Revista Tekhnê
Julio - Diciembre 2015, Vol. 12, No. 2, pp. 23– 34

ISSN 1692-8407

Algoritmo de navegación a bordo en ambientes controlados a partir de procesamiento de imágenes

Board navigation algorithm in controlled environments based on image processing

Andrés F. Barrero A.
Universidad Distrital Francisco José de Caldas
afbarreroa@correo.udistrital.edu.co

Mario F. Robayo R.
Universidad Distrital Francisco José de Caldas
mfrobayor@correo.udistrital.edu.co

Edwar Jacinto Gómez
Universidad Distrital Francisco José de Caldas
ejacintog@udistrital.edu.co

Los sistemas de navegación para ambientes estáticos se soportan en la planificación de rutas con herramientas geométricas. Implementan también métodos estocásticos y heurísticos lo que los hace complejos en su ajuste. Cada día aumenta la complejidad en la implementación de controles de navegación de robots, debido al avance constante de las características técnicas del hardware, dando posibilidad a mejores métodos de planeación de las rutas, con mayor rapidez en los cálculos y con menor error entre la trayectoria planeada y la realizada. Esta investigación propone un algoritmo adecuado para navegación con cámara a bordo en un sistema digital móvil. Resultados de laboratorio demuestra el alto desempeño de la estrategia.

Palabras clave: Algoritmo, hardware, navegación, sistema

Navigation systems for static environments are supported in planning with geometric tools. Also implement statistical and heuristic navigation methods, which makes them complex. Each day increases the complexity in implementing navigation controls, due to the constant progress of the technical characteristics of the hardware, giving possibility to better methods of path planning, with faster calculations and less error between the planned and carried out trajectory. This research proposes an algorithm suitable for navigation with camera in a mobile digital system. Laboratory results demonstrates the high performance of the strategy.

Keywords: Algorithm, hardware, navigation, system

Tipología del artículo: Investigación

Fecha recepción del manuscrito: Mayo 1, 2015
Fecha aceptación del manuscrito: Junio 5, 2015

Investigación financiada por: Universidad Distrital Francisco José de Caldas.

Edición digital: http://revistas.udistrital.edu.co/ojs/index.php/tekhne/issue/view/755
Cómo citar: Barrero, A., Robayo, M. y Jacinto, E. (2015). *Algoritmo de navegación a bordo en ambientes controlados a partir de procesamiento de imágenes*. Revista Tekhnê, 12(2), 23-34.

Introducción

En la actualidad, los robots requieren cierto grado de inteligencia, dado a que el entorno en donde navegan es cambiante, por lo cual deben ser adaptativos, aunque un autómata solo tenga ruedas, sensores, y demás dispositivos. En este caso para que el móvil sea automático y autónomo dicha inteligencia debe estar abordo. Es por esto que el robot debe estar dotado con las habilidades necesarias o suficientes para su navegación.

Para entender el problema principal es necesario aclarar el concepto principal de navegación autónoma (Beltrán y Cruz, 2013). Se entiende por navegación autónoma la capacidad de ir de un punto del espacio a otro evitando obstáculos (Gutiérrez, 2009). Esta práctica ha sido pensada y discutida como importante para los robots debido a las necesidades del mundo moderno. Está comprobado que la navegación es uno de los campos que más energías ha condensado en el campo de la ingeniería moderna. Hoy en día, es un campo de estudio muy maduro que está sirviendo de base para investigaciones muy interesantes como el campo de la manipulación móvil.

El reto principal que tiene la navegación autónoma es la localización, o sea, que cualquier robot debe saber en dónde se encuentra, a cualquier instante. Es muy significativo que este sepa dónde está, porque con eso él realiza una representación de su entorno y sabe cómo navegar dentro de él, evitando los obstáculos que se puedan presentar, y calcula un mejor camino para llegar a su destino.

Con el desarrollo del proyecto se da inicio a nuevas tendencias de mejora en el uso de sistemas de navegación a bordo, en los cuales solo es necesaria la modificación del software para implementar un nuevo control. Este desarrollo se realiza en miras a cubrir la necesidad de utilizar cada vez más y mejores sistemas de navegación sencillos para el usuario, y más en un país donde acceder a dispositivos electrónicos de gran capacidad de procesamiento, como obtener información completa acerca de algoritmos es bastante complicado, por lo cual, el aporte científico global es muy grande y de gran ayuda.

Fundamentos teóricos

Procesamiento digital de imágenes

Es el conjunto de métodos o técnicas las cuales se emplean en imágenes digitales con el fin de mejorar atributos o proporcionar información dentro de la misma.

> *...Antes de extraer información directamente de la imagen, se acostumbra a hacer un preprocesamiento, que permite resaltar los detalles más importantes de la imagen, eliminando el ruido producido en la adquisición de la imagen, como por ejemplo el producido por los cambios de iluminación espontánea o un fallo esporádico en el sistema embebido...* (Valencia y Abril, 2007)

Antes de procesar cualquier imagen, el primer paso es digitalizarla. Básicamente consiste en un sistema óptico y el digitalizador, mediante el cual la imagen visual se convierte en señales eléctricas que permitirán su procesamiento (Valencia y Abril, 2007). Ya digitalizada, las imágenes se pueden clasificar por dimensión o por paleta de colores; por dimensión se encuentran las imágenes 2D y 3D, y por paleta de colores se encuentran las imágenes binarias, en escala de grises y a color.

Las imágenes 2D son imágenes cuadradas o rectangulares, cuyos pixeles (x, y) representan regiones cuadradas. Las columnas están representadas con la coordenada y, y la coordenada representan las filas. Por conformidad, el pixel (0,0) se encuentra ubicado en la esquina superior izquierda de la imagen (Gonzalez y Woods, 2007). Una imagen digital de MxN pixeles es una función (Valencia y Abril, 2007) (ec. 1).

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix} \tag{1}$$

Al digitalizar una imagen el sensor envía un valor dentro de una escala. Este valor pueden ser, o bien un único valor (escala de grises) o bien un vector con tres valores por pixel (RGB) que corresponden con la intensidad de color rojo (R), verde (G) y azul (B). Esta escala tiene un rango discreto y combinándolas se obtienen todos los colores visibles. Las imágenes en escala de grises con sólo 2 colores: blanco y negro (0 y 1, respectivamente), se llaman imágenes binarias (Fig. 1). A este proceso de discretización del color se le llama cuantificación (Calle, 2014).

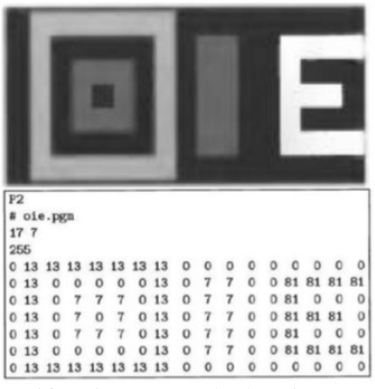

Figura 1. Cuantificación en escala de grises.

ISSN 1692-8407

Sistemas embebidos

Son sistemas electrónicos planteados concretamente para realizar unas funciones establecidas, habitualmente formando parte de un sistema de mayor entidad (Pedre, 2012). La particularidad principal es que emplea para ello uno o varios procesadores digitales también llamados CPU's (como microprocesadores, microcontroladores o DSP). Esto permite contribuir a que el sistema tenga cierto nivel de *inteligencia* en la tarea o aplicación para la que se esté usando.

Son sistemas diseñados para llevar a cabo una o pocas funciones dedicadas y que esta empotrado como parte de cierto mecanismo de hardware completo (Pedre, 2012). Algunos ejemplos que se encuentran son (Úbeda, 2009):

- Embebidos en industria automotriz: navegador GPS, de la aviación: piloto automático, control de aterrizaje.
- Embebidos en telecomunicaciones: routers, módems
- Embebidos en comunicaciones: teléfonos celulares
- Embebidos en el hogar: control de heladeras, microondas, robots que aspiran, cortan pasto.

Un sistema embebido usualmente se basa en un módulo electrónico albergado dentro de un sistema de mas grande como un *host*, el cual ayuda en la ejecución de trabajos tales como el procesamiento de información generada por sensores, el control de determinados actuadores, etc.

El software tendrá algunos requisitos específicos dependiendo de la aplicación (Pedre, 2012). Normalmente, para el diseño de un sistema embebido no se tiene recursos ilimitados, sino que la cantidad de memoria será escasa, y la capacidad de cálculo y dispositivos externos será limitada (Pedre, 2012).

Ambiente controlado

Un ambiente controlado es un entorno cerrado donde parámetros tales como luz, temperatura, humedad relativa y demás variables medioambientales están completamente controlados (Betancourt y Serrato, 2014). Normalmente, el ambiente controlado busca fijar un medio simple dentro del cual la plataforma robótica se pueda desplazar y navegar sin ningún problema. Para esto se debe determinar una tasa de luz constante, el cual sería el entorno de trabajo ideal, además de identificar unos elementos, los cuales la plataforma deberá reconocer e identificar para navegar dentro del ambiente.

Dentro del desarrollo del ambiente controlado se tienen en cuenta varios factores para determinar una tasa de luz constante, además de evitar mezclar la luz fluorescente con fuentes de luz de otra naturaleza. Por lo tanto se establece que para que la propagación de la luz sea uniforme de debe utilizar un fondo totalmente blanco, para que el ángulo de reflexión de los rayos de luz sea uniforme en cualquier parte del ambiente (Serway y Jewett, 2008).

Ahora bien, teniendo en cuenta el fondo del ambiente, se debe identificar una serie de elementos los cuales hacen que una plataforma navegue de manera correcta. Estos elementos son una serie de líneas y contornos, los cuales permiten identificar un objeto o elemento, por lo tanto, es debido optar por el uso y manejo de diferentes figuras geométricas (Fig. 2). Estas figuras son las que se identificarán, y de acuerdo con la instrucción dada por el usuario, la plataforma navegará hacia esta.

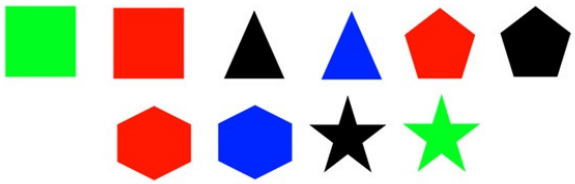

Figura 2. Figuras del ambiente controlado.

Transformada de Hough

La transformada de Hough es una técnica utilizada para aislar características de forma particular dentro de una imagen. La idea básica es encontrar curvas que puedan ser parametrizadas como líneas rectas, polinomios y círculos (Urrea y Ospina, 2004). Una línea recta en el plano de coordenadas $(x - y)$ puede ser descrita plenamente por solo dos medidas, la pendiente y la intersección con eje y. La parametrización más conocida es (ec. 2):

$$y = mx + b \tag{2}$$

Donde m es la pendiente y b es la intersección con el eje y. Esta línea, cuando se dibuja en el plano de parámetros $(m - b)$ se convierte en un punto (Berrio, Orozco, y Caicedo, 2012). Esto se puede observar, cuando, dado un punto (x, y), el conjunto de líneas a través de (x, y) donde cada una es parametrizada por algún par (m, b) forman en una línea recta en el plano de parámetros $(m - b)$ (Berrio et al., 2012). En la siguiente figura se ejemplifica la conversión de líneas a puntos, y viceversa. Un punto con líneas de varios parámetros (m, b) a través de estos se transforma en una línea recta en el plano de parámetros, mientras que los puntos colíneales se transforman en líneas rectas que se interceptan en un único punto en el plano de parámetros (Fig. 3).

Y de manera inversa, es indiscutible que la ecuación de transformación que indica la relación lineal entre m y b es (ec. 3):

$$b = -xm + b \tag{3}$$

Cuando los puntos que se encuentran alineados con otros en el plano de coordenadas $(x - y)$, se transforman en sus respectivas líneas en el plano de parámetros $(m - b)$, estas tendrán un único punto de intersección. Las

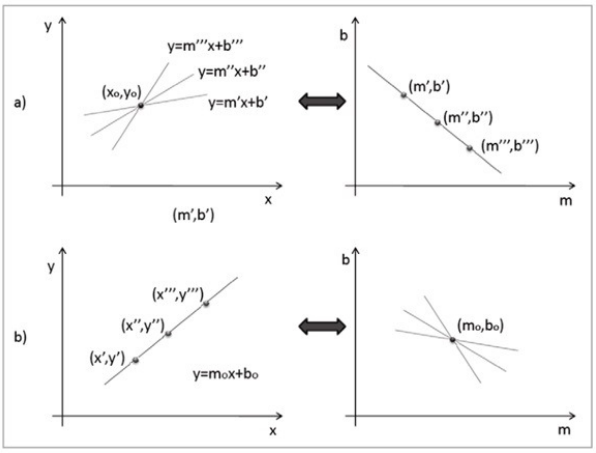

Figura 3. Transformaciones de líneas. a) Punto a línea, b) Puntos colíneales (línea) a punto (Berrio et al., 2012).

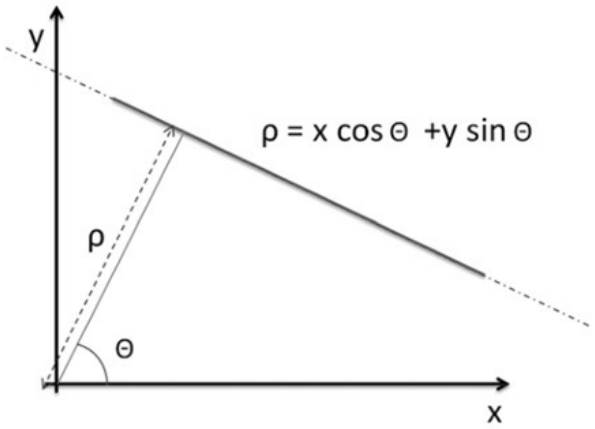

Figura 4. Parametrización normal (Berrio et al., 2012).

coordenadas (m, b) del punto de corte son exactamente la pendiente y el punto de intersección con el eje y de la línea que describen los puntos. Usando esta parametrización, existen particularidades cuando la pendiente, la variable independiente en este caso, no tiene límite (cuando una línea en el plano de coordenadas es paralela al eje y, la pendiente tiende a infinito) (Parker, 2010).

Teniendo en cuenta esta particularidad, una manera más conveniente para describir la línea recta es utilizando la parametrización normal representada en la ec. 4 (Berrio et al., 2012).

$$\rho = x\,cos\theta + y\,sen\theta \tag{4}$$

Una línea en el plano de coordenadas es descrita completamente por dos parámetros, el ángulo de la normal desde el origen (θ), y su distancia del origen (ρ). Siguiendo la conversión de punto a línea, dado un punto (x, y) en el plano de coordenadas, el grupo de línea a través del punto mencionado, cada una de las cuales es parametrizada por algún par, será transformada en una curva sinusoidal en el plano de parámetros (θ, ρ), curva correspondiente a la ecuación de la parametrización normal (Gonzalez y Woods, 2007) (Fig. 4).

En este caso, los puntos que se encuentran sobre una línea en el plano de coordenadas $(x - y)$ son transformados en sus respectivas curvas sinusoidales en el plano de parámetros. Estas curvas en el plano de parámetros se interceptarán en un único punto, las coordenadas (θ, ρ) de este punto de intersección son exactamente el ángulo de la normal y la distancia desde el origen a la línea que pasa a través de los puntos en el plano de coordenadas.

El algoritmo de la transformada de Hough hace uso de la transformación punto-curva. Este algoritmo se basa en la inferencia que puntos colíneales en el plano de coordenadas serán transformadas en curvas con un único punto de intersección en el plano de parámetros. Así, para encontrar una línea, la forma es transformar los puntos a curvas en el plano de parámetros y localizar las intersecciones de las curvas para determinar la presencia de líneas (Berrio et al., 2012; Gonzalez y Woods, 2007; Parker, 2010).

En la Fig. 5 se ejemplifica las transformaciones de punto - curva en donde se observa en primera instancia, como un punto con líneas con diferentes parámetros pasando a través de él se transforma en una curva sinusoidal en el plano de parámetros. Seguido se muestra como varios puntos colíneales se convierten en curvas que se interceptan en un único punto en el plano de parámetros (Berrio et al., 2012).

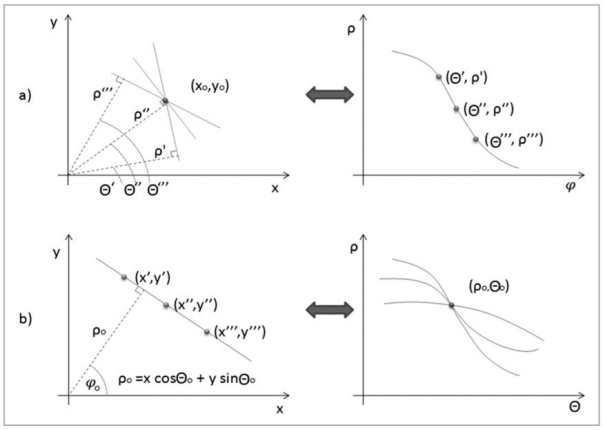

Figura 5. Transformaciones. a) Punto a curva, b) Puntos colíneales (línea) a curvas interceptadas (Berrio et al., 2012).

Metodolocía para el procesamiento de imágenes

Implementación del hardware

Para la implementación es necesario acoplar el sistema embebido adecuado para el procesamiento de imágenes a bordo. Para la ejecución del sistema, se utilizó un

ISSN 1692-8407

sistema embebido móvil llamado Raspberry Pi (Fig. 6), el cual es una computadora del tamaño de una tarjeta de crédito que se puede conectar una TV o un monitor, un teclado y un mouse para poder ser una computadora 100 % funcional. La Raspberry Pi es capaz de funcionar con diversos sistemas operativos GNU/Linux como Debian, Fedora, Arch Linux, Qton, Risc OS, ó Raspberry PWN, además de su propia versión Debian modificada llamada Raspbian (Vujović y Maksimović, 2015).

Figura 6. Raspberry Pi 2 Modelo B (Adafruit, 2015).

Se escoge este sistema embebido móvil para la implementación de los algoritmos de procesamiento de imágenes, debido a sus excelentes características en relación con su tamaño, especialmente con su velocidad de procesamiento. Es importante resaltar también la velocidad de procesamiento de video de la Raspberry Pi, ya que esta característica permite que la implementación de los algoritmos de procesamiento de imágenes sean más ligeros, y lo suficientemente rápidos para que el cálculo del algoritmo sea a bordo.

Adquisición de la imagen

Para realizar la adquisición de la imagen, teniendo en cuenta que se va a utilizar un sistema embebido, el cual será la Raspberry Pi, se manejó una cámara compatible con este sistema. El módulo de la cámara es un accesorio de diseño personalizado para la Raspberry Pi. Se adhiere a esta a través de una de las dos pequeñas tomas de corriente en la superficie superior. Esta interfaz utiliza la interfaz dedicada CSI, que fue diseñado especialmente para la conexión a las cámaras. El bus CSI es capaz albergar muy altas velocidades de datos, y lleva exclusivamente datos de píxeles. Esto ayuda bastante a que el procesamiento de la información sea exclusivo, por lo que será más rápido el proceso.

El sensor en sí tiene una resolución nativa de cinco megapíxeles, y tiene una lente de foco fijo a bordo. En cuanto a las imágenes fijas, la cámara es capaz de 2592 x 1944 píxeles imágenes estáticas, y también es compatible con 1080p30, 720p60 y 640x480p60/90 video (Laing, 2013).

Implementación del software

La segunda etapa a realizar para la puesta en marcha del sistema, es la implementación del software que permitirá el correcto funcionamiento del procesamiento de imágenes a bordo. Para la ejecución del sistema se utilizó Python como lenguaje de programación base. Python es un lenguaje muy flexible en cuanto a su manejo, por lo tanto es capaz de operar datos obtenidos por medios externos (en este caso la cámara a bordo), modificarlos o interpretarlos como mejor se suponga. Es por esta razón que se puede abreviar el trabajo y reducir el número de programas o bibliotecas ejecutadas paralelamente.

En la ejecución de los algoritmos de procesamiento de imágenes, se utilizó la biblioteca de funciones de programación para la visión de computadora en tiempo real OpenCV. La librería de visión por computador OpenCV trata de un grupo de funciones orientadas a la resolución de problemas típicos de la visión por computador y con una comunidad detrás muy importante de profesionales de la visión que se encarga de mantenerla actualizada (Bradski y Kaehler, 2008).

Para que todas las etapas del procesamiento funcionaran correctamente, se creó un programa que fusionara todas las operaciones y cálculos que realiza el sistema, para hacerlo trabajar en conjunto. Aunque las etapas estén encadenadas, la etapa de comunicación no es propiamente una etapa programada. Esta etapa se encargará de la monitorización del procesamiento de imágenes a bordo, y permite verificar que funcione de manera correcta.

En la Fig. 7 se muestra un diagrama general de la unión entre las etapas. Como se puede observar, la etapa de procesamiento a bordo es la más importante, la cual se realizan en el sistema embebido. También se puede prestar atención que si bien la etapa de comunicación no está incluida dentro de la etapa de procesamiento, es por esta etapa donde se visualizan los datos provenientes de la etapa de procesamiento. Otra de las características que se pueden comprobar que la cámara a bordo es la que se encarga de introducir y extraer los datos que está tomando.

Figura 7. Diagrama general del procesamiento de imágenes a bordo.

La primera elección que se tuvo que hacer en el software, fue la elección del sistema operativo que utilizara la Raspberry Pi, debido a que actualmente existen varias distribuciones de Linux que funcionan perfectamente en el sistema embebido. Para esto se escogió el sistema operativo Raspbian, basado en la distribución Debian. Este sistema operativo ofrece más de 35.000 paquetes y software pre-compilado (Bradski y Kaehler, 2008), lo que hace que sea más fácil la instalación en la Raspberry Pi. Igualmente, el software incluido es muy versátil y permite agilizar el uso del sistema embebido.

Etapa de comunicación

Esta etapa consiste en controlar la plataforma móvil dentro del ambiente controlado de manera remota. Para esto se conecta desde un computador o dispositivo móvil a un navegador web, haciendo petición al servidor web, que está instalado en la Raspberry Pi, el cual responderá con una interfaz de control, donde se elegirá la figura a buscar dentro del ambiente controlado. Con esta petición, esta se conecta internamente con PHP, donde ejecuta el script de Python, el cual envía las señales a la plataforma móvil, para poner en marcha los dos motoreductores y navegue en el ambiente controlado (Fig. 8).

Figura 8. Diagrama de bloques de la etapa de comunicación.

En principio se debe realizar una conexión de tipo SSH entre el dispositivo Raspberry Pi y un cliente SSH. Dicho cliente puede estar construido en cualquier sistema operativo, en este caso, se utilizó en el cliente de SSH para el sistema operativo Windows. El cliente SSH para Windows es llamado PuTTY, dicho cliente es compatible con los sistemas operativos Windows XP, Vista, 7 así como los sistemas operativos basados en UNIX y GNU/Linux. Con la conexión realizada, el programa en PHP realiza la comunicación entre la interfaz gráfica de la página web, y ya funcionara el programa de manera autónoma. La interfaz gráfica principal vista desde cualquier navegador de Windows se observa en la Fig. 9.

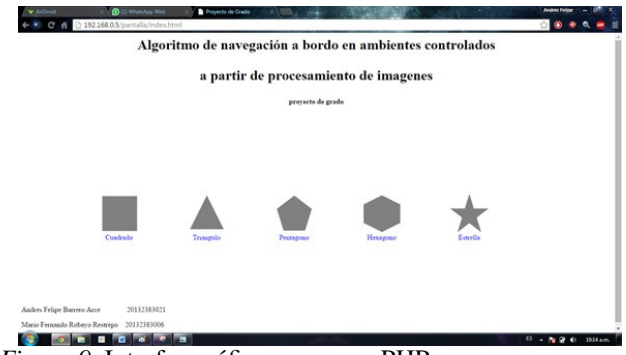

Figura 9. Interfaz gráfica programa PHP.

Comprobación

Para comprobar el procesamiento de imágenes a bordo se utilizaron dos tipos de algoritmos básicos y especiales, los cuales son la primera parte del algoritmo de navegación en un ambiente controlado. El primer algoritmo a probar fue la conversión de la imagen RGB a escala de grises, y seguido se aplicará el filtro Canny.

RGB a escala de grises. Después de ejecutada la adquisición de la imagen, a esta se le realiza un filtrado que permita obtener la imagen en escala de grises (pixeles entre 0 y 255), para reducir el costo computacional, pues se trabaja únicamente con una matriz y no con tres como en el caso de imágenes en formato RGB (Alezones, Borrero, y Baquero, 2011).

La Fig. 10, en la parte izquierda, muestra la imagen original en RGB, mientras que en la parte derecha se ilustra la misma imagen después de ser convertida a escala de grises.

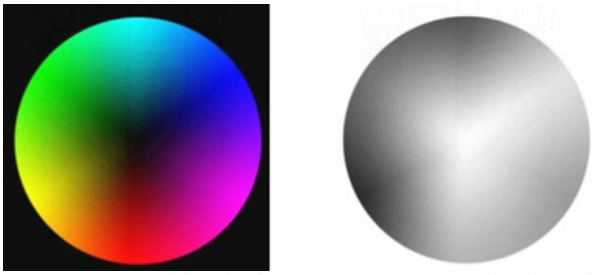

Figura 10. Cambio de imagen RGB a escala de grises (Bradski y Kaehler, 2008).

La trasformación de la imagen se realiza mediante el uso de luminancia del color en sus tres componentes de color. La luminancia favorece el componente verde de un pixel, seguido por el rojo, y después el azul. La fórmula de

ISSN 1692-8407

aceptación general para el cálculo de la luminancia de un valor RGB es la mostrada en la ec. 5 (Parker, 2010):

$$f(x,y) = 0,299 \times R + 0,587 \times G + 0,114 \times B \qquad (5)$$

Donde R, G y B son los valores de los píxeles en cada uno de los planos de color rojo, verde y azul respectivamente, y $f(x,y)$ es su respectivo valor en niveles de gris. La imagen de muestra para realizar las pruebas del algoritmo RGB a escala de grises, que es tomada por el sistema embebido móvil hace parte del ambiente controlado utilizado para el proyecto (Fig. 11).

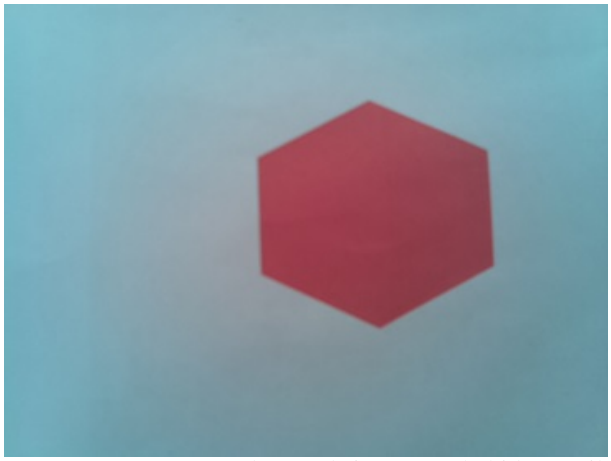

Figura 11. Imagen tomada por el sistema embebido móvil.

Al final se obtiene como resultado una representación de la misma imagen pero con valores que se almacenan en una sola matriz (Fig. 12) (Gonzalez y Woods, 2007).

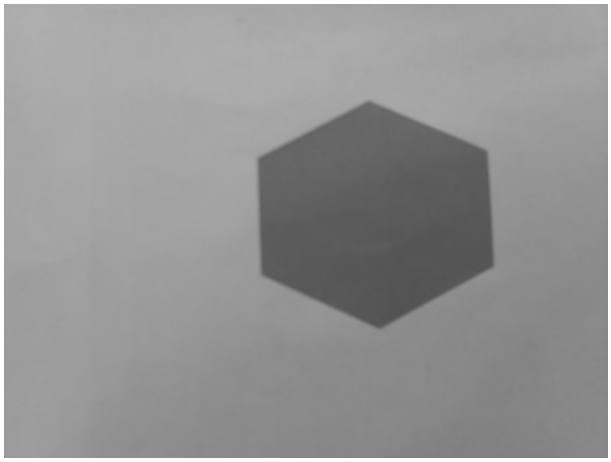

Figura 12. Imagen en escala de grises.

Filtro Canny. Inmediatamente después de tener nuestra imagen en escala de grises, se pasará por el filtro Canny, el cual es un método para la detección de bordes en una imagen. Este filtro se basa en tres criterios (Valverde, 2007):

- Un criterio de detección expresa el hecho de evitar la eliminación de bordes importantes y no suministrar falsos bordes.
- El criterio de localización establece que la distancia entre la posición real y la localizada del borde se debe minimizar.
- El criterio de una respuesta que integre las respuestas múltiples correspondientes a un único borde.

Para probar el funcionamiento del algoritmo se utiliza la misma obtenida del paso de RGB a escala de grises, debido a que este algoritmo para su funcionamiento es necesario una imagen en escala de grises (Fig. 13).

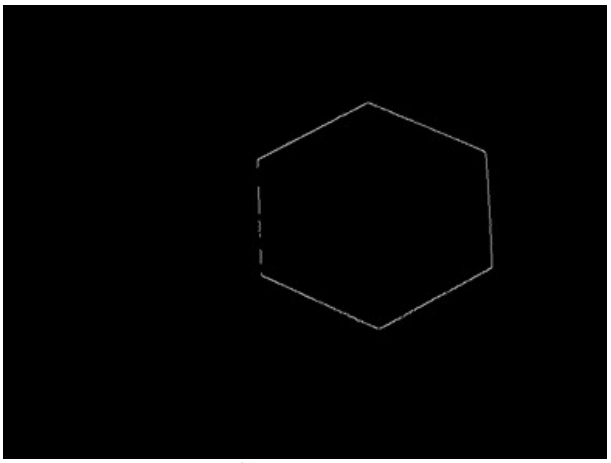

Figura 13. Imagen con filtro Canny.

El propósito de estas pruebas es comprobar el funcionamiento de procesamiento de imágenes a bordo en un sistema embebido móvil, como lo es la Raspberry Pi. Se evidenció que el dispositivo está diseñado para bastantes funciones, entre las cuales adquirir y procesar imágenes es una de ellas. Se observa como el sistema embebido móvil hace el procesamiento de imágenes debido, desde la adquisición hasta el algoritmo final Canny. Se tiene en cuenta que el algoritmo Canny no está calibrado, este paso se hace en la implementación completa de algoritmo para la navegación en un ambiente controlado.

Implementación del algoritmo

Para las primeras pruebas en el sistema embebido móvil se utiliza la figura con más lados posibles dentro del ambiente controlado. Esta figura es la estrella, la cual tiene 10 lados, lo que permite realizar una calibración correcta para esta y las demás figuras del ambiente controlado. El algoritmo de procesamiento de imágenes contiene varias etapas para su funcionamiento: primero, se toma la imagen correspondiente al ambiente controlado y se pasa por un preprocesamiento para resaltar las características más importantes de la misma, y así facilitar el trabajo al siguiente algoritmo. Luego se utiliza el algoritmo de la transformada de Hough para realizar

la segmentación respectiva de la imagen, la cual seleccionara los puntos más relevantes del ambiente controlado, y así, de esta manera, se posee una representación digital del ambiente que enfoca sus características más importantes, y se descarta lo redundante.

Resultados de preprocesamiento

Con el preprocesamiento de la imagen se pretende encontrar los detalles más importantes y eliminar el ruido que se pueda producir en la adquisición de la imagen. El preprocesamiento de la imagen sigue los mismos algoritmos utilizados en la metodología del procesamiento de imágenes, o sea, en principio adquiere la del ambiente controlado, después la pasa a escala de grises y finalmente realiza el filtro Canny.

La primera parte del preprocesamiento es la conversión a escala de grises de la imagen adquirida, como se evidencia en la Fig. 14. Esta transformación no necesita de calibración alguna debido a que se realiza mediante la ecuación de la luminancia del color.

Figura 14. Figura en escala de grises.

Para aplicar el respectivo procesamiento de la imagen, se ejecuta el filtro Canny, para realzar los bordes de la imagen con la búsqueda de los máximos locales del gradiente de la imagen.

En la Fig. 15 se ilustra la imagen con el filtro Canny ya desarrollado con los valores de histéresis máxima y mínima según lo recomendado por los desarrolladores de OpenCV (200 y 100 respectivamente) (Bradski y Kaehler, 2008). Para que tenga mejor calidad para el procesamiento se cambian estos valores mediante diferentes pruebas hasta obtener los umbrales de histéresis adecuados.

Resultados de procesamiento

Para el procesamiento utilizando la trasformada de Hough, se debe tener en cuenta que la imagen ya procesada es una imagen binaria, que es el resultado del filtro Canny.

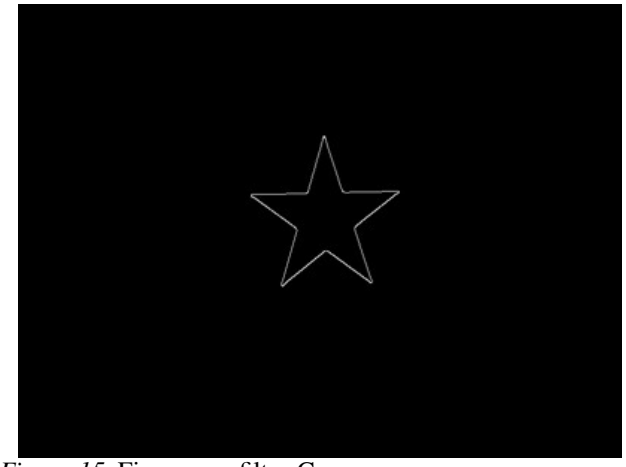

Figura 15. Figura con filtro Canny.

Observando las virtudes que proporciona la librería OpenCV, se encuentra que se puede implementar dos tipos de funciones que permiten realizar el algoritmo de la Transformada de Hough. La primera es la función *cv2.HoughLines()*, la cual simplemente devuelve una matriz de valores (ρ, θ). Rho (ρ) se mide en píxeles y theta (θ) se mide en radianes.

En la transformada de Hough se puede ver que, incluso para una línea con dos argumentos, se necesita una gran cantidad de cómputo. Por lo que la segunda función es la Transformada de Hough Probabilística (*cv2.HoughLinesP()*), que es una optimización de la transformada de Hough clásica. En esta función no se toman todos los puntos en consideración, solo se toma un subconjunto aleatorio de puntos que es suficiente para la detección de línea.

Para verificar el correcto funcionamiento de la función, al final de su procesamiento se guarda la imagen original con las líneas detectadas dibujadas en ella, y así contrastar las líneas que la función detectó.

A pesar de realizar las medidas y calibraciones en los algoritmos anteriores, se encuentra que en ciertas ocasiones, el algoritmo identifica líneas de más al final del procesamiento, las cuales son paralelas a las líneas de la figura, pero que no hacen parte de la figura. Para resolver esta dificultad se opta por realizar un pequeño cálculo a cada segmento de línea de la figura, para eliminar las líneas paralelas que no hacen parte de la figura, y solo contar la línea correspondiente a la figura (Fig. 16).

Con el cálculo anterior, se tiene certeza de la implementación del algoritmo de procesamiento de imágenes en un ambiente controlado con varias pruebas. En principio se observa que se tiene un problema por la cantidad de líneas detectadas, pero utilizando un tratamiento de la imagen adicional, se observa que la cantidad de líneas detectadas son las mismas que la figura tomada, solo

ISSN 1692-8407

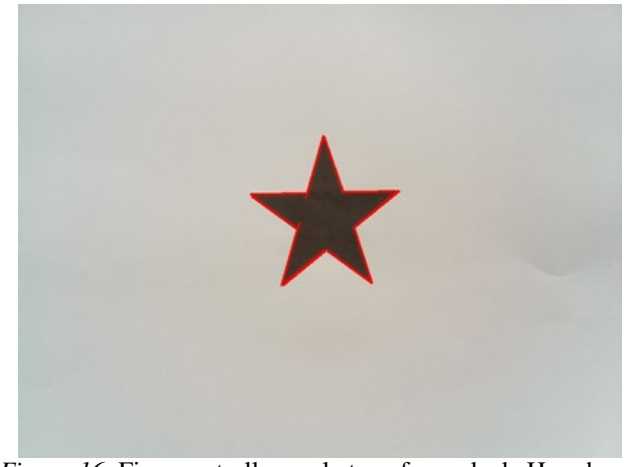

Figura 16. Figura estrella con la transformada de Hough.

aumenta en 200 milisegundos el proceso completo del script de Python (Fig. 17).

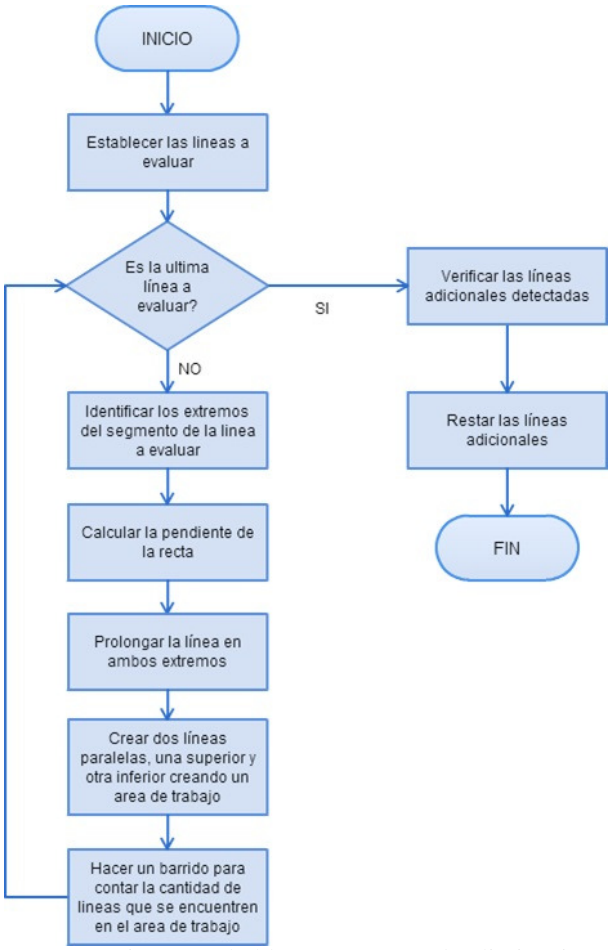

Figura 17. Diagrama de bloques proceso de eliminación de líneas paralelas.

Pruebas de navegación

Arquitectura de la plataforma móvil

Para que la plataforma móvil funcione correctamente se utilizaron dos tarjetas electrónicas de potencia, cada una controla la velocidad y el sentido de giro de dos motoreductores. Para la puesta en marcha de estas tarjetas se utilizó el circuito integrado L293B, el cual reúne cuatro drivers, dos para cada motoreductor, y es capaz de proporcionar una corriente de salida de máximo 1 A (SGS-Thomson, 2003), más que suficiente para el sistema.

Para conseguir un acople entre los pines de salida del GPIO de la Raspberry Pi y el circuito de potencia, se utilizó un conversor el cual posibilita acoplar los niveles de 3,3 V de la Raspberry Pi a 5 V, de la plataforma móvil. El circuito integrado para este conversor es el TXB0108, el cual es un conversor bidireccional que detecta automáticamente la dirección de las señales (Texas Instruments, 2010). La Fig. 18 muestra el circuito típico de operación y la placa electrónica.

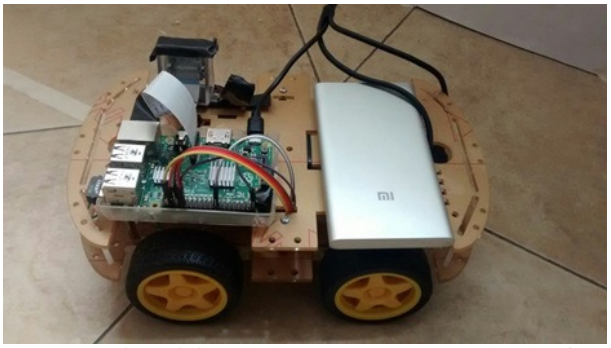

Figura 18. Disposición de los componentes en la plataforma móvil.

Para llevar a cabo la implementación final, se requiere de una ubicación específica de las tarjetas de potencia, el conversor de niveles, el sistema embebido y la cámara a bordo. Durante las primeras pruebas se comprobó que la versatilidad de la plataforma móvil permite ubicar de manera ordenada y segura los componentes ya mencionados, optimizando la navegación a bordo. Se aprovechó el distintivo de la plataforma de tener dos estructuras en acrílico, lo que permite poner en la parte inferior de estas las tarjetas de potencia, el conversor de niveles y la batería, utilizando los agujeros que incluye la estructura. Ya en la parte superior se ubica la Raspberry Pi con la cámara a bordo, estos dos protegidos con su caja correspondiente.

Proceso de navegación

En cuanto a la navegación de la plataforma móvil en el ambiente controlado se tiene en cuenta el diagrama de flujo de la Fig. 19, que permite recorrer el ambiente controlado

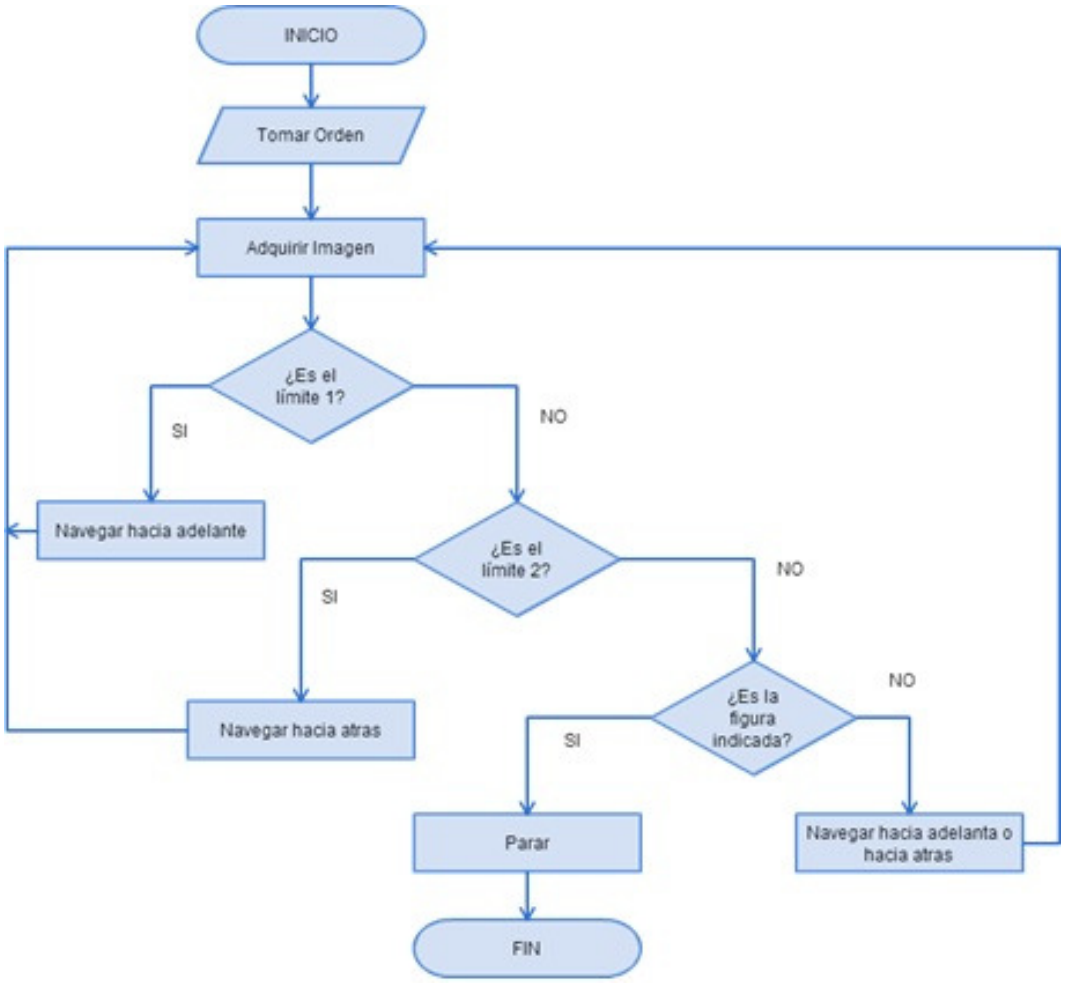

Figura 19. Diagrama de flujo del proceso de navegación.

hacia adelante o hacia atrás, dependiendo de la orden que fue tomada en la etapa de comunicación.

Los limites mencionados en el diagrama de flujo son la figura estrella, la cual es la que tiene más lados, y serán los puntos de referencia para el posicionamiento inicial y final de la plataforma móvil en el ambiente controlado.

Los desplazamientos hacia adelante y hacia atrás de la plataforma móvil los realiza mediante pasos, de aproximadamente 3 a 4 cm los cuales son controlados por PWM, directamente desde el sistema embebido móvil. Por cada paso que la plataforma realiza, adquiere la imagen del ambiente controlado para su debido procesamiento.

Pruebas

Las primeras pruebas se desarrollaron con un ambiente controlado improvisado. Este procedimiento se realizó para verificar el comportamiento de los algoritmos ante un ambiente controlado con incidencia de luz exterior (nivel moderado de ruido), diferente a la que debe tener el entorno de trabajo.

Las siguientes pruebas fueron realizadas en un ambiente controlado más organizado, teniendo en cuenta las especificaciones mínimas de iluminación con las cueles se calibraron los algoritmos de procesamiento de imágenes Canny y la Transformada de Hough.

Análisis de pruebas

En las primeras pruebas se encontró que los resultados del procesamiento de la imagen dan una efectividad de un 60 % aproximadamente, esto ya que varía la cantidad de líneas detectadas, debido a que los valores calibrados anteriormente son para un ambiente controlado limpio y sin ruido exterior, por lo tanto se evidencia que existen resultados negativos.

Las pruebas con un ambiente controlado más constituido arrojan que la navegación mejora hasta tener una efectividad aproximada de un 84 % basado en 50 pruebas, que aunque no es un desempeño defectuoso, debería cumplir con un estándar más alto de certeza. Una de las maneras que pueden o no afectar el comportamiento del algoritmo de procesamiento de imágenes son las variables calibradas

ISSN 1692-8407

en principio, porque aunque se hacen para un ambiente controlado estructurado, la reflexión de la luz siempre varia, además de la frecuencia con la que se genera esta, que para el ojo humano es imperceptible, pero para el sensor de la cámara a bordo no, por lo cual la imagen adquirida siempre varia.

La confianza que se tiene con el algoritmo utilizado no es la más óptima, aun habiendo creado un cálculo adicional de corrección de errores para líneas paralelas detectadas. Por lo tanto, es bueno experimentar con la utilización de otro tipo de algoritmos que, aunque no sean lo más rápidos como lo es la transformada de Hough, tienen una mejor garantía en el resultado final de navegación dentro de un ambiente controlado.

Conclusiones

Se comprobó el funcionamiento del procesamiento de imágenes a bordo en un sistema embebido móvil, con la evaluación de los algoritmos Canny y RGB a Escala de Grises. Esta implementación permitió verificar todas las etapas correspondientes para su correcto funcionamiento en todas las fases de implementación, desde la adquisición de la imagen hasta su debido proceso de cómputo.

Se realizaron las pruebas correspondientes a la navegación de la plataforma móvil robótica para verificar su funcionamiento en un ambiente controlado. En principio se ejecutaron una serie de pruebas en un ambiente poco controlado, el cual demostró que con un 60 % de efectividad, no funciona de manera recomendable debido a la calibración del algoritmo. Seguido a esto se realizaron pruebas en un ambiente controlado completamente vigilado y con una efectividad del 84 %, evidentemente aumenta la garantía del algoritmo implementado permitiendo mejorar la navegación autónoma.

Tras la realización de un análisis para una posible al proyecto, es posible concluir que el algoritmo de procesamiento de imágenes es la parte más importante del proyecto ya que permite que su propósito final, el cual es la navegación autónoma de un robot dentro de un ambiente controlado sea cumplido a cabalidad.

Las pruebas de navegación mostraron que el sistema cumplía satisfactoriamente con el proyecto, sin embargo cabe destacar dos debilidades del sistema. La primera, que solo es funcional cuando el ambiente controlado es totalmente vigilado, ya que una vez que la cantidad de luz que incide sobre este varia bastante, el algoritmo no funciona correctamente. Para este caso es recomendable estar en constante vigilancia del ambiente controlado en el cual está navegando la plataforma robótica móvil.

La segunda debilidad en donde se presentan errores es cuando se detectan más contornos en la figura adquirida, lo cual se soluciona implementando una pequeña operación adicional que soluciona este problema. Este cálculo borra las líneas añadidas y solo ubica la línea correspondiente de la figura. Esto sucede de igual manera por la incidencia de luz que pueda contener la imagen adquirida por el sistema embebido móvil.

Referencias

Adafruit. (2015). *Introducing the raspberry pi 2 - model b.* online. Descargado de `https://learn.adafruit.com/introducing-the-raspberry-pi-2-model-b?view=all`

Alezones, Z., Borrero, H., y Baquero, Y. (2011). Sistema de reconocimiento de figuras geométricas mediante comandos de voz implementado en un robot móvil. En *Congreso internacional de ingeniería mecatrónica - unab* (Vol. 2, p. 1-5).

Beltrán, E., y Cruz, J. (2013). Desarrollo de aalgoritmos para la coordinación de un sistema multirrobot cooperativo para tareas de búsqueda de fuentes de calor en entornos dinámicos, heatbot. *Tekhnê*, *10*(1), 28-37. (ISSN 1692-8407)

Berrio, J., Orozco, S., y Caicedo, E. (2012). Extracción de lííneas a partir de escaneos láser integrando transformada de hough, mínimos cuadrados totales y seguimiento de bordes sucesivos. *Revista Ingeniería*, *17*(1), 48-59.

Betancourt, D., y Serrato, B. (2014). *Sistema de visión por computador para detectar hierba no deseada en prototipo de cultivo de frijol usando ambiente controlado.* (Universidad Católica de Colombia)

Bradski, G., y Kaehler, A. (2008). *Learning opencv: Computer vision with the opencv library* (1.ª ed.). O'reilly.

Calle, I. (2014). *Navegación autónoma de un robot móvil usando técnicas probabilísticas de localización y mapeo basadas en métodos monte carlo secuenciales.* (Universidad Nacional de Ingeniería)

Gonzalez, R., y Woods, R. (2007). *Digital image processing* (3.ª ed.). Pearson.

Gutiérrez, J. (2009). *Trayectorias para robots móviles de navegación autónoma.* (Instituto Politécnico Nacional)

Laing, A. (2013). *Unofficial guide to getting up and running with the raspberry pi camera.* online. Descargado de `www.element14.com/community/servlet/JiveServlet/downloadBody/54413-102-1-273177/Unofficial%20guide%20to%20getting%20up%20and%20running%20with%20the%20Raspberry%20Pi%20Camera.pdf`

Parker, J. (2010). *Algorithms for image processing and computer vision* (2.ª ed.). Wiley.

Pedre, S. (2012). *Sistemas embebidos - notas.* (Facultad de Ciencias Exactas y Naturales - Universidad de Buenos Aires)

Serway, R., y Jewett, J. (2008). *Física para ciencias e ingeniería* (7.ª ed., Vol. 2). Cengage Learning.

SGS-Thomson. (2003). *Push-pull four channel drivers l293b.* (Datasheet)

Texas Instruments. (2010). *8-bit bidirectional voltage-level translator with auto-direction sensing and +-15 kv esd protection txb0108.* (Datasheet)

Úbeda, B. (2009). *Apuntes de: Sistemas embebidos.* (Universidad de Murcia)

Urrea, J., y Ospina, E. (2004). Implementación de la transformada de hough para la detección de líneas para un sistema de visión de bajo nivel. *Scientia Et Technica*, *X*(24), 79-84.

Valencia, J., y Abril, M. (2007). *Registro de transeúntes en tiempo real utilizando un sistema de visión artificial sobre un ambiente controlado.* (Universidad Tecnológica de Pereira)

Valverde, J. (2007). *Detección de bordes mediante el algoritmo de canny.* online. Descargado de `www.researchgate.net/publication/267240432_Deteccion_de_bordes_mediante_el_algoritmo_de_Canny`

Vujović, V., y Maksimović, M. (2015). Raspberry pi as a sensor web node for home automation. *Computers & Electrical Engineering*, *44*, 153-171.