algoritmo para determinar si la línea está centrada, a la izquierda o a la derecha. Si se encuentra centrada el sistema robótico debe seguir el curso. En la Fig. 3, se pueden observar líneas de 90 grados en distintas posiciones.

Tercer caso: *Hay una recta detectada con un ángulo menor a 90 grados*, momento en el cual el robot deberá avanzar derecho hasta que centre la línea y gire hacia la derecha.

Cuarto caso: *Hay una recta detectada con un ángulo mayor a 90 grados*, momento en el cual el robot deberá avanzar derecho hasta que centre la línea y gire hacia la izquierda.

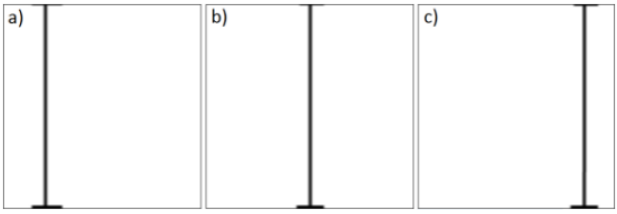

Fig. 3. Rectas con un ángulo de 90 grados. a) Línea a la izquierda, b) Línea centrada, c) Línea a la derecha. Fuente: Autores

Quinto caso: *Hay dos rectas detectadas, pero la recta predominante tiene un ángulo de 90 grados y la secundaria cero grados*, en este momento se ha detectado un tipo de cruce e inmediatamente se aplica un algoritmo para conocer el punto de intersección entre las rectas y otro algoritmo para saber si es un cruce tipo "T", tipo "L" a la izquierda o tipo "L" a la derecha. El sistema avanzará derecho hasta cambiar de caso.

Sexto Caso: *Hay dos rectas detectadas pero la recta secundaria tiene un ángulo de 90 grados y la predominante de cero grados*, en este momento de nuevo sigue existiendo el cruce, solo que la diferencia con el caso anterior es que el punto de intersección entre las rectas se está acercando al límite de la imagen por lo que el sistema robótico estará tomando la decisión de voltear según el tipo de cruce. En la Fig. 4, se aprecian los diferentes tipos de rectas.

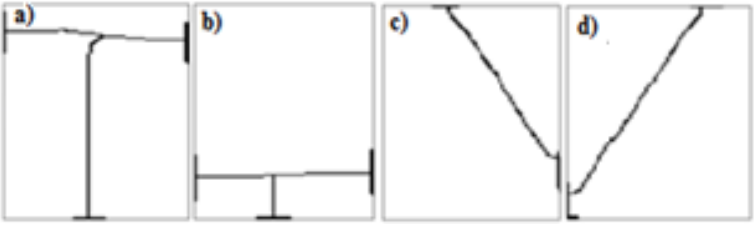

Fig. 4. a) Línea predominante con ángulo de 90 grados, b) Línea secundaria con ángulo de 90 grados, c) Línea cuyo ángulo es mayor a 90 grados, d) Línea cuyo ángulo es menor a 90 grados. Fuente: Autores

Para encontrar la intersección de las rectas se usó la ecuación punto-pendiente (2):

$$y - y_1 = m(x - x_1) \tag{2}$$

Donde y_1 es la coordenada del eje y, x_1 es la coordenada del eje x y m es la pendiente. Para hallar x y y se usaron las ecuaciones (3) y (4) para pasar de coordenadas polares a cartesianas:

$$y = \rho(\sin\theta) \tag{3}$$

$$x = \rho(\cos\theta) \tag{4}$$

Para la pendiente se usó la ecuación (5):

$$m = \tan\alpha \tag{5}$$

Donde α es el ángulo de la recta.

3. RESULTADOS Y DISCUSIÓN

Aunque la CMUcam3 es un sistema versátil, de bajo costo y que captura imágenes de baja resolución, es importante realizar un análisis de costo de procesamiento para algunos algoritmos, con el fin de determinar las capacidades y limitaciones de la CMUcam3 y sus posibles aplicaciones. Para iniciar se ha bajado la resolución de la imagen a $122 * 66$ para mejorar el tiempo de procesamiento. A continuación se presentará un análisis de tiempo de procesamiento para diferentes algoritmos.

3.1 Análisis de procesamiento

En la Fig. 5, se muestra la comparación de procesamiento entre la CMUcam3 y un computador de escritorio con un procesador AMDPhenom Triple-Core de 2.3GHz y 2 de memoria RAM.

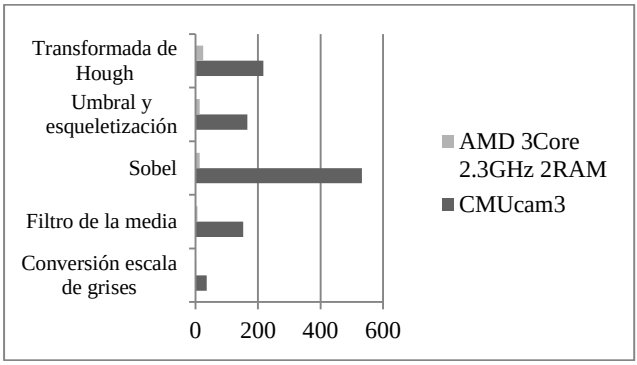

Fig. 5. Comparación de tiempos de procesamiento medidos en milisegundos. Fuente: Autores

3.2 Análisis de procesamiento para la transformada de Hough

La transformada de Hough tiene un gran coste computacional, por lo que se ha limitado algunos de sus aspectos sin afectar en gran medida su funcionamiento. Se escogió una resolución de 3 grados entre 0 y 180 para reducirlo el espacio de acumulación a un total de 60 ángulos detectables y así disminuir las operaciones a desarrollar. El espacio de acumulaciones también se ha normalizado para disminuir el tiempo de recorrido, y se ha limitado la búsqueda de rectas a solo dos en la imagen.

El tiempo que puede tomar una transformada de Hough lineal con el método propuesto para la detección de una línea es de 32ms y para la detección de un cruce es de 40ms. En la Fig. 6, se muestran imágenes con la línea detectada.

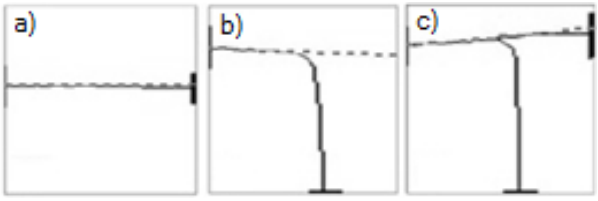

Fig. 6. Línea punteada que indica que hay una recta detectada. Fuente: Autores.

3.3 Resultados en Pistas

El sistema se probó en tres pistas, tomando diferentes puntos de inicio y llegada, verificando su rendimiento para todos los segmentos de las pistas. En la Tabla 1, se aprecian los resultados obtenidos. En la Fig. 7, se muestra el robot recorriendo una de las pistas.

Tabla1. Resultados obtenidos en pistas. Fuente: Autores

Tipos de cruces	Número de pruebas	% de aciertos	% de error
Línea Recta	100	100%	0%
Cruce izquierda	100	86%	14%
Cruce derecha	100	87%	13%
Cruce en T	100	77%	23%
	400	87.5%	12.5%

Fig. 7. Sistema robótico en pista. Fuente: Autores

4. CONCLUSIONES

En este artículo se presentó el diseño e implementación de algoritmos de navegación sobre el sistema embebido CMUcam3.A nivel de software las variaciones de los métodos propuestos en el trabajo han dado buenos resultados, logrando obtener tiempos de procesamiento por imagen entre 150 ms y 280 ms, llegando a procesar hasta 6 imágenes por segundo, dando un buen tiempo de respuesta al sistema en conjunto. El tiempo de procesamiento también está muy sujeto a la cantidad de iteraciones que demore el algoritmo para esqueletizar los patrones de líneas, ya que si son más gruesas, más lento será la respuesta del sistema, por lo que se aconseja trabajar con líneas que demoren en esqueletizarse menos

de 9 iteraciones. Para los diferentes cruces y estando fijo sobre estos, el sistema los detecta en un 99%, pero cuando está en pista pueden ocurrir errores de detección debido a la inestabilidad del sistema, ya que la cámara puede estar desalineada y tomar imágenes que pueden alterar la capacidad de percepción y llevar a procesar imágenes con datos erróneos. También se puede dar el caso cuando el robot llega desalineado y por consiguiente no alcanza a tomar la imagen correcta del cruce teniendo dificultades para guiarse de manera fluida durante la transición del cruce.

Para futuros trabajos, se espera realizar una comparación más rigurosa de los tiempos de procesamiento para diferentes sistemas embebidos con capacidad de procesar imágenes, además se adicionarán escenarios más complejos y con navegación totalmente autónoma sin necesidad de seguir patrones predeterminados usando información de profundidad a través de la visión estéreo.

5. AGRADECIMIENTOS

Especial agradecimiento al semillero de investigación en Visión Artificial y Robótica (SIVAR) del ITM. Los resultados mostrados en este artículo alimentan el proyecto P09238 financiado por el ITM.

6. REFERENCIAS

Bhave, Ajinkya, Hsiu, Thomas, Nourbakhsh, Illah, Perez-Bergquist, Andres, Richards, Steve. "Designing a low-cost, expressive educational robot", International Conference on Intelligent Robots and Systems, 2003, 2404-2409, Las Vegas, Nevada, USA.

Carelli, R, (2008). Vision-based tracking control for mobile robots, 12th International Conference on Advanced Robotics, 148-152, Seattle, USA.

Duda, Richard O., Hart, Peter E., "Use of the Hough transformation to detect lines and curves in pictures", Communications of the ACM, Vol. 15, 1972, 11 – 15.

Huang, Hua. (2008). "Automatic Detection and Recognition of Circular Road Sign", International Conference on Mechtronic and Embedded Systems and Applications IEEE/ASME, 626-630, Beijing, China.

Ikeda, M. "Navigation strategy for a quadruped robot on soft flat ground", International Conference on Control Automation and Systems (ICCAS), 2010, 62 – 65, Gyeonggi-do, Corea.

Tall, M.H., Rynne, P.F., Lorio, J.M., von Ellenrieder, K.D. "Visual-Based Navigation of an Autonomous Tugboat", OCEANS 2009, MTS/IEEE Biloxi, 1 - 9, Mississippi, USA.

Zhang, T. Y., Suen, C. Y. "A fast parallel algorithm for thinning digital patterns", Communications of the ACM, Vol. 27, 1984, 236 – 239.