ISSN 1870-4069

# Reconocimiento de objetos cuasi-planos mediante un sistema de tratamiento digital de imágenes embebido en una plataforma tipo Raspberry Pi

Julián Jerónimo[1], Humberto Sossa[2,3]

[1] Instituto Tecnológico Superior de Coatzacoalcos, Coatzacoalcos, Veracruz, México

[2] Instituto Politécnico Nacional, CIC, Ciudad de México, México

[3] Tecnológico de Monterrey, Zapopan, Jal., México

julianjb@protonmail.com, humbertososa@gamil.com

**Resumen.** En este trabajo se describe un sistema embebido para el reconocimiento de objetos cuasi-planos. El sistema combina técnicas estándar de tratamiento digital de imágenes, lo que permite describir cada objeto a través de un vector de atributos numéricos. Un objeto detectado a través de la cámara del sistema es reconocido mediante un clasificador típico de distancia euclidiana. El sistema completo es embebido en una plataforma tipo Raspberry Pi con una cámara conectada a él. El sistema es probado en varios escenarios.

**Palabras clave:** Reconocimiento de objetos, sistemas embebidos, momentos invariantes.

# Recognition of Quasi-plane Objects Using Digital Image Processing System Embedded in a Platform of Raspberry Pi Type

**Abstract.** In this work, an embedded system for the recognition of quasi-plane objects is described. The system combines standard techniques of digital image processing, which allows describing each object through a vector of numerical attributes. An object detected through the system's camera is recognized by a typical Euclidean distance classifier. The complete system is embedded in a platform of Raspberry Pi type with a camera connected to it. The system is tested in several scenarios.

**Keywords:** Recognition of objects, embedded systems, invariant moments.

## 1. Introducción

El desarrollo de aplicaciones de reconocimiento de objetos mediante tratamiento digital de imágenes embebidas en sistemas de bajo consumo, plantea retos importantes a la comunidad de visión artificial. Las características limitadas en el hardware con respecto a la capacidad de procesamiento, la cantidad de memoria y el número reducido de puertos en estos sistemas complican el uso de técnicas de reconocimiento de objetos en tiempo real debido a su alto costo de procesamiento y al hardware especializado que estas requieren. Por lo tanto, la implementación de sistemas de reconocimiento de objetos en tiempo real en plataformas embebidas requiere de un importante nivel de eficiencia y un adecuado aprovechamiento de las capacidades de los componentes del hardware.

A pesar de las evidentes carencias que este tipo de sistemas presentan, en comparación con los equipos de cómputo convencionales, cuentan con una ventaja importante que ha llevado a aumentar su popularidad en las comunidades de desarrolladores: son sistemas relativamente económicos y fáciles de adquirir. De esta manera, un sistema de reconocimiento de objetos embebido en una plataforma Raspberry Pi 3 modelo B conectado a una cámara para procesamiento en tiempo real puede ser desarrollado con una inversión cercana a los 100 dólares, la cual está muy por debajo del costo promedio de un equipo de cómputo convencional.

Este tipo de sistemas de reconocimiento de objetos embebidos en plataformas tipo Raspberry Pi ofrecen una solución al problema de la falta de equipo en instituciones educativas de nivel medio superior, superior y posgrado, las cuales requieren de equipos de cómputo para el aprendizaje de conceptos básicos de técnicas procesamiento de imágenes y visión artificial por parte de los alumnos, sin la necesidad de realizar una inversión elevada que pueda afectar el presupuesto de estas instituciones. Asimismo, otra ventaja de este tipo de sistemas es su facilidad de uso y su tamaño reducido, lo cual representa una opción ideal para su uso en actividades educativas, ya que son menos propensos a presentar alguna falla en sus componentes durante su manipulación.

Un ejemplo de una implementación de reconocimiento de objetos se describe en [1]. En este trabajo, los autores utilizan marcas de referencia y segmentación de color, así como la detección de contornos. Una de las limitantes que presenta dicha implementación es que no puede procesar más de un objeto a la vez, además, dicho objeto debe aparecer en una posición determinada de acuerdo a las marcas de referencia utilizadas, de lo contrario el reconocimiento no funciona.

En este trabajo se describe un sistema de reconocimiento embebido en una plataforma tipo Raspberry Pi mediante una cámara conectada, el cual es compatible con múltiples clases de objetos cuasi-planos procesados simultáneamente en tiempo real. Para el desarrollo de este sistema se utilizó un conjunto de técnicas de procesamiento de imágenes, comenzando por la medición del valor promedio del color en el cuadro procesado como se propone en [2] que permita la elección adecuada del método de umbralado como se muestra en [3] y [4]. El resultado de este método adaptativo permite generar regiones de interés utilizando un análisis de componentes conectados como se ilustra en [5]. De las regiones de interés se realiza el cálculo de los momentos centrales normalizados como se menciona en [6]. A partir de estos

momentos se obtienen los momentos invariantes de Hu como se hace en [7]. Para determinar la clase de pertenencia de un objeto, se utiliza un clasificador de distancia euclidiana, el cual, ofrece una solución efectiva para el problema en cuestión como se describe en [8].

El resto del artículo está organizado de la siguiente forma: en la sección 2 se presenta el desarrollo del sistema propuesto. Los resultados obtenidos a partir del funcionamiento del sistema propuesto y la medición de su desempeño se muestran en la sección 3. Finalmente, en la sección 4 se mencionan las conclusiones alcanzadas y las propuestas de trabajo a futuro.

## 2. Desarrollo

El desarrollo del sistema se realizó a lo largo de tres etapas: (1) definición de una metodología para la detección, clasificación y reconocimiento de objetos; (2) instalación y configuración de la plataforma tipo Raspberry Pi; y (3) ejecución en tiempo real de la implementación. Cada etapa que compone al proceso de desarrollo del sistema se describe a continuación:

### 2.1. Metodología de detección, clasificación y reconocimiento de objetos

Sin pérdida de generalidad, para probar la eficiencia y eficacia del sistema se seleccionaron cinco clases de objetos (ver Fig. 1). No obstante, es posible definir nuevas clases de objetos realizando un entrenamiento el cual consiste en la evaluación del promedio de la sumatoria del primer momento invariante de Hu para un conjunto determinado de imágenes en donde se encuentre el objeto que se pretende definir. Se ha escogido el primer momento de Hu por ser el más representativo en aplicaciones de reconocimiento de formas [1,6].

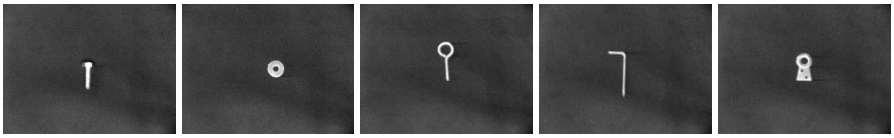

**Fig. 1.** Las cinco clases de objetos propuestos, de izquierda a derecha: tornillo, rondana, alcayata, armella y cola de milano.

La ejecución del sistema propuesto se realizó a partir de la captura de video en tiempo real por medio de una cámara de ocho megapíxeles conectada a la plataforma Raspberry Pi. El número de fotogramas por segundo procesados por la cámara es de 32, lo que equivale a 32 cuadros o imágenes procesadas por el sistema por cada segundo de video. Para disminuir el costo de procesamiento en la plataforma de la Raspberry Pi se estableció manualmente una resolución de imagen de $320 \times 240$ píxeles y un tiempo de retraso para preparar la cámara de 1 milisegundo de duración. Cada cuadro procesado por la cámara fue convertido a un arreglo multidimensional para acceder a la información de los píxeles que lo componen y de esta manera poder realizar el

reconocimiento de objetos. Las etapas que componen la metodología del sistema pueden observarse en el diagrama de la Fig. 2. Cada una de estas etapas se explican enseguida.

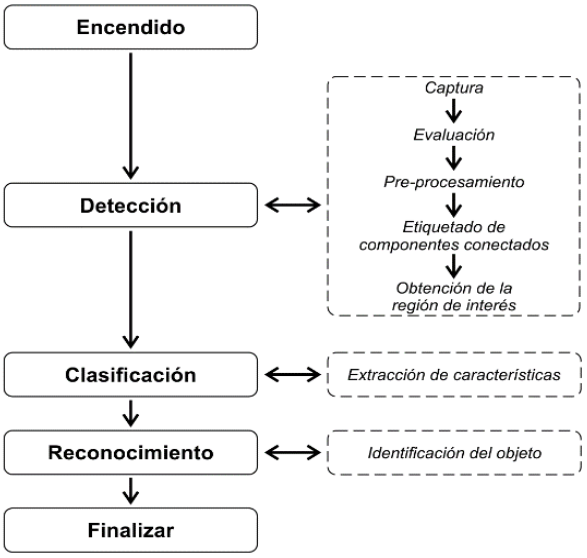


**Fig. 2.** Etapas de la metodología del sistema propuesto.

**Encendido.** Al encender la plataforma de la Raspberry Pi las siguientes tareas son realizadas de manera automática: (1) se inicia el sistema operativo, y (2) se cargan las dependencias utilizadas por el sistema. El usuario ejecuta manualmente el sistema a través de un archivo de procesamiento de lotes (comúnmente llamado archivo `script`), el cual se encuentra embebido en la plataforma de la Raspberry Pi. El sistema verifica que la cámara se encuentre presente y no esté siendo ocupada por ningún otro proceso, tras esto el sistema comienza a ejecutarse.

**Detección.** De acuerdo a la Fig. 2, esta etapa consta de los siguientes cinco pasos.

**Captura:** Al iniciar el programa, se obtiene el cuadro actual capturado por la cámara de la Raspberry Pi y se prepara para ser procesado por el sistema.

**Evaluación:** Se identifican los componentes candidatos que pertenecen a alguna clase de objeto. Para esto, el sistema calcula el valor promedio del color del cuadro procesado por la cámara de acuerdo a los valores presentes en los canales BGR (Blue, Green, Red) que utiliza por defecto la biblioteca de OpenCV. El valor promedio del color de la imagen viene dado por la sumatoria del valor promedio del color de cada fila de píxeles en la imagen. Debido a que no es necesario conocer el valor promedio de cada canal para determinar el ruido presente en la imagen, el valor obtenido de la sumatoria de las filas de píxeles se redondea para seleccionar el método de umbralado más conveniente para el tipo de imagen según el nivel de ruido presente.

**Pre-procesamiento:** De acuerdo al nivel de ruido presente en la imagen el sistema elige la técnica de umbralado adecuada para realizar la detección de componentes. En el caso de una imagen con un valor promedio superior a 90 se aplica un umbralado adaptativo, caso contrario en una imagen con un valor promedio igual o inferior a 90 en donde se aplica un umbralado manual. Ambos escenarios se muestran en la Fig. 3.

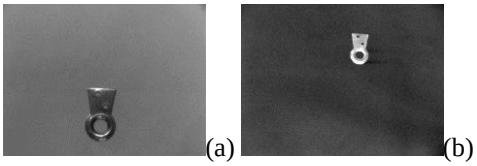


**Fig. 3.** (a) Imagen con valor promedio superior a 90; (b) Imagen con valor promedio igual o inferior a 90.

**Etiquetado de componentes conectados:** A continuación se realiza el etiquetado de los componentes conectados sobre la imagen umbralada. Mediante esta técnica, como se sabe, el sistema asigna de forma automática etiquetas a cada componente encontrado en el cuadro procesado por la cámara de la plataforma de la Raspberry Pi. El proceso del etiquetado de componentes conectados parte de la idea de analizar cada píxel dentro de una determinada fila en la imagen, repitiendo el proceso por cada una de las filas de píxeles que componen a la imagen, asignando etiquetas sucesivas de acuerdo al número de componentes conectados encontrados.

**Obtención de la región de interés:** Para cada componente conectado se define una región de interés a través de una caja delimitadora cuyas dimensiones toman como base a la altura y la anchura del componente, así como sus coordenadas en el eje X y en el eje Y. Cada región de interés generada representa un objeto potencial perteneciente a alguna de las clases descritas durante el entrenamiento. En la Fig. 4 se puede ver cómo se ha logrado realizar correctamente la separación de los cuatro objetos en la imagen.

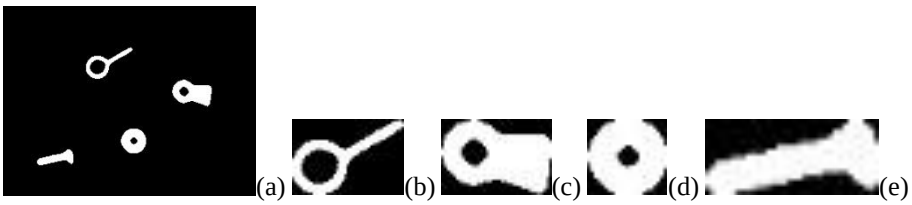


**Fig. 4.** (a) Imagen original con los componentes conectados; (b), (c), (d) y (e) regiones de interés generadas a partir del etiquetado de componentes conectados.

**Clasificación.** De acuerdo a la Fig. 2, esta etapa consta del siguiente paso.

**Extracción de características:** Las características de una clase de objeto son definidas a partir del cálculo de los momentos centrales normalizados de cada región de interés

obtenida en la etapa previa. Los momentos centrales normalizados, como es bien sabido, pueden ser calculados de la siguiente forma:

$$n_{pq} = \frac{\mu_{pq}}{\mu_{00}^{\gamma}}, \gamma = \frac{(p+q+2)}{2}, p+q = 2,3,\ldots \quad (1)$$

En este caso $\mu_{pq}$ representan los momentos del centroide, $\mu_{00}$ representa al momento central de orden cero y $p, q$ son valores enteros no negativos. A partir de estos momentos centrales normalizados se realiza el cálculo del primer momento invariante de Hu el cual es utilizado para definir la clase a la que pertenece cada componente. El valor del primer momento invariante de Hu es obtenido de la siguiente manera:

$$\emptyset_1 = n_{20} + n_{02}. \quad (2)$$

En este caso $n_{20}$ y $n_{02}$ son momentos centrales normalizados de orden dos. El uso de los momentos invariantes de Hu en la etapa de clasificación es especialmente útil gracias a sus propiedades de invariancia ante traslaciones, rotaciones y cambios de escala [6]. Esto es comprobable a través de una matriz como la mostrada en la Tabla 1, donde se comparan los valores del primer momento invariante de Hu para 10 imágenes de cada objeto. Como puede observarse, los datos mostrados en la tabla reflejan poca dispersión entre los valores de cada clase, lo cual es un indicador, como es bien sabido, de que el proceso de clasificación se podrá realizar de manera robusta sin importar variaciones en los cambios de posición, las rotaciones y los cambios de escala de los objetos.

**Reconocimiento:** De acuerdo a la Fig. 2, esta etapa consta del siguiente paso:

**Tabla 1.** Los datos mostrados en la matriz de momentos reflejan poca dispersión para los 10 escenarios propuestos para cada clase de objeto.

| $\emptyset_1$ | Tornillo | Rondana | Armella | Alcayata | Cola de milano |
|---|---|---|---|---|---|
| Valor 1 | 2.8003 | 3.1431 | 2.6098 | 2.2586 | 3.0346 |
| Valor 2 | 2.7835 | 3.1466 | 2.5967 | 2.2829 | 3.0320 |
| Valor 3 | 2.7882 | 3.1517 | 2.5889 | 2.2923 | 3.0288 |
| Valor 4 | 2.8043 | 3.1573 | 2.5833 | 2.2795 | 3.0281 |
| Valor 5 | 2.7770 | 3.1464 | 2.6161 | 2.2829 | 3.0310 |
| Valor 6 | 2.8149 | 3.1512 | 2.6073 | 2.2814 | 3.0412 |
| Valor 7 | 2.7949 | 3.1525 | 2.6093 | 2.3052 | 3.0496 |
| Valor 8 | 2.7802 | 3.1529 | 2.6029 | 2.2885 | 3.0434 |
| Valor 9 | 2.7809 | 3.1493 | 2.6150 | 2.3056 | 3.0449 |
| Valor 10 | 2.7872 | 3.1466 | 2.6079 | 2.3046 | 3.0374 |

**Identificación del objeto:** Con las características obtenidas se elige la clase de objeto a la que pertenece el componente mediante el uso de un clasificador de distancia euclidiana, el cual se encarga de calcular la distancia, en este caso, del valor del primer momento invariante de Hu del componente en relación con los valores representantes de las clases de objetos. Solo para recordar, esta distancia se define como:

$$distancia = \sqrt{(\emptyset_1 - representante)^2}. \quad (3)$$

Del conjunto de los valores de las distancias obtenidas se determina la clase de pertenencia para el componente mediante la selección de la distancia mínima, dada por la siguiente operación:

$$clase = argmin(distancia). \quad (4)$$

**Finalizar.** Si el sistema, por medio de la cámara de la plataforma de la Raspberry Pi, encuentra al menos un objeto que pertenezca a algunas de las clases definidas, la biblioteca de texto a voz `pyttsx` es instanciada pronunciando a través de un altavoz conectado a la Raspberry Pi el nombre de la etiqueta de la clase del objeto. Se usa un rango de velocidad promedio de 130 palabras por minuto por cada cuadro procesado en el que el objeto este presente.

### 2.2. Instalación y configuración de la plataforma Raspberry Pi

Existen varios modelos de plataformas tipo Raspberry Pi en el mercado, la diferencia entre estos modelos radica en las diversas configuraciones de hardware que presentan, siendo algunos modelos más avanzados gracias a sus especificaciones técnicas superiores permitiendo obtener una mayor eficiencia en su capacidad de procesamiento y administración de memoria. Una comparación detallada de los componentes técnicos de los distintos modelos se puede consultar en [9].

El modelo de la plataforma elegido para el desarrollo e implementación del sistema fue la Raspberry Pi 3 modelo B, sin embargo el sistema puede ser utilizado en otros modelos de la plataforma así como en otras plataformas embebidas que sean compatibles con las dependencias del sistema.

En lo que se refiere a la cámara de la plataforma, se utilizó la versión 2 del módulo de la cámara de la plataforma de la Raspberry Pi, la cual tiene una resolución de imagen de ocho megapíxeles y se conecta mediante el conector de la interfaz en serie de la cámara (CSI, por sus siglas en inglés) de la plataforma de la Raspberry Pi.

La implementación del sistema embebido en la plataforma tipo Raspberry Pi requirió de la integración de los componentes de hardware, los cuales son el módulo de la cámara y la tarjeta microSD con el sistema operativo instalado. El uso de un altavoz es opcional ya que su función dentro del sistema es con fines demostrativos únicamente.

En cuanto a la instalación y configuración del software en la plataforma, el sistema requiere de un sistema operativo para funcionar y ejecutar el archivo `script`. Para realizar la implementación descrita en este artículo, el sistema operativo utilizado fue `Raspbian`. No obstante, debido a que el sistema de reconocimiento de objetos fue diseñado para su uso en sistemas embebidos, este es multiplataforma, lo que significa que puede ser ejecutado en diferentes sistemas operativos sin necesidad de cambiar el código fuente. En lo que se refiere a las dependencias, el sistema requirió de la instalación de la biblioteca de visión por computadora OpenCV, la instalación de la interfaz `picamera` para la manipulación del módulo de la cámara de la Raspberry Pi y la instalación de la biblioteca de texto a voz `pyttsx`.

### 2.3. Ejecución en tiempo real de la implementación

Los resultados obtenidos son mostrados, de manera gráfica, en una ventana dividida en tres secciones como aparece en la Fig. 5: la sección "Original" muestra el cuadro actual procesado por la cámara, la sección "Umbralado" muestra los resultados de la etapa de binarizado del sistema para dicho cuadro y la sección "Resultado" permite visualizar los resultados de las etapas de clasificación y reconocimiento del sistema para el cuadro en cuestión.


**Fig. 5.** Resultados de la ejecución del sistema durante el procesamiento de video en tiempo real capturado por la cámara de la plataforma tipo Raspberry Pi.

## 3. Resultados experimentales

En esta sección se presentan los resultados de las pruebas de desempeño del sistema propuesto en distintos escenarios. A través de la matriz de confusión (Tabla 2) se muestran los resultados para una configuración de la cámara paralela al plano trabajo de los objetos a 10 centímetros de distancia entre la cámara y el espacio de trabajo. En este caso, como se puede ver los resultados siempre fueron los deseados y esto es respaldado por las propiedades de invariancia de los momentos de Hu.

**Tabla 2.** Matriz de confusión con los resultados del sistema para el escenario con cámara en posición paralela a los objetos a 10 centímetros de distancia.

| Matriz de confusión Escenario 1 | | Valor predicho | | | | |
|---|---|---|---|---|---|---|
| | | Tornillo | Rondana | Armella | Alcayata | Cola de milano |
| Valor real | Tornillo | 10 | 0 | 0 | 0 | 0 |
| | Rondana | 0 | 10 | 0 | 0 | 0 |
| | Armella | 0 | 0 | 10 | 0 | 0 |
| | Alcayata | 0 | 0 | 0 | 10 | 0 |
| | Cola de milano | 0 | 0 | 0 | 0 | 10 |

Por otro lado, en la matriz de confusión de la Tabla 3 se muestran los resultados para una configuración en la posición de la cámara con un ángulo de inclinación de 30° con respecto al plano de trabajo y a 15 centímetros de distancia. Nótese como en este caso ya se tienen algunos errores en la clasificación de algunos objetos.

**Tabla 3.** La matriz de confusión con los resultados del sistema para el escenario con la posición de la cámara a un ángulo de inclinación de 30° a 15 centímetros de distancia.

| Matriz de confusión Escenario 2 | | Valor predicho | | | | |
|---|---|---|---|---|---|---|
| | | Tornillo | Rondana | Armella | Alcayata | Cola de milano |
| Valor real | Tornillo | 9 | 0 | 1 | 0 | 0 |
| | Rondana | 0 | 10 | 0 | 0 | 0 |
| | Armella | 0 | 0 | 8 | 0 | 2 |
| | Alcayata | 0 | 0 | 0 | 10 | 0 |
| | Cola de milano | 0 | 1 | 0 | 0 | 9 |

**Tabla 4.** La matriz de confusión con los resultados del sistema para el escenario con posición de la cámara a un ángulo de inclinación de 45° y a 10 centímetros de distancia.

| Matriz de confusión Escenario 3 | | Valor predicho | | | | |
|---|---|---|---|---|---|---|
| | | Tornillo | Rondana | Armella | Alcayata | Cola de milano |
| Valor real | Tornillo | 8 | 0 | 2 | 0 | 0 |
| | Rondana | 0 | 9 | 0 | 0 | 1 |
| | Armella | 0 | 0 | 7 | 3 | 0 |
| | Alcayata | 0 | 0 | 1 | 9 | 0 |
| | Cola de milano | 0 | 2 | 0 | 0 | 8 |

En un tercer experimento, la cámara se situó a 10 centímetros del plano de trabajo a 45° de inclinación entre la cámara y dicho plano. La matriz de confusión de la Tabla 4 muestra los resultados obtenidos. Nótese como en esta situación el sistema tiende a presentar más errores. Es de suponer que a mayores inclinaciones y más alturas el sistema ofrecerá mayores errores.

El desempeño del clasificador es calculado con base a la siguiente fórmula:

$$d = 1 - \frac{errores}{150} x100 = n\%. \quad (5)$$

Por lo tanto, considerando los 13 errores obtenidos, el desempeño $d$ del sistema propuesto para los 150 escenarios adoptados es de un 91.3%. En la Tabla 5 se muestran los tiempos y la cantidad de memoria requeridos por la plataforma tipo Raspberry Pi para procesar los objetos contenidos en un cuadro capturado por la cámara.

**Tabla 5.** Tiempo de procesamiento y cantidad de memoria requeridos según el número de objetos.

| Número de objetos procesados (por cuadro) | Tiempo de procesamiento (en milisegundos) | Cantidad de memoria (en mebibytes) |
|---|---|---|
| 0 objetos | 38.2 ms | 17.73 MiB |
| 1 objeto | 54.0 ms | 17.92 MiB |
| 2 objetos | 54.9 ms | 18.11 MiB |
| 3 objetos | 55.4 ms | 18.30 MiB |
| 5 objetos | 58.1 ms | 18.68 MiB |
| 10 objetos | 66.5 ms | 19.63 MiB |
| 15 objetos | 75.4 ms | 20.58 MiB |

De esta tabla se puede observar que el número de objetos en la imagen efectivamente influye en los tiempos invertidos para el procesamiento completo de la imagen, de la misma manera ocurre en la cantidad de memoria utilizada en donde se puede observar una variación incremental de 0.19 MiB por cada objeto presente en la imagen.

## 4. Conclusiones y trabajo a futuro

En este artículo se describió un sistema eficiente para el reconocimiento de objetos cuasi-planos compatible con sistemas embebidos. Esta propuesta es aplicable a cualquier plataforma embebida cuyo sistema operativo sea compatible con las dependencias utilizadas por el sistema; solo los resultados en la plataforma de la Raspberry Pi 3 modelo B fueron presentados en este artículo.

Para medir el desempeño del sistema se propusieron cinco clases de objetos. No obstante el método permite definir hasta *n* objetos siempre y cuando los representantes de las clases sean lo suficientemente diferentes entre sí. Si la diferenciación entre los objetos adquiere un carácter no lineal, se deberá adoptar otro tipo de clasificador, por ejemplo, un arreglo de perceptrones en capas.

Una de las ventajas principales del sistema propuesto es su bajo consumo de procesamiento y memoria, lo que permite que varios cuadros capturados por la cámara conectada a la plataforma de la Raspberry Pi sean procesados simultáneamente, consiguiendo clasificar los objetos de manera correcta sin afectar el rendimiento del sistema.

Otra ventaja importante es la capacidad de reconocimiento en tiempo real de múltiples clases de objetos ubicados de forma dispersa de manera simultánea por cada cuadro procesado por la cámara, lo cual significa que el sistema puede ser implementando en líneas de producción industrial para el reconocimiento de múltiples objetos optimizando la toma de decisiones en los sistemas de producción en cadena.

De los experimentos realizados con cinco tipos de clases de objetos en una plataforma tipo Raspberry Pi se puede concluir que el desempeño del sistema es lo suficientemente bueno como para ser utilizado en dispositivos embebidos de características limitadas sin sacrificar la robustez del proceso de reconocimiento, obteniendo resultados similares a los que se obtendrían en un equipo de cómputo convencional.

Los trabajos a futuro incluyen: 1) medir el desempeño del sistema propuesto en problemas con más de cinco clases de objetos, 2) cambiar de clasificador, por ejemplo, una red neuronal artificial o una máquina vector soporte para poder afrontar problemas de carácter no lineal, e 3) implementar el sistema en robots móviles que puedan realizar determinadas acciones a partir de las clases de objetos reconocidos.

**Reconocimientos.** Julián Jerónimo agradece a la Academia Mexicana de Ciencias y a su programa "Verano de la Investigación Científica" por otorgarle la beca para realizar su estancia de investigación en el Centro de Investigación en Computación del IPN. H. Sossa agradece al Instituto Politécnico Nacional y al CONACYT por los apoyos

económicos, en el marco de los proyectos SIP 20170693, SIP 20180730 y 50 (Fronteras de la Ciencia).

## Referencias

1. Arévalo, E., Zúñiga, A., Villegas, J., Avilés, C. Implementación de reconocimiento de objetos por color y forma en un robot móvil. Research in Computing Science, 91, pp. 21–31 (2015)
2. Ali, M., Dong, L., Liang, Y., Xu, Z., He, L. Feng, N.: A color image retrieval system based on weighted average. International Conference on Signal Processing, Communication and Computing, pp. 184–189 (2014)
3. Rosebrock, A.: Practical Python and OpenCV. 1ra ed. PyImageSearch, USA (2014)
4. Firdousi, R., Parveen S.: Local Thresholding Techniques in Image Binarization. International Journal of Engineering and Computer Science Vol: 3, pp. 4062–4065 (2014)
5. Burger, W., Burge, M.: Principles of Digital Image Processing, Undergraduate Topics in Computer Science. Springer-Verlag, London (2009)
6. Flusser, J.: Moment Invariants in Image Analysis, International Journal of Computer. Electrical, Automation, Control and Information Engineering Vol: 1, No: 11, pp. 196–201 (2006)
7. Li, D.: Analysis of Moment Invariants on Image Scaling and Rotation. In: Sobh T., Elleithy K. (eds.) Innovations in Computing Sciences and Software Engineering, pp. 415–419, Springer, Dordrecht (2010)
8. Jing, L., Bao-Liang, L.: An adaptive image Euclidean distance. Pattern Recognition 42, pp. 349–357 (2009)
9. Pajankar, A., Kakkar, A.: Raspberry Pi by Example. 1ra ed. Packt Publishing, UK (2016)