Navegación de robot móvil usando Kinect, OpenCV y Arduino

Mobile robot navigation using Kinect, OpenCV and Arduino.

César Augusto Díaz Celis[1], César Augusto Romero Molano[2].

[1]Magister (c) en Sistemas de Información Geográfica, Ingeniero de Sistemas, Docente Tiempo Completo Programa Ingeniería de Sistemas, Universidad de los Llanos, Meta – Colombia
Cesar.diaz@unillanos.edu.co
[2]Especialista en Redes de Datos, Ingeniero Electrónico, Docente Tiempo Completo
Programa Ingeniería Electrónica, Universidad de los Llanos, Meta – Colombia
Cesar.romero@unillanos.edu.co

Recibido 25/11/11, Aceptado 25/06/2012

RESUMEN

Este artículo presenta los resultados de investigación de la visión artificial que sirve de apoyo a la navegación por medio de imágenes de profundidad y el reconocimiento de objetos por sus canales primarios. El dispositivo utilizado para la captura de la imagen RGB y la imagen de profundidad es el sensor Kinect de Microsoft, este consta de una cámara RGB y un emisor de infrarrojos que proyecta un patrón irregular de haces con una intensidad variable. El sensor de profundidad reconstruye una imagen a partir de la distorsión del patrón, el siguiente paso es buscar todos los puntos rojos de la escena, contarlos, calcular el centroide, diámetro, posición y distancia al Kinect. Por último procesar estos resultados, tomar la decisión de movimiento para ser enviada al Arduino, el cual controla los motores. Los resultados obtenidos en la investigación indican que las imágenes de profundidad capturadas por el Kinect requieren de escenarios con iluminación controlada; este aspecto es compensado con la creación del algoritmo de navegación con procesamiento digital de imágenes.

Palabras clave: Kinect, OpenCV, Robot Móvil, Visión Estéreo, Visión RGB.

ABSTRACT

This paper presents the research results of the artificial vision that support navigation through depth images and recognition of objects by their primary channels. The device used for RGB and depth image capture is Microsoft's kinect sensor, which has a RGB camera and an infrared transmitter that projects an irregular beam pattern with a variable intensity. This depth sensor reconstructs an image from the distortion pattern. The next step is to find all the red points of the scene, count them, calculate the center point, diameter, position and its distance from the kinect. Finally, process these results, take the movement decision to be sent to the Arduino, which controls the motors. The results obtained in the research indicate that the depth images captured by the kinect require controlled illumination scenarios; this aspect is compensated with the creation of the navigation algorithm with digital image processing. The results indicate that research showed the depth images captured by the Kinect require controlled lighting scenarios, for this reason we created the navigation algorithm with digital image processing which serves to support the navigation depth images.

Keywords: Kinect, OpenCV, Mobile Robot, Stereo Vision, RGB Vision.

1. INTRODUCCIÓN

Para el control de la navegación de un robot móvil existen infinidad de alternativas una de estas alternativas es la utilización de la visión artificial por lo cual se planteó el interrogante ¿Es posible navegar en un ambiente parcialmente controlado usando solo visión estéreo?

Uno de los requerimientos indispensables para un robot móvil es poseer la habilidad de navegar en el medio que lo rodea, en este campo existen variedad de trabajos como: algoritmos de navegación reactiva [1] [2], planificada [3], hibrida [4], entre otros.

El kinect nace como un dispositivo para juegos, pero en la actualidad son infinitas las aplicaciones que se le pueden dar no solo en la industria de los videos juegos si no también aplicado en áreas como la robótica, medicina, industria, entre otros. Ejemplo de ello son las investigaciones en áreas como la identificación y seguimiento de personas analizando la interacción hombre maquina [5, 6], navegación reactiva de un robot con información procedente de sensores 2D [7], cinemática en lugares cerrados [8], detección de objetos por color y profundidad [9], entre otros.

El robot móvil utilizado para el desarrollo de la investigación es un prototipo desarrollado bajo el concepto de tracción diferencial no comercial, lo cual le permite ser fácilmente adaptable a cualquier situación que se ajuste a la dinámica del robot.

Aprovechando que el Sensor Kinect captura tanto imágenes de profundidad como imágenes RGB lo cual permite mejorar la navegación reactiva [10], se toma la iniciativa de trabajar con este nuevo dispositivo y utilizar herramientas de hardware y software libre.

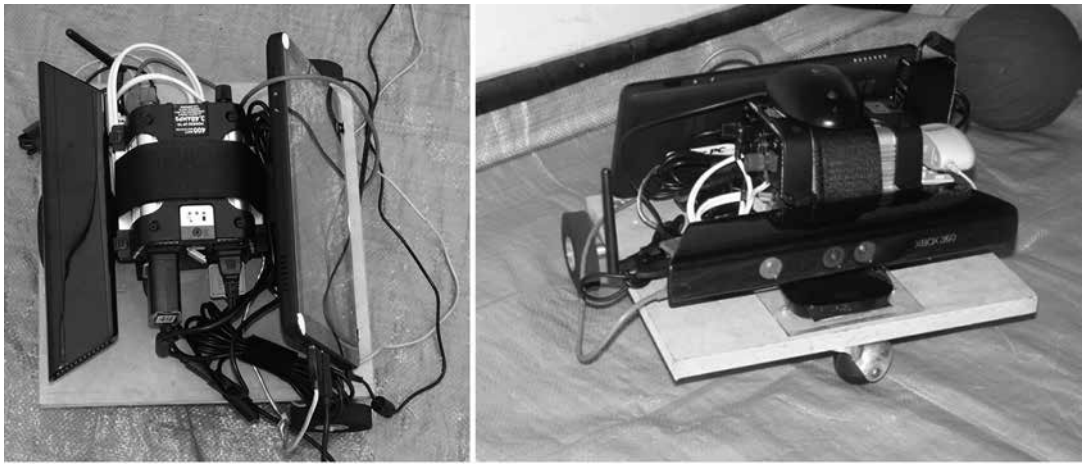
2. METODOLOGÍA

Para el control de la navegación del robot móvil se desarrollaron tres bloques con funciones definidas las cuales son: adquisición de la imagen de la escena utilizando el sensor kinect por medio de la librería Libfreenect [11], procesamiento de la imagen en procesador IA-32 usando OpenCV[12], comunicación procesador IA-32 por medio de un puerto RS-232 y control del robot móvil, el prototipo usado es el que se observa en la Figura 1, las etapas de la metodología del sistema de navegación con visión artificial se encuentran en la Figura 2 y el diagrama de bloques de la aplicación y del prototipo final en la Figura 3.

2.1 Adquisición de imágenes

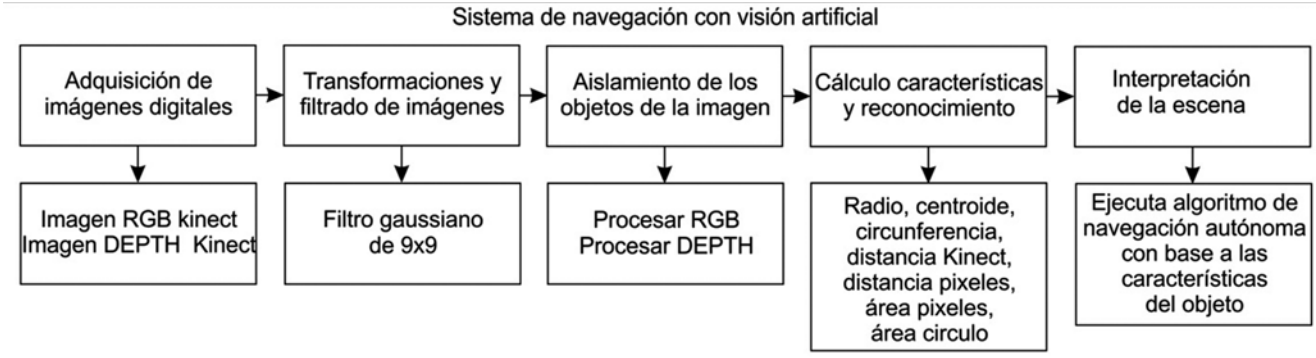
Para la adquisición de las imágenes se usó el sensor kinect de la empresa Microsoft, el cual es un dispositivo hardware formado por: un hub USB genérico, Xbox motor nui, Xbox cámara nui y Xbox audio nui, los cuales en conjunto forman el sensor de movimiento kinect, inicialmente desarrollado para trabajar con la consola de juegos Xbox de Microsoft. El hub USB genérico proporciona la comunicación entre el hardware (motor, cámaras y audio nui) con el computador el cual puede tener instalado como sistema operativo Windows o Linux para el caso particular de esta aplicación se usó el sistema operativo Linux con la distribución de Ubuntu 10.04 [13]. El Xbox motor nui es el encargado de controlar el ángulo de inclinación del dispositivo kinect el cual proporciona una variación en la panorámica de visión en el rango de +31° y -31°.

Figura 1. Prototipo robot terminado
Figure 1. Final robot prototype



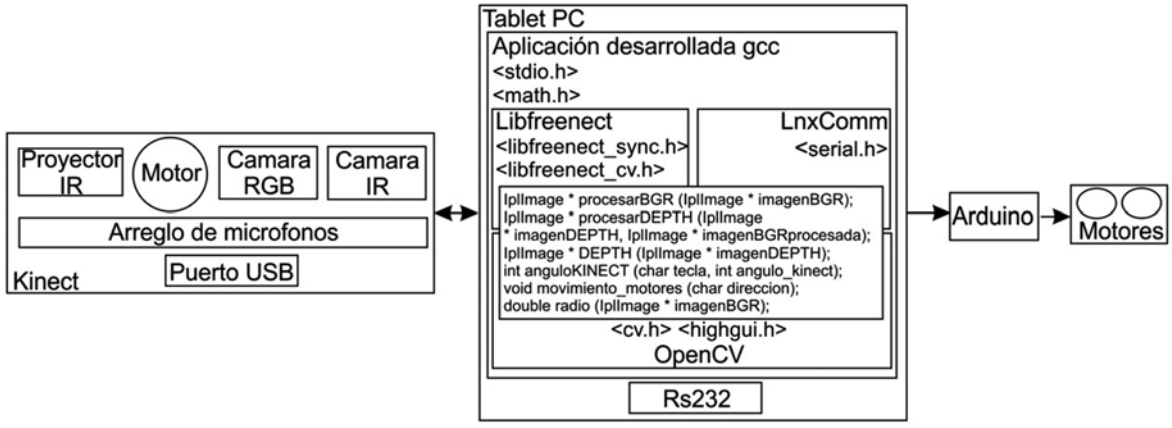
Fuente: Los autores.

Figura 2. Etapas del sistema de navegación con visión artificial
Figure 2. Stages of the navigation system with vision



Fuente: Adaptado de [14] por los Autores del proyecto

Figura 3. Diagrama de bloques del prototipo
Figure 3. Block diagram of the prototype



Fuente: Los autores

La Xbox cámara nui es el elemento de hardware principal del kinect ya que proporciona una imagen de color RGB como la observada en la Figura 4, y una imagen de mapa de profundidad la cual entrega información de distancia sobre un objeto que se encuentre entre un mínimo de 40cm y un máximo de 180 cm de distancia del sensor kinect como la observada en la Figura 5, el paquete de datos de información es de 320pixelesx240pixelesx16bit para trabajar una imagen de 640x480 pixeles con tres canales (RGB).

Finalmente el Xbox audio nui es el hardware encargado de proporcionar la información de cuatro micrófonos los cuales cancelan el ruido proveniente del exterior para crear compatibilidad con una habitación tipo 3D este elemento hardware no es usado en este proyecto de investigación.

2.2 Procesamiento de las imágenes en procesador IA-32

Para el procesamiento de las imágenes RGB y el mapa de profundidad entregado por el sensor kinect se utilizó el driver libfreenect el cual permite la comunicación entre el sensor y el computador, las imágenes obtenidas se procesan accediendo a los datos de dos matrices que son la matriz RGB (640 x 480 x 3) y la matriz de profundidad (640 x 480 x 1), para este procesamiento se usó la librería OpenCV descrita en [12] para lo cual se desarrollaron cuatro funciones en lenguaje GCC las instrucciones usadas se estudian en [15] [16].

2.2.1. Función procesarBGR

Esta función recibe una estructura básica de OpenCV (IplImage) la cual contiene la información capturada por el sensor; a las imágenes RGB capturadas se les aplica un filtro gaussiano de 9x9 para eliminar el ruido de la escena, este se detalla en [17], con base en esta estructura ya filtrada se identifica un objeto de color especifico (rojo), para ello se recorren todos los pixeles que forman la imagen, identificando los que pertenecen al objeto, esto se logra verificando el canal (rojo) que es donde deben estar los niveles más altos, para el caso de los otros canales (verde y azul) se verifica que los niveles no sean más altos que los del canal de estudio (rojo), ya que si son muy cercanos al canal (rojo) estos valores podrían corresponder a otro color (ejemplo: morado o naranja).

Al verificar los pixeles validos se realiza un conteo para obtener el área en pixeles, también se realiza la sumatoria de las coordenadas (x, y) para obtener el centroide del objeto y con base en este se calcula el radio, para obtener el contorno del objeto en análisis, ver Figura 6.

2.2.2. Función DEPTH

Es la función original del driver Libfreenect [11], esta se encarga de capturar la imagen de profundidad del sensor, la función no se modificó, el resultado que se obtiene de esta se observa en la Figura 5.

2.2.3. Función procesarDEPTH

En esta función se analiza la imagen de profundidad que captura el Kinect, los pixeles que se tendrán en cuenta son los seleccionados en la función procesarBGR y los que se encuentran en el rango de distancia entre 40 y 100 centímetros, en esta misma función se acumula el valor de distancia de cada pixel analizado, para luego calcular el promedio de distancia según la imagen de profundidad, también se calcula la distancia tomando como punto base el centro inferior de la imagen con respecto al centroide del objeto, luego de esto se calcula el área del circulo utilizando la función longitud_radio, ver Figura 7.

La función tiene implementado dos modos que son: modo Kinect el código se observa en la Figura 8 y modo PDI en la Figura 9, el primero trabaja con los datos que se generan con base a la imagen de profundidad y el segundo con base al procesamiento digital de la imagen RGB [18] [19].

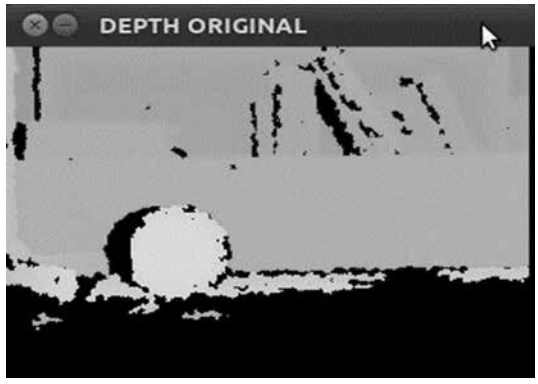
2.2.4. Función radio

En esta función se recorre el objeto partiendo del centroide hacia sus extremos para obtener el diámetro de la circunferencia, luego se hace la diferencia y se divide para obtener el radio.

Figura 4. Imagen RGB proporcionada por el kinect
Figure 4. RGB image provided by the kinect

Fuente: Los autores

Figura 5. Imagen de profundidad proporcionada por el kinect
Figure 5. Depth image provided by the kinect



Fuente: Los autores

Figura 6. Imagen RGB Procesada en GCC con la librería OpenCV
Figure 6. Processed in GCC RGB image with OpenCV library

Fuente: Los autores

Figura 7. Imagen de profundidad procesada en GCC con la librería OpenCV
Figure 7. Depth image processed with the library OpenCV GCC

Fuente: Los autores

Figura 8. Código fuente modo Kinect
Figure 8. Source Kinect mode

```
si (modo_kinect) entonces
        ##parar el carro
        si ((pixeles_objeto>30000 v promedio_distancia<=420) ∧ movimiento ≠ 'p') entonces
                movimiento='p'
        ##ir adelante
        si no ((pixeles_objeto>1000 ∧ promedio_distancia>420) ∧ (centroideX>300 ∧ centroideX<340) ∧ movimiento ≠ 'i') entonces
                movimiento='i'
        ##girar a la izquierda
        si no ((pixeles_objeto<=1000 ∧ centroideX<320 ∧ movimiento ≠ 'j') entonces
                movimiento='j'
        ##girar a la derecha
        si no ((pixeles_objeto<=1000 ∧ centroideX>320 ∧ movimiento ≠ 'l') entonces
                movimiento='l'
        fin si
        movimiento_motores(movimiento)
fin si
```

Fuente: Los autores

Figura 9. Código fuente modo PDI
Figure 9. PDI source mode

```
si (modo_pdi) entonces
        ##parar el carro
        si (longitud_radio>140) entonces
                si (movimiento ≠ 'p') entonces
                        movimiento='p'
                fin si
        ##ir adelante
        si no ((pixeles_objeto>1000 ∧ (centroideX>300 ∧ centroideX<340) ∧ movimiento ≠ 'i') entonces
                movimiento='i'
        ##girar a la izquierda
        si no (centroideX<320 ∧ movimiento ≠ 'j') entonces
                movimiento='j'
        ##girar a la derecha
        si no (centroideX>320 ∧ movimiento ≠ 'l') entonces
                movimiento='l'
        fin si
        movimiento_motores(movimiento)
fin si
```

Fuente: Los autores

2.3. Comunicación procesador IA-32 y robot móvil

2.3.1 Hardware

Se usó para la comunicación entre la computadora y el robot móvil a nivel de hardware el Arduino Uno, el cual cuenta con un ATmega328 ofreciendo una comunicación serial utilizando una UART TTL este puerto serial se refleja en el sistema operativo como un COM virtual. El control del robot se realiza por medio de los pines de entrada salida digital, ver Figura 10. Realizando una comunicación serial a 9600 baudios, 8 bit, sin bit paridad y sin control de flujo del tipo simplex SX.

Para la etapa de potencia en el control de los motores D.C. Del robot móvil se utilizó el PmodHB3 desarrollado por digilent [20] el cual es un puente H de 2 amperios a un máximo de 12 voltios, esté opera con señales en el rango de 2.5 voltios a 5 voltios.

2.3.2 Software

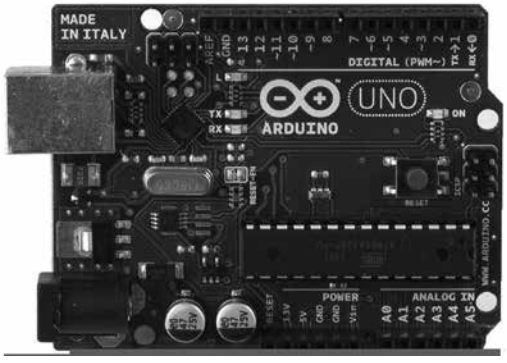
Para la comunicación a nivel de software se utilizó la biblioteca LnxComm, desarrollada por Fernando Pujaico Rivera bajo Copyright© 2011 GPL [21], esta librería permite la conexión serial que se requiere y además por estar diseñada para ser compilada en sistemas operativos Linux y

Windows se acomodó perfectamente a los requerimientos del proyecto.

Para el desarrollo del código que se escribió en el microcontrolador se utilizó el software Arduino alpha el cual es el IDE desarrollado del Arduino Uno, para el código finalmente utilizado consultar [22], las instrucciones y funciones usadas se estudian en [23].

Adicionalmente se crearon dos funciones que son: angulokinect para controlar la orientación del Kinect (arriba, abajo, nivelar) y movimiento_motores la cual recibe la dirección de movimiento de los motores (adelante (i), atrás (k), derecha (l), izquierda (j), parar (p)).

Figura 10. Circuito de comunicación y control
Figure 10. Communication and control circuit



Fuente: Tomado de: http://arduino.cc/en/Main/ArduinoBoardUno

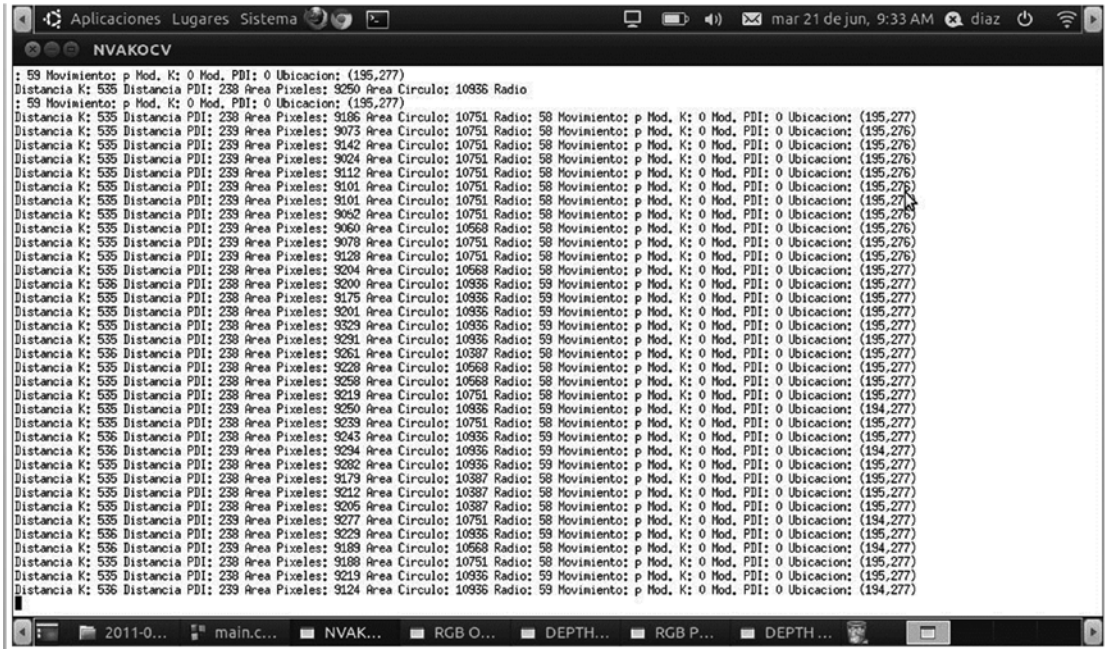
3. RESULTADOS DE LA APLICACIÓN EN VISIÓN ARTIFICIAL PARA LA NAVEGACIÓN DEL ROBOT MÓVIL Y LOS POSIBLES TRABAJOS FUTUROS

Una vez integrados los módulos del prototipo se puede observar que se realizan los cálculos de las características del objeto de forma adecuada, esto se puede apreciar en la Figura 11. Calculando el margen de error del cálculo del area sumando los pixeles validos (color rojo) y el area calculado según el valor del radio se obtiene un margen de error del 14.09% con respecto al area del círculo, está perdida se presenta pos las sombras que se generan en la superficie analizada y por la ubicación de la fuente de luz (luz natural, artificial, sombras, entre otros).

El robot móvil navegaba buscando el objeto de color definido (rojo) hasta una proximidad de 45 centímetros, pero si las condiciones de luz cambiaban el vehículo en algunas ocasiones chocaba con el objeto buscado.

Como trabajos futuros esta la implementación de todo el procesamiento de imágenes en un procesador ARM, como el que dispone la BeagleBoard XM o PandaBoard ES, esto con el fin de diseñar un prototipo de navegación embebido y reducir los costos asociados al hardware usado en esta investigación [24]. También la implementación de soluciones de navegación con visión artificial en vehículos aéreos no tripulados UAV como el ArDrone de la empresa Parrot.

Figura 11. Calculo de características imagen RGB e imagen de profundidad
Figure 11. Calculating RGB image features and image depth



Fuente: Los autores

4. CONCLUSIONES

- Una vez presentado los anteriores resultados y resolviendo la pregunta ¿Es posible navegar en un ambiente parcialmente controlado usando solo visión estéreo? Planteada al inicio del presente trabajo de investigación se concluye que no es completamente eficiente la navegación del robot móvil usando solo visión estéreo ya que las condiciones del entorno (luminosidad, características de color del objeto, entre otros) hacen que el algoritmo de navegación deba ser ajustado permanentemente y es por esto que se decide implementar la navegación usando procesamiento digital de imágenes (modo PDI del carro) y dejar la navegación estéreo como otra alternativa de navegación, de igual forma la visión estéreo es una excelente herramienta para el cálculo de la distancia al objeto y como complemento a la navegación.

- El uso del kinect como sensor de visión estéreo en el presente trabajo de investigación fue una herramienta de muy buenas prestaciones ya que entrega un mapa de profundidad, fácil de usar y aplicar al objetivo del proyecto el cual fue la navegación estéreo olvidando los problemas típicos o propios de la visión estero como son la alineación y sincronización de dos cámaras RGB independientes.

AGRADECIMIENTOS

Este trabajo fue apoyado por la Universidad de los llanos y el instituto de investigaciones de la Orinoquia colombiana IIOC.

REFERENCIAS

[1] Blanco, J., Gonzalez, J., Fernandez, J., Extending Obstacle Avoidance Methods through Multiple Parameter-Space Transformations, Autonomous Robots, vol. 24 (1), pp. 29-48, 2008.

[2] Alvarez, J., Lopez, F., Cuadra, J., Santos, D., Reactive navigation in real environments using partial center of area method, Robotics and Autnomous Systems, Vol. 58, n.12, 2010.

[3] Wang, Y., Chen, D., Autonomous Navigation based on a novel Topological Map, Asia-Pacific Conference on Information Processing, 2009.

[4] Fiorini, P., Shiller, Z., Motion Planning in Dynamic Environments using Velocity Obstacles, in The International Journal of Robotics Research, vol. 17, no. 7, pp. 760, 1998.

[5] Franco, O.; Perez-Sala, X.; Angulo, C. Identificación y seguimiento de personas usando kinect por parte de un robot seguidor. A: Jornadas de ARCA. "XIII Jornadas de ARCA: eficiencia energética y sostenibilidad en inteligencia ambiental : sistemas cualitativos y sus aplicaciones en diagnosis, robótica e inteligencia ambiental". Islantilla - Huelva: Universidad de Sevilla, 2011, p. 51-55. 978-84-615-5513-0. [on line]. Disponible desde <http://hdl.handle.net/2117/14270> [Acceso 10 de Junio 2012].

[6] Schwarz L., Mkhitaryan A., Mateus D., Navab N. 2012. Human skeleton tracking from depth data using geodesic distances and optical flow. Image and Vision Computing. Vol 30 pp 217-226. [on line]. Disponible desde <http://www.sciencedirect.com/science/article/pii/S026288561100134X> [Acceso 10 de Junio 2012].

[7] Ruiz J., Galindo C., Gonzales J., Blanco J. 2011. Navegacion reactiva de un robot móvil usando kinect. Robot 2011, Robotica experimental. Escuela superior de ingenieros de la Universidad de Sevilla (España). [on line]. Disponible desde <http://mapir.isa.uma.es/~jlblanco/papers/robot2011ruiz-sarmiento.pdf> [Acceso 10 de Junio 2012].

[8] Dutta T. 2012. Evaluation of the Kinect™ sensor for 3-D kinematic measurement in the workplace. Applied Ergonomics. Vol 43 pp 645-649. [on line]. Disponible desde <http://www.sciencedirect.com/science/article/pii/S0003687011001529> [Acceso 10 de Junio 2012].

[9] Hernandez J., Quintanilla A., Lopez J., Rangel F., Ibarra M., Almanza D. 2012. Detecting objects using color and depth segmentation with Kinect sensor. Procedia Technology. Vol 3 pp 196-204. [on line]. Disponible desde <http://www.sciencedirect.com/science/article/pii/S2212017312002502> [Acceso 10 de Junio 2012].

[10] Ruiz, J., Galindo, C., Gonzales, J., Blanco, J., Navegación reactiva de un robot móvil usando Kinect, Robot 2011, Semana Europea de Robótica, 2011.

[11] (2010) Open Kinect [on line]. Disponible desde <http://openkinect.org> [Acceso 10 de mayo 2011].

[12] Bradsky, G., Kaeblar, A., Learning OpenCV computer vision with the OpenCV library, O'Reilly Books, United States of America, 2008.

[13] Hagen, W., Ubuntu Linux Bible: Featuring Ubuntu 10.04 LTS, Wiley Publishing Inc, United States of América, 2010.

[14] Gómez, D., Reconocimiento de formas y visión artificial, Ra-ma, Madrid, 1993, p. 235.

[15] Grifith, A., The complete reference GCC, McGraw-Hill, 2002.

[16] Hagen, W., The definitive guide to GCC, Apress, United States of América, 2006.

[17] Sobrino, J., Teledetección, Guada Impresores S.L., 2000, p. 249.

[18] Riaño, O., Algebra lineal en el procesamiento digital de imagines, Fondo de publicaciones Universidad Distrital Francisco José de Caldas, Colombia, 2010.

[19] Pajares, G., De La Cruz, J., Visión por computador imagines digitales y aplicaciones, Alfaomega Ra-Ma, Madrid, 2002.

[20] Digilent [on line]. Disponible desde <http://digilentinc.com> [Acceso 2 de Julio 2011].

[21] (2007) GNU General Public License [on line]. Disponible desde <http://gnu.org/licenses/gpl.html> [Acceso 12 de Agosto 2011].

[22] Arduino Language Reference [on line]. Disponible desde <http://arduino.cc/en/Reference/HomePage> [Acceso 5 de Julio 2011].

[23] (2011) Visión Artificial Unillanos [on line]. Disponible desde <http://visionartificialunillanos.wikispaces.com/> [Acceso 20 de Septiembre 2011].

[24] Kisacanin, B., Bhattacharyya, S., Chai, S., Embedded Computer Vision, Springer, 2009.