Identificación visual sobre sistema embebido para navegación robótica autónoma

Visual Identification on Embedded System for Autonomous Robotic Navigation

Recibido: 17 de febrero de 2014 • Aprobado: 25 de marzo de 2014

Para citar este artículo: F. H. Martínez, F. Martínez, H. Montiel, «Identificación visual sobre sistema embebido para navegación robótica autónoma», *Ingenium*, vol. 15, n.º 29, pp. 71-84, mayo, 2014.

Fredy H. Martínez S.*,
Fernando Martínez S.** y Holman Montiel***

Resumen

Este artículo de investigación ataca el problema de la navegación robótica autónoma basada en sistemas de visión artificial en tiempo real. Como una primera aproximación a un sistema de visión artificial que permita la navegación del robot, se ha desacoplado el módulo sensor e implementado como una unidad independiente sobre un pequeño robot diferencial. Sobre él, se han realizado estudios básicos de navegación por identificación visual de *landmarks* (marcas especiales) en un ambiente desconocido. Los estudios de desempeño, realizados tanto a través de simulación como por evaluación directa sobre prototipos en laboratorio, demuestran la eficacia de las estructuras propuestas, tanto del hardware de identificación en tiempo real como de la estrategia básica de navegación considerada para su evaluación.

Palabras clave

Android, imágenes, procesamiento en tiempo real, sistema embebido.

* Ph. D. (c). Ingeniería. Ingeniero Eléctrico. Profesor Universidad Distrital Francisco José de Caldas. E-mail: fhmartinezs@udistrital.edu.co

** M. Sc. Ingeniería Electrónica y Computadores. Ingeniero en Control Electrónico e Instrumentación. Profesor Universidad Distrital Francisco José de Caldas. fmartinezs@udistrital.edu.co

*** Especialista en Telecomunicaciones. Ingeniero en Control Electrónico e Instrumentación. Profesor Universidad Distrital Francisco José de Caldas. E-mail: hmontiela@udistrital.edu.co

Abstract

This research paper attacks the problem of autonomous robot navigation based on artificial vision systems in real-time. As a first approach to a computer vision system for navigation of the robot, we have decoupled the sensor module and implemented it as a separate unit on a small differential drive robot. On it, we have carried out basic navigational studies for visual identification of landmarks (special marks) in an unfamiliar environment. Performance studies conducted both through simulation and by direct evaluation on prototypes in the laboratory, demonstrate the effectiveness of the proposed structures of both the hardware for real-time identification as the basic navigation strategy considered for evaluation.

Keywords

Android, embedded system, images, real-time processing.

I. Introducción

El diseño y puesta en funcionamiento de robots, sistemas mecánicos autónomos al servicio del ser humano, desde el punto de vista teórico puede verse conceptualmente muy intuitivo, pero su implementación real en la práctica puede llegar a ser bastante compleja, en particular si el diseño incluye el acople de múltiples objetivos (normalmente en conflicto entre sí) tales como criterios de control, estabilidad, autonomía y bajo consumo de energía [1], [2].

El problema de la navegación autónoma de robots móviles es actualmente de gran actividad investigativa [3], [4], [5]. De acuerdo a las exigencias requeridas en una determinada aplicación, es posible tener diferentes enfoques: unidad central de control o agentes inteligentes autónomos, un solo robot o múltiples agentes coordinados, ambientes estáticos o con obstáculos en continuo movimiento, ambientes desconocidos, observables o conocidos. Cada uno de estos enfoques plantea condiciones particulares de diseño.

En el caso de la navegación autónoma multiagente en ambientes dinámicos observables, la coordinación y comunicación entre agentes cobra particular importancia, y es lo que se busca resolver en esta investigación con un sistema de visión artificial [6].

El conjunto de técnicas y procesos utilizados para extraer información de interés de una imagen, utilizando como herramienta un sistema digital se conoce como procesamiento digital de imagen [7]. Este concepto se puede extender a nuevas herramientas tecnológicas, como es el caso de los sistemas embebidos, y es para la investigación de particular interés el caso de los *smartphones* y las tabletas electrónicas. Los sistemas de imagen en tiempo real normalmente procesan una cadena de imágenes digitales, las analizan, y controlan alguna parte del sistema de acuerdo al resultado de este análisis. Esto es de particular importancia en aplicaciones de control [8], [9].

A fin de enfocar adecuadamente el problema, en esta investigación se ha desacoplado el sistema de identificación visual, y se ha trasladado a una plataforma con dinámica

simple, un robot diferencial. Esto permite atacar el problema de identificación como tal desde su base, analizando y estudiando algoritmos que evolucionarán con el avance de la investigación.

Uno de los propósitos básicos de la visión artificial en robótica, es permitir la navegación autónoma de los agentes. Este problema fundamental de la robótica se ha tomado como excusa para la implementación de esquemas minimalistas que permitan resolver problemas reales relacionados con la coordinación de múltiples robots móviles autónomos en tareas de navegación, sin la necesidad de implementar identificación de sus sistemas, generación del mapa geométrico de los ambientes, localización o estimación de estados.

En lugar de utilizar estas herramientas, el grupo de investigación ha buscado un enfoque descentralizado que implique el menor uso posible de información, tanto a nivel de la dinámica del sistema de robots, como a nivel interno de los agentes. En este enfoque, un conjunto de robots realizan un conjunto de movimientos *sin control* (*wild motions* [10]) a lo largo de los límites de su ambiente.

Estos robots se han equipado con sensores básicos, los cuales les permiten extraer información mínima del ambiente (solamente la información necesaria para la interacción): un sensor de impacto y un sensor visual. El sensor de impacto le sirve para identificar los límites del ambiente, y el sensor visual le sirve para establecer una mínima comunicación local (el esquema no procesa ni transmite información de estados).

La coordinación de los robots, que en últimas permite la realización de las tareas asignadas, se logra gracias a la información visual capturada por cada robot, y a un conjunto de reglas de comportamiento inspiradas en modelos biológicos. Utilizando este esquema, es posible demostrar que el sistema conformado por los robots es capaz de resolver tareas complejas, como lo son por ejemplo la exploración de un ambiente o el agrupamiento autónomo de los agentes. El análisis realizado es inicialmente soportado por experimentos en el mundo real, y luego analizado a través de simulaciones.

II. Formulación del problema

Sea $W \subset \mathbf{R}^2$ la clausura de un conjunto abierto contráctil en el plano que tiene un interior abierto conectado con obstáculos que representan regiones inaccesibles. Sea el espacio libre en el ambiente, que es el subconjunto abierto de sin los obstáculos (con los obstáculos removidos).

Asuma un conjunto de agentes autónomos en este espacio libre con capacidad de desplazamiento e interacción local con el ambiente a través de sensores. Los agentes descubren y conocen el ambiente en el que se mueven a partir de observaciones, utilizando sensores. Estas observaciones les permiten construir un espacio de información . Un mapeo de información es de la forma:

$$q: E \longrightarrow O \qquad (1)$$

Donde O denota un *espacio de información*, construido a partir de las lecturas del sensor a lo largo del tiempo, es decir, a través de una historia de observación de la forma:

$$\tilde{o}: [0, t] \longrightarrow O \tag{2}$$

La interpretación de este espacio de información, es decir, , es la que le permite al agente tomar decisiones [11].

Considere ahora a un grupo de agentes, robots diferenciales, cuya cinemática está definida por:

$$\begin{aligned} \dot{\mathbf{z}_i} &= \mathbf{u}_i \\ \mathbf{z}_i &= (x_i, y_i, \theta_i) \end{aligned} \tag{3}$$

con:

$$\begin{aligned} \dot{x_i} &= v_i cos\theta_i \\ \dot{y_i} &= v_i sen\theta_i \\ \dot{\theta_i} & \end{aligned} \tag{4}$$

donde $(x_i, y_i) \epsilon \mathbf{R}^2$ es la posición del robot, i^{th} es su orientación, y v_i y $\dot{\theta_i}$ son respectivamente las velocidades controladas traslacional y rotacional. $\mathbf{u}_i$ denota la entrada de control del agente.

Se asume que los agentes son capaces de sensar las cercanías de sus compañeros y/o los obstáculos en el ambiente utilizando una mínima cantidad de información visual. Por lo tanto, la vecindad de $\mathbf{z}_i$ está dada por el rango y el campo de visión del hardware sensor.

El objetivo es desarrollar el mínimo hardware visual y extraer de las imágenes la mínima cantidad de información necesaria para desarrollar exitosamente tareas de navegación colectiva en un ambiente dinámico y desconocido.

III. Metodología

Para el desarrollo de la investigación se consideraron tres elementos claves:

- El hardware de diseño
- El esquema de control
- Los filtros digitales requeridos

El hardware de procesamiento constituye un primer elemento clave. El perfil inicial considera operación en tiempo real al mismo tiempo que impone minimalismo y bajo costo. Además, el desempeño de este hardware se encuentra íntimamente ligado al software de análisis, núcleo del segundo elemento clave, que va de la mano también con el tipo de información que se requiere extraer de las imágenes capturadas. La extracción de esta información se soporta fundamentalmente en un pequeño conjunto de filtros digitales.

A. Hardware de diseño

Un sistema embebido es un circuito electrónico soportado por una unidad de procesamiento y diseñado específicamente para desarrollar una determinada tarea [12]. La programación embebida o dedicada permite desarrollar instrucciones en funciones específicas con un alto desempeño, en comparación a un hardware de propósito general (por ejemplo, un computador personal). Los sistemas embebidos se encuentran en la mayoría de los equipos digitales de la vida actual, desde celulares hasta cámaras y sistemas de inyección electrónica de vehículos.

Con la alta disponibilidad de plataformas embebidas, los verdaderos elementos de selección se orientan hacia las herramientas de desarrollo, la disponibilidad y el costo final. En este sentido, en el mercado es posible encontrar sistemas operativos (OS) para sistemas embebidos como OS/2 (eComStation), Windows CE, Windows EmbeddedAutomotive, OSEK, FreeBSD, Android, Symbian, etc. Dada la necesidad de acceso abierto a las herramientas, y la amplia disponibilidad de hardware de prueba en el mercado (kits de desarrollo y celulares inteligentes), se seleccionó como OS del sistema a Android [13].

El sistema de desarrollo (*Software Development Kit*, SDK) para Android suministra las herramientas y las interfaces para programación de aplicaciones (*ApplicationProgramming Interfaces*, APIs) necesarias para el desarrollo con Java [13]. Android Maneja los recursos del hardware de forma transparente para el usuario [14]. La figura 1 ilustra los componentes de la estructura por capas que son parte de la arquitectura interna de Android [12].

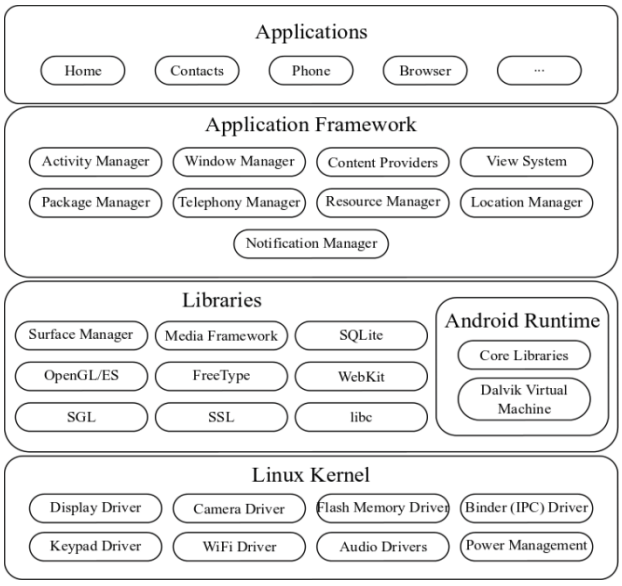


Figura 1. Diferentes capas de la arquitectura de Android OS [12].

Las plataformas hardware utilizadas para el diseño en laboratorio y las pruebas sobre prototipos fueron:

- Development board Real6410 (ARM11, 256 MB mDDR SDRAM, 1 GB Nand Flash, Android 2.1).
- Smartphone Motorola MB501 Quench (ARM 11, 512 MB ROM, 256 MB RAM, Android 2.3.7).
- Smartphone Samsung SCH-R720 Vitality (ARM 11, 512 MB RAM, 196 MB ROM, Android 2.3).

B. Regiones en imágenes binarias

La clave del procesamiento de imagen en la investigación radica en la identificación y extracción de regiones en imágenes binarias [7]. En una imagen binaria, cada uno de los pixeles toma solo uno de dos posibles valores. Esta modificación de las imágenes permite separar el fondo de la imagen del primer plano de interés. El procesamiento requerido entonces sobre la imagen se centra en la identificación de las regiones en la imagen (figura 2, cada objeto corresponde a una región).

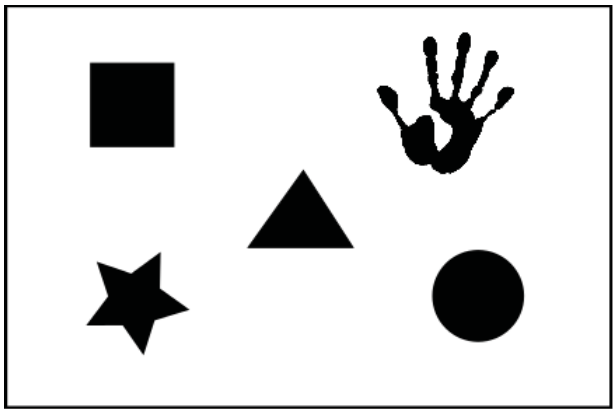


Figura 2. Imagen binaria con cinco objetos.

La identificación de las regiones implica primero determinar sobre la imagen qué pixeles pertenecen a las regiones de interés. Segundo, determinar el número de regiones en la imagen. Y tercero, localizar estas regiones en la imagen. Estos tres procesos en conjunto se conocen como *etiquetado de regiones* o *coloreado de regiones*.

Este proceso de etiquetado opera de pixel a pixel, y se puede realizar de diferentes formas. Una primera estrategia consiste en inundar regiones hacia todas las direcciones a partir de un primer punto o *semilla*, otra estrategia consiste en hacer un barrido de la imagen del extremo superior al inferior. En cualquier caso, se comparan los valores entre pixeles, y de acuerdo a su cercanía, se clasifica al nuevo pixel como perteneciente a la región o no. Al final, a cada región se le asigna una única etiqueta.

Las imágenes binarias resultantes del proceso de etiquetado, denominadas (y denotan las coordenadas de cada pixel), utilizan valores de cero para el fondo y de uno para el primer plano (regiones). En el caso del rastreo en colores, también se utilizan otros valores para el etiquetado de las diferentes regiones, es decir (ecuación 5):

$$T(p_x, p_y) = \begin{cases} 0 \text{ para el fondo} \\ 1 \text{ para pixeles del primer plano} \\ 2, 3, 4, \ldots \text{ para etiquetdo de regiones} \end{cases} \quad (5)$$

C. Diseño de filtros

Una vez la imagen es capturada por la cámara del hardware, la identificación de las regiones y la extracción de la información requerida para navegación es realizada mediante la aplicación de ciertos filtros. Un filtro es un proceso por medio del cual una señal es alterada, en este caso cada pixel de la imagen. El filtro se aplica alterando la magnitud de los componentes RGB en sus correspondientes matrices.

Los filtros se implementaron completamente en Java sin el uso de librerías para procesamiento de imágenes. Esta característica permite definir su implementación como *soft real-time*, bajo los límites que confiere un lenguaje de alto nivel como Java. Esto también diferencia el diseño de otras aplicaciones similares.

1. Filtro escala de grises

En los casos de navegación por identificación geométrica de *landmarks*, uno de los primeros filtros aplicados a las imágenes fue el de escala de grises. Para su aplicación se utilizó la ecuación de luminiscencia (ecuación 6) [15]:

$$Y = (0{,}299 * R + 0{,}587G + 0{,}114B) \quad (6)$$

Donde R, G y B son los valores de cada pixel, y es la nueva imagen filtrada. La figura 3 muestra los resultados de aplicar este filtro sobre una imagen prediseñada.

(a) Imagen original. (b) Imagen filtrada

Figura. 3. Desempeño filtro escala de grises sobre board Real6410.

2. Filtro de separación de componentes RGB

Este filtro es fundamental en los casos de navegación por identificación de *landmarks* por colores. El cálculo es también mucho más simple. Su aplicación consiste en ajustar a 0 los componentes no deseados de la imagen. La figura 4 muestra los resultados de aplicar este filtro sobre una imagen prediseñada.

Figura. 4. Desempeño filtro separación RGB sobre board Real6410.

3. Filtro inversión de color

Este filtro transforma el color de cada pixel en su inverso. Cada canal RGB contiene valores de 0 a 255, donde 0 es el valor más oscuro, y el valor 255 es el más iluminado. Para el cálculo de los nuevos valores se resta 255 al valor de cada pixel (ecuación 7):

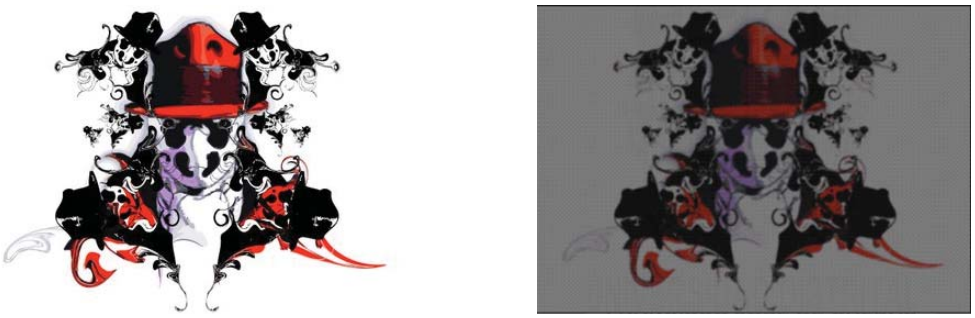
$$\begin{aligned} R &= 255 - R \\ G &= 255 - G \\ B &= 255 - B \end{aligned} \tag{7}$$

La figura 5 muestra los resultados de aplicar este filtro sobre una imagen prediseñada.

Figura. 5. Desempeño filtro inversión de color sobre board Real6410.

4. Filtro de suavizado Blur

Este filtro de suavizado es en realidad un promedio de valores, y es utilizado para remover ruido de las imágenes. Su implementación consiste en generar una máscara o *kernel* para cada pixel a fin de cambiar su valor. Para su evaluación, el valor de cada pixel se dividió por 9, y el resultado de este suavizado para una segunda imagen de prueba se muestra en la figura 6.



(a) Imagen original. (b) Imagen filtrada

Figura 6. Desempeño filtro de suavizado Blur sobre board Real6410.

5. Filtro Sobel-Prewitt

La función más importante de este filtro es la detección de bordes dependiendo de la dirección o posición. En la implementación realizada de este filtro se detectaron los bordes tanto del eje x como del y [15].

El filtro se basa en los gradientes de los pixeles. Analizando el comportamiento del gradiente es posible detectar los bordes de acuerdo a las derivadas espaciales en la imagen. Estas derivadas se calculan por medio de convolución. La convolución es un barrido que no altera los operandos, que son el *kernel* (ahora en la forma de una matriz) y la imagen original [7]. El resultado de este barrido o interacción es la imagen filtrada.

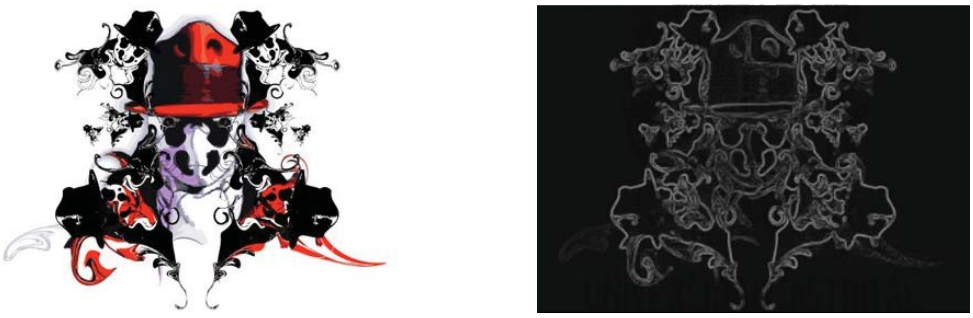
El *kernel* utilizado en forma matricial sobre el eje es (ecuación 8):

$$\begin{matrix} -3 & 0 & -3 \\ -10 & 0 & 10 \\ -3 & 0 & 3 \end{matrix} \tag{8}$$

Y sobre el eje es (ecuación 9):

$$\begin{matrix} -3 & -10 & -3 \\ 0 & 0 & 0 \\ 3 & 10 & 3 \end{matrix} \tag{9}$$

En estas matrices la fila o la columna con ceros es la responsable de dividir la imagen. La figura 7 muestra el resultado del filtro para la imagen de prueba.



(a) Imagen original. (b) Imagen filtrada

Figura 7. Desempeño filtro Sobel-Prewitt sobre board Real6410.

IV. Resultados

La solución al problema de la diferenciación entre regiones implica en primer lugar la utilización de filtrado morfológico [7]. Esta técnica de análisis de estructuras geométricas consiste en un conjunto de operadores (erosión, dilatación, apertura y cierre) que transforman las imágenes. Sin embargo, aplicar esta técnica en tiempo real sobrepasa la capacidad de procesamiento utilizada en los prototipos iniciales.

Aumentar la capacidad de procesamiento sin dejar de lado el concepto de minimalismo y costo final del sistema, llevó a la búsqueda de plataformas de bajo costo, abiertas y de fácil integración a los robots. La solución seleccionada fue el uso de teléfonos inteligentes o *smartphones* basados en Android [16]. El calificativo de inteligente hace referencia a la capacidad de usarse como un computador de bolsillo, en algunos casos con capacidad de procesamiento similar a la de computadores portátiles. Estos dispositivos vienen equipados con cámara, micrófono, GPS, acelerómetro y otros sensores, en conjunto con un sistema abierto para el desarrollo de aplicaciones basado en Java. Algunos de estos dispositivos se pueden conseguir hoy en día nuevos desde 100 dólares, y ya los usados comienzan a inundar el mercado.

Para las pruebas con los robots se utilizaron en dos de estos teléfonos:

- Motorola MB501 Quench
- Samsung SCH-R720 Vitality

El código escrito directamente en Java utiliza la cámara del smartphone y procesamiento de imagen en tiempo real para calcular las diferentes matrices a partir de la información RGB. Una demostración de este código realizando la identificación de regiones en la imagen, y su etiquetado, se observa en la figura 8. La prueba se realizó con *landmarks* de color amarillo de un tono particular, etiquetando la imagen de color negro. Para depuración, se agregó en la esquina superior derecha de la imagen el conteo de pixeles en la región identificada. El vídeo de este comportamiento puede ser observado en: http://youtu.be/-NidWcidpZM. Se debe notar que el *landmark* se coloca al lado de uno de los robots, y el software logra identificarlo en tiempo real sin importar el movimiento o demás elementos en la imagen.

Figura 8. Identificación en tiempo real de landmark.

Para múltiples regiones se realizó el etiquetado coloreando cada región identificada con un color diferente (negro, rojo, blanco, etc.). El número de colores utilizados informa el número de regiones identificadas en la imagen. La prueba se realizó con diversos bloques geométricos de diferentes formas (cuadrados, rojos y triangulares) y colores (amarillos, verdes y rojos), con la intención de probar que el software era capaz de identificar y etiquetar solo los bloques de color amarillo (figura 9). Luego, a las regiones identificadas, es posible aplicarles análisis morfológico. El vídeo de este comportamiento puede ser observado en: http://youtu.be/kgeoys2AX8I.

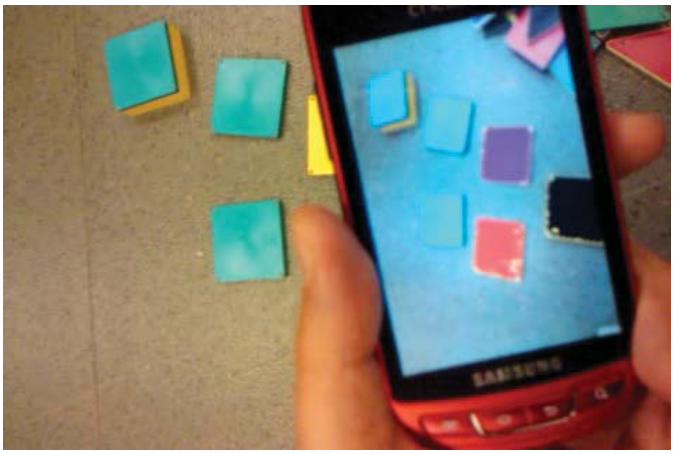


Figura 9. Etiquetado por colores en tiempo real.

El smartphone, controlado a través de Android, posee los sensores del robot (inicialmente la cámara). La otra parte importante de la integración del smartphone al robot diferencial es la comunicación para activar los actuadores, es decir, las señales que se deben enviar a los servomotores. Nuevamente, teniendo presente los criterios de sencillez, minimalismo, robustez y bajos costos, se optó por utilizar la salida de audio del smartphone a través de su *headsetjack* de 3.5 mm. Esta salida es estándar, está presente en todos los smartphone, se puede controlar directamente desde el código Java, y da la posibilidad de tener hasta cuatro señales diferentes con un ancho de banda de hasta 20 kHz.

El prototipo de tarjeta de comunicación transmite solo dos señales del smartphone hacia los servomotores. Estas señales corresponden a los semiciclos positivos de las dos señales de audio (audio estéreo). Inicialmente, los semiciclos negativos no se utilizan. La tarjeta utiliza un comparador de voltaje de dos canales y alimentación sencilla, el LM319, y un opto-aislador de dos canales, el LTV826, para garantizar la seguridad del smartphone frente a eventuales fallas del circuito.

El funcionamiento del software de reconocimiento y la comunicación con el robot diferencial se evaluó inicialmente con una sencilla tarea de navegación. Se programó el robot para buscar en el ambiente los *landmark* de color amarillo, y una vez identificados, dirigirse hacia ellos (figura 10). Una vez el robot está cerca del *landmark*, para e inicia la búsqueda de otro *landmark*. Este comportamiento puede observarse en: http://youtu.be/LWs9D2S8w2g.

Figura 10. Navegación sencilla por landmark de colores.

Dada la dificultad de realizar experimentos con un gran número de robots, a fin de analizar el comportamiento colectivo en un ambiente dinámico, se optó por realizar este análisis a través de simulaciones. Las simulaciones fueron desarrolladas en Player/Stage [17]. Esta plataforma permitió que la simulación ejecutara exactamente el mismo código en C desarrollado para el robot real. Además, estas simulaciones no solo consideran la funcionalidad del algoritmo, ellas tienen en cuenta también las características físicas del robot, es decir: forma, tamaño (0.26 m x 0.22 m), peso (1.2 kg) y colores. La cámara (smartphone) se instaló justo sobre su centro geométrico tanto en el prototipo real como en el modelo de simulación.

El ambiente de simulación fue diseñado de forma cuadrada, con características similares al laboratorio real, y con un área total de . Dentro del ambiente, se colocaron tres agujeros inaccesibles para los robots como obstáculos para la navegación y para los sensores. En las simulaciones se utilizó un total de 10 agentes, todos completamente independientes entre sí, reportando sus señales individualmente al servidor de Player.

Para la simulación se consideró una tarea básica de agrupamiento bioinspirada. Los individuos esquivan los obstáculos y tratan de acercarse a los grandes grupos, identificándolos por color, cantidad y movimiento, además considera algún nivel de aprendizaje [18]. Se inició la simulación con los robots dispersos en el ambiente. Se inició la simulación en t = 26 s, y el caso documentado aquí terminó a los 12 min. y 16 s (tiempo transcurrido al interior de la simulación, figura 11). Esta simulación mostró comportamientos interesantes. Por ejemplo, se observaron primeros signos de agrupamiento de dos y/o tres agentes a lo largo de toda la simulación, y un comportamiento que se asemeja a un baile entre dos robots a los 6 min. y 42 s que duró un total de 38 s. Al final de esta simulación se observó un grupo bastante estable de robots en la parte inferior derecha del ambiente. La simulación en Player/Stage puede ser observada en http://youtu.be/Lc4DUSW-BFo.

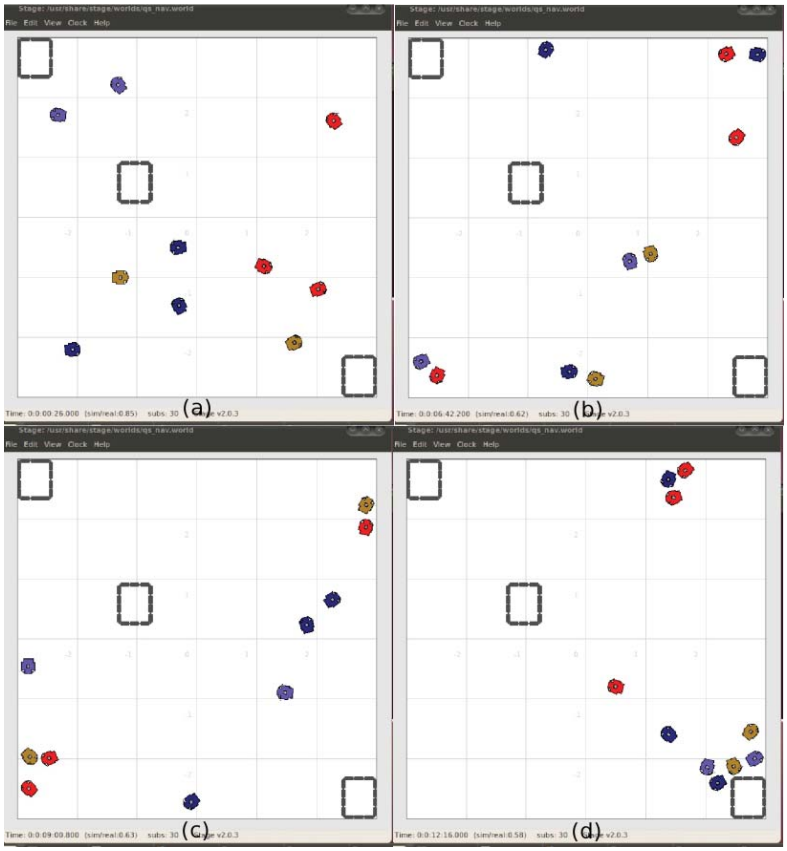


Figura 11. Simulación tarea de agrupamiento. a) Estado inicial de la simulación. b) Los dos robots en el centro (azul y dorado) inician un baile que dura por 38 segundos. c) A los 9 min. se observa la formación de pequeños grupos de robots tratando de estar juntos. d) Estado final de la simulación, los robots han establecido un grupo estable en la parte inferior derecha del ambiente.

V. Conclusiones

En este artículo se presenta el concepto de un sensor óptico minimalista soportado en herramientas embebidas de visión artificial en tiempo real para la navegación autónoma de robots. Sobre un procesador ARM 11 con Android se evaluaron e implementaron un total de cinco filtros para procesamiento de imágenes. Los filtros fueron codificados directamente en Java a modo de *soft real-time*, es decir, sin el uso de librerías de procesamiento como es el caso de OpenCV. Este tipo de implementación dedicada permitió reducir considerablemente las exigencias hardware, y por tanto, fue posible utilizarlos en esquemas simples de navegación autónoma.

A lo largo de los experimentos en laboratorio y las simulaciones en Player/Stage, se logró mostrar que un grupo de robots pueden ser utilizados para resolver tareas de navegación colectiva, utilizando un simple y minimalista diseño en hardware, software y algoritmo de operación. Los robots en los experimentos conformaron un sistema complejo que emuló algunas características típicas de enjambres de organismos vivos, y bajo este esquema, coordinaron sus movimientos utilizando solamente información local del medio, en particular, información visual.

El enfoque minimalista empleado demuestra incluso que, pese a tratarse de una aplicación de reconocimiento visual con cámaras digitales, no se tienen problemas de invasión de privacidad, dado que la información que se toma y procesa es tan solo la mínima necesaria para cumplir la tarea, no se transmite, y no se mantienen registros de ella.

Referencias

[1] R. Lenain, B. Thuilot, C. Cariou y N. Bouton, «Off-road mobile robots control: An accurate and stable adaptive approach,» en 2012 2nd International Conference on Communications, Computing and Control Applications (CCCA), 2012, pp. 1-6.

[2] M. Sawada y K. Itamiya, «A position control of 2 DOF flexible link robot arms based on computed torque method», en 2012 International Conference on Advanced Mechatronic Systems (ICAMechS), 2012, pp. 547-552.

[3] G. Tuna, y K. Gulez, «Aided navigation techniques for indoor and outdoor unmanned vehicles», en 2012 5th International Conference on New Technologies, Mobility and Security (NTMS), 2012, pp. 1-4.

[4] N. Sato, K. Kon y F. Matsuno, «Navigation interface for multiple autonomous mobile robots with grouping function», en 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics (SSRR), 2011, pp. 32-37.

[5] A.B. S. Saman, P. Sebastian, N. A. Malek y N. Z. M. Hasidin, «Implementation of autonomous vehicle navigation algorithms using event-driven programming», en 2012 4th International Conference on *Intelligent and Advanced Systems* (ICIAS), Vol.1, 2012, pp. 12-17.

[6] E. Montijano, J. Thunberg, X. Hu y C. Sagues, «Epipolar Visual Servoing for Multirobot Distributed Consensus», en *IEEE Transactions on Robotics*, Vol. 29, Issue 5, 2013, pp. 1212-1225.

[7] W. Burger y M. Burge, Digital Image Processing: An Algorithmic Introduction using Java, 1st ed., Springer, ISBN 978-1846283796, 2008.

[8] S. Karungaru, M. Sugizaki y M. Fukumi, «Biped robot walking control using image processing», en ICCAS-SICE 2009, 2009, pp. 4020-4024.

[9] X. Zeng, G. Zhao, Y. Wang, C. Wang, y J. Wang, «Research on the elevator door control system based on the image processing technology», en 2010 International Conference on Electrical and Control Engineering (ICECE 2010), 2010, pp.1781-1784.

[10] L. Bobadilla, O. Sánchez, J. Czarnowski, y S. M. LaValle, «Minimalist multiple target tracking using directional sensor beam», en International Conference on Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ, pp. 3101-3107.

[11] S. M. LaValle, Planning Algorithms. Cambridge, U.K.: Cambridge University Press, 2006. Available at http://planning.cs.uiuc.edu/

[12] G. Galeano, Programación de sistemas embebidos en C. 1.ra edición, Editorial Alfaomega, 2009.

[13] F. Khomh, Hao Yuan, Ying Zou, «Adapting Linux for mobile platforms: An empirical study of Android», 28th IEEE International Conference on Software Maintenance (ICSM), 2012, pp. 629-632.

[14] M. Polanco, B. Taibo, y J. Luis, «Android el sistema operativo de Google para dispositivos móviles», *Revista Negotium*, 7, 2011, pp. 79-96.

[15] R. González, y R. Woods, *Digital Image Processing*. 3.ª edición, Editorial Pearson, 2007.

[16] D. Serfass, «Wireless sensor networks using android virtual devices and near field communication peer-to-peer emulation», en 2012 *Proceedings of IEEE Southeastcon*, 2012, pp. 1-6.

[17] B. Gerkey, R. T. Vaughan y A. Howard, «The player/stage project: Tools for multi-robot and distributed sensor systems», en Proceedings IEEE 11th International Conference on Advanced Robotics ICAR'03, 2003, pp. 317-323.

[18] F. H. Martínez y A. Delgado, «Hardware emulation of bacterial quorum sensing», en Proceedings of the 6th international conference on Advanced intelligent computing theories and applications: intelligent computing ICIC'10, 2010, pp. 329-336.