Enfoque UTE, V.8-N.4, Sep.2017, pp. 27 - 40
http://ingenieria.ute.edu.ec/enfoqueute/
e-ISSN: 1390-6542 / p-ISSN: 1390-9363

Recibido (Received): 2017/07/20
Aceptado (Accepted): 2017/09/29

Hacia la construcción de un dispositivo de asistencia para personas no videntes en el juego de cuarenta

(Towards the construction of a device to support blind people in the "cuarenta" game)

Holger Ortega[1], Rodrigo Tufiño[1], Juan Estévez[1]

Resumen:
El presente trabajo tiene como objetivo el desarrollo de un sistema de reconocimiento automático de una carta de baraja ubicada sobre una mesa, como parte del proyecto más general de crear un dispositivo para asistir a personas no videntes en el juego de azar denominado "cuarenta". El objeto de dicho dispositivo será informar al usuario de las cartas en juego, vía audio. Para esta fase del proyecto se utilizó el algoritmo de k-vecinos más cercanos entrenado con un conjunto de caracteres alfanuméricos sintéticos. El conjunto de prueba constó de fotografías tomadas en condiciones controladas de iluminación, con las cartas ubicadas en orientación arbitraria. La parametrización del algoritmo arrojó un valor de 1 como k óptimo, con el cual se obtuvo un error de clasificación en el conjunto de prueba de 5%. Solo dos caracteres fueron confundidos por el clasificador, la "A" y la "J", con errores de 20% y 40% cada uno. El algoritmo fue implementado en un sistema embebido Raspberry Pi 3, y obtuvo un tiempo de respuesta de 5 segundos, incluida la conversión a audio, y una ocupación de memoria RAM que no superó el 60% de la capacidad del sistema. Estos resultados sugieren su aplicabilidad en dispositivos portátiles.

Palabras clave: reconocimiento automático**;** visión artificial; k vecinos más cercanos; juegos de azar; inclusión.

Abstract:
The present work has the objective of developing a system for the automatic recognition of a playing card on a table, as part of a more general project to create a device to assist the blind in the chance game called "cuarenta". The goal of this device is to inform the user about the cards being played, via audio. For this phase of the project the algorithm used was k-NN, trained with a set of alphanumeric synthetic characters. The test set contained photographs taken in controlled lighting conditions, with the cards positioned in arbitrary orientations. The parameterization of the algorithm gave a value of 1 as the optimal k, with which a classification error of 5% was obtained in the test set. Only two characters were confused by the classifier, the "A" and the "J", with 20% and 40% error each one. The algorithm was implemented in an embedded Raspberry Pi 3 system, obtaining a response time of 5 seconds, including the conversion to audio, and a memory occupation no greater than 60% of the total capacity of the system. These results suggest its applicability in portable devices

Keywords: automatic recognition; artificial vision; k-nearest neighbors; playing cards; inclusion

[1] Universidad Politécnica Salesiana, Quito – Ecuador ({hortega, rtufino} @ups.edu.ec; jestevez@est.ups.edu.ec)

1. Introducción

En el Ecuador existen alrededor de 274846 personas no videntes (López, 2010), las personas de este sector de la población aún deben sobrellevar grandes dificultades para poder tener una vida normal, y es una política estatal del gobierno del Ecuador permitir que este escenario cambie en el país y se logre una inclusión más eficaz. Los aspectos en los que se ha puesto más énfasis al momento de desarrollar tecnologías para personas no videntes han sido, obviamente, aquellos que son prioritarios para la supervivencia: dispositivos que les permiten caminar con cierta seguridad por las calles, cocinar los alimentos para su familia, acceder a la educación y capacitación, etc. Sin embargo, en el mismo Plan Nacional del Buen Vivir se contempla la satisfacción de necesidades básicas como apenas uno de ocho elementos que conforman el *sumak kawsay* (Secretaría del Buen Vivir, 2017), y por lo tanto es necesario atender también otras necesidades de este sector de la población, como son la recreación y la interacción social con otros sectores. En un artículo publicado por el diario El Comercio (Tello, 2014), se habla de las necesidades de recreación de las personas no videntes, y se menciona que practican habitualmente deportes como el fútbol o el ajedrez adaptado.

El presente proyecto pretende aportar a cubrir esa necesidad mediante un dispositivo que permitirá la inclusión de personas no videntes en el juego del cuarenta (Wikipedia, 2016). Al ser este uno de los juegos más tradicionales y representativos de la cultura ecuatoriana, el dispositivo aportará en el bienestar y el sentimiento de pertenencia al país de las personas no videntes. Cabe resaltar que el dispositivo permitirá que la persona no vidente compita no solo con personas de su misma condición sino con jugadores que sí gozan del sentido de la vista.

Para la consecución del objetivo propuesto en la investigación se ha contemplado que la implementación debería ser en un dispositivo portable, fácil de manipular y que sea asequible; uno de los dispositivos comerciales que cumple con estas características es Raspberry PI (Raspberry Pi Foundation, 2016), un minicomputador de placa reducida y bajo costo sobre el cual se puede instalar *software* libre. Las aplicaciones científicas y de investigación son innumerables y la mayoría se han inclinado hacia el área de la mecatrónica con sus derivadas (Hayward, 2014). El presente trabajo consiste en crear una aplicación para dicho dispositivo en el campo de la inteligencia artificial, puntualmente en la visión por computadora, para realizar las cinco fases del procesamiento de imágenes (Szeliski, 211) descritas en la *Figura 1*.

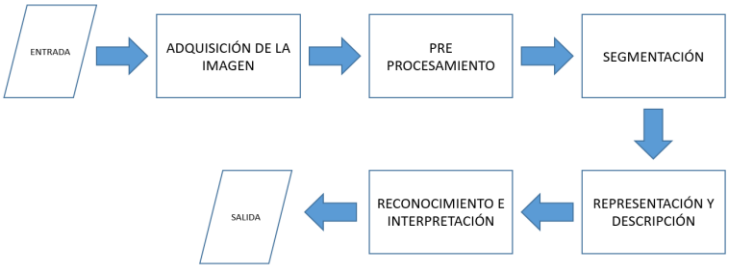


Figura 1. Etapas del procesamiento digital de imágenes.

En el proceso de adquisición de imagen se disciernen dos etapas: la primera se conoce como captura y consiste en utilizar un dispositivo óptico para obtener información relativa a una escena o entorno deseado; la segunda se conoce como digitalización, y consiste en transformar una señal con uno o varios componentes continuos en una imagen digital (Raikwal & Saxena, 2012). La siguiente fase es conocida como segmentación, consiste en aislar las áreas de interés de una escena para interpretarlas. A menudo es necesario refinar esta etapa debido al ruido y al desenfoque que se presenta en la imagen, para lo cual se utilizan diferentes técnicas como: umbralización, agrupación por rasgos comunes, detección de líneas, crecimiento de región y extracción de bordes. La fase de representación y descripción es conocida como extracción de características. En esta fase, la imagen es convertida en un vector que contiene la información más relevante de la misma para el problema. Dado que la representación y descripción son a menudo desconocidas a priori, generalmente se introducen varias características candidatas para intentar representar de una manera adecuada las clases, aun cuando puedan ser redundantes o irrelevantes. Los datos extraídos recogen las características o los rasgos diferenciadores de la imagen analizada.

Cabe mencionar que no siempre las cinco fases siguen un orden secuencial lineal, sino que puede ser necesario volver a una etapa anterior, por ejemplo, a la etapa de segmentación, si la etapa de reconocimiento ha fallado o incluso a la adquisición de la imagen si fuese necesario.

En el reconocimiento e interpretación se utilizan técnicas de reconocimiento geométrico de objetos. El problema fundamental de las técnicas de reconocimiento radica en su comportamiento en sistemas específicos, lo que dificulta la selección del método. Además, en este trabajo se utiliza un sistema embebido de bajas prestaciones, por lo que se deben considerar las capacidades de procesamiento y memoria.

La investigación de trabajos previos (Raikwal & Saxena, 2012), (Sudha & Bhavani, 2012), (Lojo, Losada, & Barreiro, 2009), (Gutiérrez, Lastra, Bacardit, Benítez, & Herrera, 2016) llevó a la elección del clasificador de aprendizaje supervisado k-vecinos más cercanos (kNN) por su procesamiento, rendimiento, precisión, uso de memoria RAM, tiempo de respuesta, tiempo de búsqueda y entrenamiento del algoritmo.

El método de k-vecinos más cercanos es un algoritmo de clasificación. El conjunto de entrenamiento consta del vector de características x_j, más una variable de atributo que es la clase, denominada C_j, para cada uno de los ejemplos. El objetivo de la clasificación es encontrar un modelo para predecir la clase a la que pertenece un vector de características nuevo (Galindo Durán, Juganaru-Mathieu, Áviles Cruz, & Vázquez, 2010). Sea x* dicho vector. Entonces el algoritmo de k-vecinos más cercanos es el siguiente:

1. Calcular las distancias dj desde el vector incógnita x* hasta cada uno de los vectores de entrenamiento xj. Puede emplearse cualquier definición de distancia, aunque la más recurrida es la euclídea.
2. Hacer una lista con los vectores de características correspondientes a las k distancias más cortas.
3. Asignar a x* la clase *Cj* de la mayoría de los vectores de esta lista.

Gráficamente, y mediante la definición euclídea de distancia en un espacio bidimensional, el algoritmo de k-vecinos más cercanos puede visualizarse en la *Figura 2*. El círculo de línea continua representa k=1, el de línea entrecortada representa k=3. En el primer caso, la clase asignada al vector x* (en rojo) sería la clase de los círculos negros; en el segundo, la de las cruces azules.

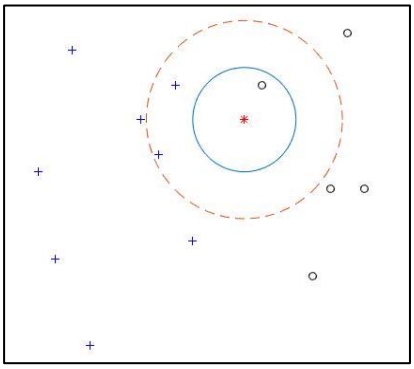
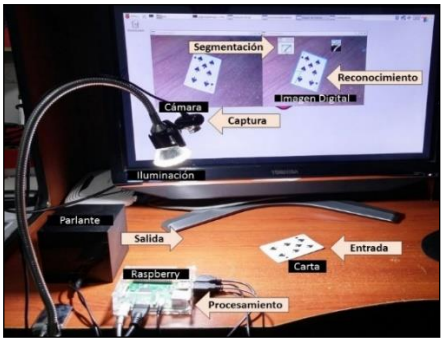


Figura 2. Algoritmo de k-vecinos más cercanos en un espacio bidimensional.

2. Metodología

La primera etapa del desarrollo del sistema corresponde a la investigación exploratoria de las características y capacidades del *hardware* y sistema operativo con respecto a la Raspberry Pi 3. La segunda etapa concierne al desarrollo del sistema de reconocimiento de imágenes y se basa en los requisitos levantados en la primera etapa. En la *Figura 3a* se puede apreciar el proceso de funcionamiento del prototipo desarrollado, mientras que en la *Figura 3b* se aprecia el dispositivo.



a)

b)

Figura 3. a) Sistema automático de reconocimiento de cartas. b) Raspberry Pi 3 modelo B

2.1 Raspberry Pi 3

El sistema embebido Raspberry Pi 3 modelo B es una computadora de placa reducida como lo muestra la *Figura 3b*. Las características principales son su procesador Broadcom BCM2387 *chipset* 1.2 GHz Quad-Core ARM Cortex-A53, su GPU Dual Core VideoCore IV Multimedia Co-Procesador proporciona OpenGL Es 2.0; dispone de una memoria de 1 GB LPDDR2 y la alimentación de energía se realiza a través de un conector Micro USB; su consumo estimado es de 5V, 2.5A. El sistema operativo utilizado es Linux bajo su distribución Raspbian Kernel 4.4 almacenado en una memoria MicroSD de 16 GB y 10 MB/s de velocidad de transferencia. Raspbian Jessie es la distribución *open source* más completa y estable (Monk, 2016). El lenguaje de programación utilizado es Python 2.7 y 3.1 y la librería de visión artificial OpenCV 2.4 y 3.1.

2.2 Dispositivo Óptico

Se utilizó una cámara web USB 2.0 con una resolución de 5 MP 3200 X 2400 a través de un sensor CMOS a color con autoajuste y tasa de transferencia: 480 Mbit/s. Los parámetros a tomar en cuenta durante la adquisición de la imagen son los siguientes: a) Tiempo de integración: mediante la cual las cartas son sometidas a la variabilidad lumínica. b) Tiempo de adquisición: el tiempo que tarda la cámara en transferir la información digital.

2.3 Sistema de iluminación

La iluminación es indispensable en el proceso de la adquisición de la imagen digital, puesto que facilita el análisis y la interpretación de la figura obtenida. Se utilizó un diodo LED de luz blanca de 280 lúmenes y potencia de 4.5 w de 6500K que cumple con las expectativas para controlar los diferentes escenarios a los que puede estar expuesta una imagen.

2.4 Salida de audio

La reproducción del audio como resultado del reconocimiento de la imagen se realiza con la ayuda de una bocina de 2W conectada al jack de audio de 3.5 mm.

2.5 Adquisición de imagen

Una imagen del mundo real, I_C, puede concebirse como una función continua que asigna un valor de intensidad a cada punto del espacio bidimensional, de coordenadas (x, y) (véase la *Ecuación 1*).

$$I_C(x, y) \in R \text{ donde } x, y \in R \tag{1}$$

La imagen digital, discreta, se obtiene tras aplicar un proceso de *muestreo*, que consiste en transformar la matriz I_C en una matriz discreta I_D. Esta última difiere de la primera en que los valores de x e y solo toman valores naturales, pertenecientes a las coordenadas del correspondiente *pixel*

en la pantalla. La matriz I_D correspondiente a una imagen de $N \times M$ pixeles; tiene la estructura mostrada en la *Ecuación 2* (Larcher, Biasoni , & Cattaneo, 2011).

$$I_D(x,y) = \begin{pmatrix} I_D(1,1) & I_D(1,2) & \cdots & I_D(1,M) \\ I_D(2,1) & I_D(2,2) & \dots & I_D(2,M) \\ \vdots & \vdots & \ddots & \vdots \\ I_D(N,1) & I_D(N,2) & \dots & I_D(N,M) \end{pmatrix} \quad (2)$$

Para ser más exactos, una imagen a color se representa mediante *tres* matrices I_D, correspondientes a los canales rojo, verde y azul en el modelo RGB.

La localización de la imagen se especifica por el centro *C(C_x, C_y)* y la orientación dada por el ángulo *Θ*, si se tiene en cuenta que toda carta posee simetría en un ángulo de 180º por su característica de doble etiqueta. La orientación es 0º cuando la carta se encuentra paralela al eje de las x (*Figura 4*).

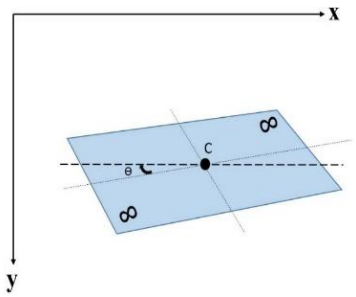


Figura 4. Posición de una carta en la mesa.

2.6 Preprocesamiento

El preprocesamiento de las imágenes en el presente trabajo consistió en la detección de los bordes de la carta. Se entiende por borde al límite que puede tener un objeto o figura y que señala el fin de la superficie en relación con la del medio externo. Una vez determinado su contorno, se puede localizar todo el objeto, así como precisar sus propiedades básicas. La utilización de la información de bordes simplifica en gran medida el análisis de imágenes. Las etapas para la obtención de los bordes se enumeran a continuación y se observan en la *Figura 5*: (a) Imagen original, (b) Escala de grises, (c) Función Gaussiana, (d) Función Canny, (e) Función Morfológica.

La discriminación de otros objetos que no son cuadriláteros se proporciona por la condición de cuatro puntos con respecto a los bordes de la imagen. Se utilizó la implementación del algoritmo de Douglas-Peucker (Bradski & Kaehler, 2008) para dibujar el contorno de la carta. Con las coordenadas obtenidas se trazó el área procesada, mediante las funciones de análisis estructural y descriptores de figuras (Suzuki & Abe, 1985).

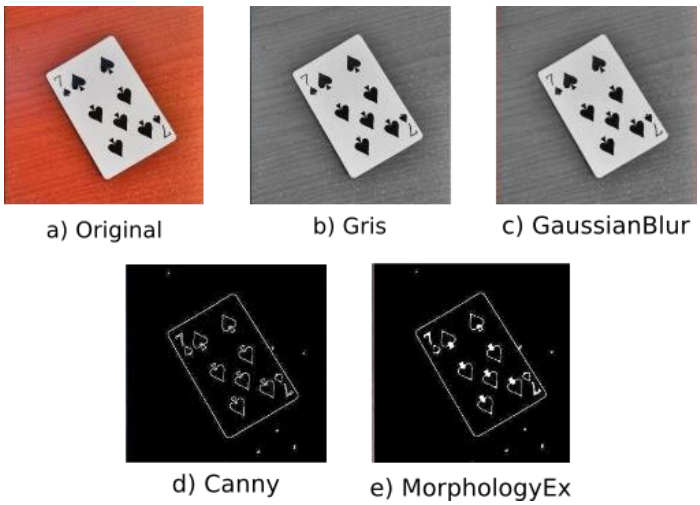


Figura 5. Filtros para la obtención de puntos del cuadrilátero de una imagen.

Para cambiar la perspectiva de las imágenes de las cartas, que aparecen en general como paralelogramos, se aplicó una transformación geométrica (OpenCV Organization, 2014). Dicha transformación convierte los paralelogramos en rectángulos. Posteriormente, se realizó la rotación de la imagen, de manera que la misma aparezca en una orientación horizontal o vertical, a partir de una orientación arbitraria, como se representa en la *Figura 6a.*

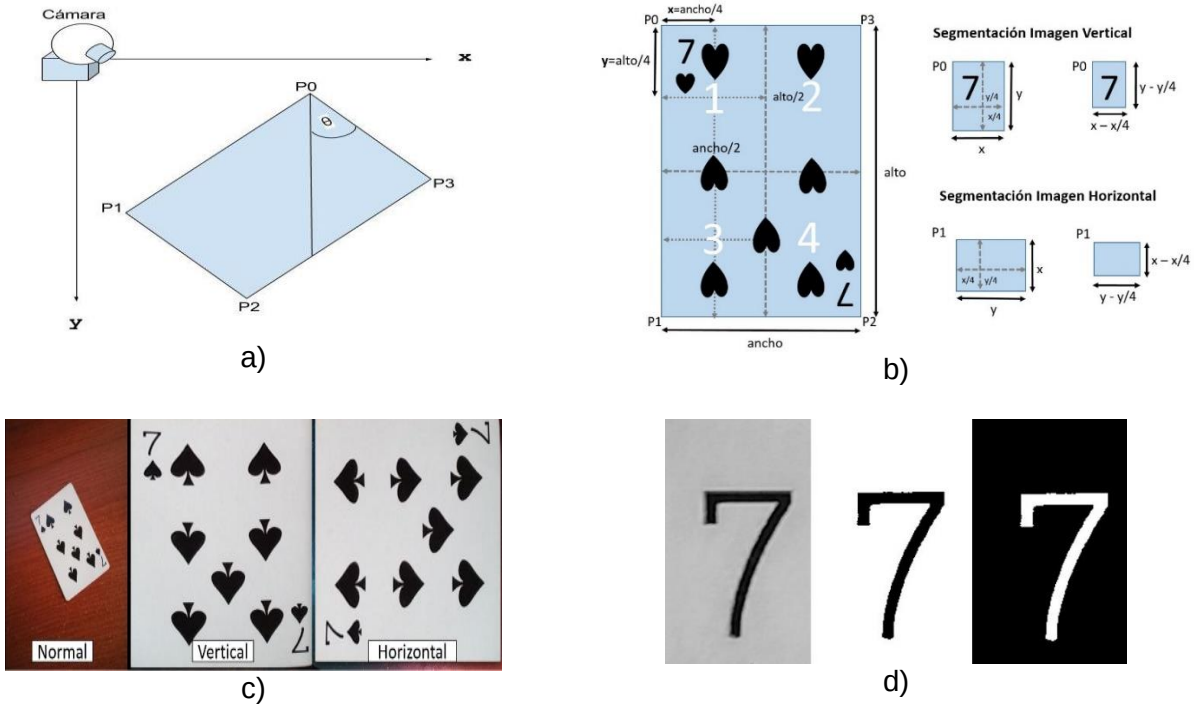


Figura 6. a) Rotación de la carta. b) Segmentación del área de interés. c) Nueva perspectiva y rotación. d) Carácter siete aplicado la operación umbral: imagen normal, umbral binario y umbral binario invertido

Si se toman los puntos en el orden P0, P1, P2, P3, la orientación de la imagen es horizontal; caso contrario, si se recogen los puntos en el orden P0, P3, P2, P1, la orientación es vertical. El algoritmo de obtención de la imagen con la orientación no distingue qué secuencia de puntos es tomada, debido a este conflicto se deben procesar las dos orientaciones posibles. La rotación de una imagen

dada por el ángulo *Θ* se representa con la matriz I_P (*Ecuación 3*) que determina la imagen con la perspectiva trasformada.

Para cambiar la orientación de horizontal a vertical o viceversa, se debe aplicar la *Ecuación 4* con el ángulo de giro de 90º, donde w y h son el ancho y la altura de la imagen, respectivamente. El resultado de aplicar el cambio de orientación se ilustra en la *Figura 6c*.

$$I_p = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \quad (3) \qquad I_t = \begin{bmatrix} \cos\theta & \sin\theta & w \\ -\sin\theta & \cos\theta & h \end{bmatrix} \quad (4)$$

2.7 Segmentación

El resultado de este proceso es obtener la región de la carta que contenga la letra correspondiente. Para esto se divide la imagen en dieciséis partes como muestra la siguiente figura. Los puntos para obtener la segmentación del área de interés son las regiones 1 o 4 mientras que las regiones 2 o 3, tienen un área segmentada vacía que será descartada en la etapa posterior. La *Ecuación 5* permite entender el proceso de binarización inversa de una imagen, donde maxval representa el valor que tomará el pixel *(x, y)* si su valor de intensidad es mayor al umbral. En la *Figura 6d* se ilustra el resultado de este proceso.

$$I_B(x,y) = \begin{cases} maxval & if\ I_s(x,y) > umbral \\ 0 & en\ otro\ caso \end{cases} \quad (5)$$

2.8 Reconocimiento e Interpretación

El algoritmo kNN infiere la clasificación correspondiente a un vector de características *x*, asignándole la clase C_j correspondiente a los *k* vecinos más cercanos (o a la mayoría de ellos). Para esto, calcula previamente la distancia entre este vector y todos los vectores del conjunto de entrenamiento. Si *R* es el número de ejemplos de prueba y *S* el número de ejemplos de entrenamiento, el algoritmo requiere calcular *R* x *S* distancias.

De este modo, cada instancia de prueba es comparada con todas las instancias de entrenamiento, midiéndose la distancia de ambas. Las instancias correspondientes a los *k* valores menores se utilizan para predecir la clase de la instancia de prueba (Gutiérrez, Lastra, Bacardit, Benítez, & Herrera, 2016).

En la fase de entrenamiento se usaron un total de 70 imágenes de números de baraja inglesa, tomadas de una base de datos de fuentes (LeCun, Cortes, & Burges, s.f.). Dichas imágenes se procesaron hasta tener imágenes binarias, que fueron transformadas en vectores de características de dimensión 600. Para calcular las distancias se utilizó la definición euclídea dada por la *Ecuación 6* (Zapata, Pérez, & Mora, 2014).

$$\|x\| = (\sum_{i=1}^{n}(x_i)^2)\text{^}½ \qquad (6)$$

2.9 Parametrización de kNN

Para parametrizar el clasificador kNN se utilizó la aplicación Classification Learner de MATLAB, y se usó un conjunto de entrenamiento de 70 ejemplos (5 por cada clase). Cada ejemplo consta de un vector de características de dimensionalidad 600 y una etiqueta correspondiente a una de las 14 clases (A, K, Q, J, 9, … ,0). Las variantes ensayadas por la aplicación y los correspondientes resultados se muestran en la *Figura 7*. La opción más viable propuesta por la aplicación es k un vecino más cercano 1NN con una precisión de 95.7 %. La velocidad de precisión fue ~ 260 obs/sec. Métrica de distancia euclídea. El tiempo total del entrenamiento fue de 1.444 segundos.

Figura 7. *Classification Learner Knn* de MATLAB

2.10 Clasificación

Se realiza la clasificación mediante matrices de 1xN, donde N es el número de atributos que describen al valor de la carta vertical segmentada. En el proceso de categorización, la función del algoritmo kNN es entregar el valor que representa la clase a la que pertenece la carta de baraja que se encuentra analizando.

2.11. Conversor *Text to Speech*

Utilizando la API de Google TTS (*Text to Speech*) se transforma el resultado de la clasificación en un archivo de audio mp3, que es reproducido por el dispositivo para informar al usuario no vidente. Para esto se utilizan las librerías de Python gtts y pygame (véase la *Figura 8*).

```
1    from gtts import gTTS
2    import pygame
3    def textoAudio(texto):
4        tts = gTTS(text=texto,lang='es-es')
5        tts.save("audio.mp3")
6        pygame.init()
7        pygame.mixer.music.load("audio.mp3")
8        pygame.mixer.music.play()
```

Figura 8. Código en Python para la conversión de texto a audio

3. Resultados

Las pruebas se realizan cumpliendo las etapas de procesamiento de una imagen como se presentó en la *Figura 1* en los diferentes ambientes a los que estuvo expuesto el sistema de reconocimiento de una carta.

El sistema final brindó la posibilidad de trabajar en ambientes variables, mediante un dicroico led de luz blanca para minimizar las variaciones de luminosidad en la imagen de la carta. El comportamiento del sistema al encenderse Raspbian se presenta en los intervalos (0,10] U (63,75] con un rendimiento del CPU de 5.3% y uso de RAM del 26.5%.

El procesamiento del CPU, la memoria RAM y el almacenamiento de la Raspberry Pi 3 bajo la ejecución del sistema automático de reconocimiento de una carta presentan dos etapas, un modo pasivo y un modo activo, como se aprecia en la *Figura 9.*

Modo pasivo

La cámara se encuentra en modo pasivo, cuando el programa se ejecuta y el dispositivo se encuentra a la espera de presionar un botón para capturar una imagen, este momento se presenta en los intervalos (10,20] U (30,40] U (50,60] U (66,75]. El uso del CPU en este instante varía de 7.3% a 13% del total de capacidad de procesamiento. La memoria RAM utilizada para esta operación es del 39%, es decir 362 MB de un total de 925.5 MB, donde solo se ocupan 172.4 MB en el modo pasivo que representa el 18.7% de uso de memoria utilizada por el aplicativo para reconocer las cartas, el resto lo ocupan los procesos del sistema. El almacenamiento del sistema embebido no sufre ninguna variación debido a que todo el proceso de modo pasivo no requiere de lectura o escritura en la microSD.

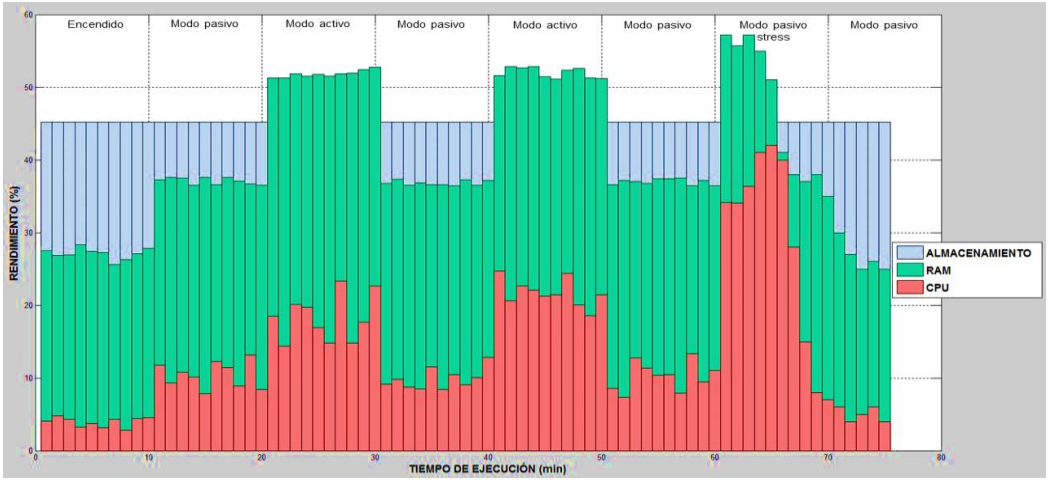


Figura 9. Resultados de las pruebas de rendimiento

Modo activo

El modo activo empieza cuando el sistema embebido realiza las cinco etapas para el procesamiento digital de una imagen en el transcurso de (20,30] U (40,50] donde intervienen los algoritmos expuestos en el apartado 2. El rendimiento del CPU durante el transcurso de predicción varía entre 15% a 25% del total de la capacidad de procesamiento. La memoria RAM utilizada para el modo activo es 52% es decir 481 MB en el cual solo el 28.3% es para uso exclusivo del tratamiento de las imágenes, el resto se divide para la memoria cache 21.2% y el búfer de datos 2.5%. El almacenamiento no se ve afectado debido a que toda imagen se guarda y se sobrescribe, el uso de la microSD es del 45% hasta el momento, que equivale a 7.2 GB de espacio libre de un total de 16 GB el resto de almacenamiento se distribuye entre el sistema operativo y las aplicaciones que vienen por defecto instaladas.

Consecuentemente, en el intervalo de (60,63] se capturaron 360 imágenes en un lapso de 3 minutos para analizar el comportamiento del CPU bajo un período de estrés. Durante este período, el procesamiento del sistema embebido tuvo una respuesta de 41.5% de rendimiento de RAM con un promedio del 37.75% de uso total del CPU. En el intervalo de (63,66] el sistema embebido terminó de realizar las operaciones en el modo pasivo de estrés descrito en el párrafo anterior. La memoria RAM para esta prueba de estrés se mantuvo en 58%, lo que equivale a 536MB. La temperatura del procesador fue de 62.27ºC.

4. Discusión

Para la evaluación del clasificador, la selección de las cartas de baraja se realizó de manera aleatoria, sin embargo, también es posible ejecutarla en forma secuencial. El método utilizado fue el método de validación cruzada proporcionada por la herramienta Matlab con sus valores por defecto. La precisión del reconocimiento de las cartas de baraja se estudió a partir de la matriz de confusión. La precisión del sistema de reconocimiento de cartas fue de 95.71% y la clase con más baja exactitud fue la clase J, que fue confundida con las clases 1 y 7 por el clasificador como se muestra en la *Figura 10*.

		CARTAS RECONOCIDAS													
		0	1	2	3	4	5	6	7	8	9	A	J	K	Q
CARTAS ORIGINALES	0	100%													
	1		80%										20%		
	2			100%											
	3				100%										
	4					100%									
	5						100%								
	6							100%							
	7								100%						
	8									100%					
	9										100%				
	A											100%			
	J		20%						20%				60%		
	K													100%	
	Q														100%

Figura 10. Matriz de confusión

La bondad de la clasificación para esta clase puede ser evaluada mediante el cálculo de la tasa de verdaderos positivos (VPR) y la tasa de falsos positivos (FPR), definidas mediante las ecuaciones siguientes:

$$VPR = \frac{VP}{VP + FN} \qquad (7)$$

$$FPR = \frac{FP}{FP + VN} \qquad (8)$$

La matriz de confusión de esta clase, calculada a partir de la matriz general, se muestra en la *Tabla 1*. Se ha llamado aquí "positivas" a las cartas pertenecientes a la clase J, y "negativas" a todas las demás.

Tabla 1. Matriz de confusión de la clase J

	CARTAS RECONOCIDAS		
CARTAS ORIGINALES		P	N
	P	60	40
	N	20	1100

De esta matriz se pueden calcular la tasa VPR y la FPR, como se muestra a continuación:

$$VPR = \frac{60}{60 + 40} = 60\%$$

$$FPR = \frac{20}{20 + 1100} = 1.79\%$$

5. Conclusiones y Recomendaciones

En este artículo se ha desarrollado un sistema automático de reconocimiento de cartas mediante un sistema embebido Raspberry Pi 3 como una alternativa económica, de fácil implementación y baja exigencia computacional. El rendimiento del CPU para este sistema está por debajo del 25% del total de la capacidad de procesamiento; y se utiliza un 28.3% de memoria RAM para realizar todas las operaciones hasta llegar a la clasificación de la carta. El algoritmo de clasificación seleccionado es el de k-NN, con un valor de k igual a 1. Dicho clasificador tuvo una precisión de un 95.71% de efectividad; y fue capaz de reconocer las 14 clases de cartas en diferentes tipos de orientación bajo diferentes condiciones de iluminación con un total de 280 pruebas. El tiempo de respuesta del clasificador en promedio es de 1.785 segundos sin utilizar el *text to speech*, mientras que utilizándolo aumentó 2.983 segundos debido a la generación y reproducción del audio, dando un promedio de respuesta de 4.768 segundos.

El reconocimiento automático de una carta de baraja sobre una mesa corresponde a una de las fases de un proyecto mayor que busca la creación de un dispositivo personal de reconocimiento de cartas a ser utilizado en el juego de cuarenta para personas no videntes. Asimismo, la plataforma

física construida podría ser utilizada en otros proyectos de reconocimiento con visión artificial. El aplicativo puede implementarse en cualquier dispositivo móvil Android debido a su compatibilidad con Kivy Android MV y P4A (Taylor, 2015), en consecuencia, el algoritmo no se limita a funcionar en dispositivos embebidos.

Bibliografía

Bradski, G., & Kaehler, A. (2008). *Learning OpenCV.* O'Reilly Media.

Galindo Durán, C. K., Juganaru-Mathieu, M., Áviles Cruz, C., & Vázquez, H. (2010). Desarrollo de una aplicación destinada a la clasificación de información textual y su evaluación por simulación. *Administración y Organizaciones*, 119-131.

Gutiérrez, P. D., Lastra, M., Bacardit, J., Benítez, J. M., & Herrera, F. (2016). GPU-SME-kNN: Scalable and memory efficient kNN and lazy learning using GPUs. *Information Sciences*, 165-182.

Hayward, D. (14 de marzo de 2014). *CNET*. Obtenido de 25 fun things to do with a Raspberry Pi: https://www.cnet.com/how-to/25-fun-things-to-do-with-a-raspberry-pi/

Larcher, L. I., Biasoni , E. M., & Cattaneo, C. A. (2011). ALGORITMO PARA DETECCIÓN DE BORDES Y ULTERIOR. *Mecánica Computacional, Asociación Argentina de Mecánica Computacional*, 2841-2852.

LeCun, Y., Cortes, C., & Burges, C. (s.f.). *THE MNIST DATABASE*. Recuperado el 06 de enero de 2017, de MNIST handwritten digit database: http://yann.lecun.com/exdb/mnist/

Lojo, D., Losada, D. E., & Barreiro, Á. (2009). CIE-9-MC Code Classification with knn and SVM. En *Bioinspired Applications in Artificial and Natural Computation* (págs. 499-508). Santiago de Compostela, España: Springer Berlin Heidelberg. doi:10.1007/978-3-642-02267-8_53

López, G. (2010). *CENSO DE POBLACIÓN Y VIVIENDA (CPV-2010).* Instituto Nacional de Estadísticas y Censos (INEC), Unidad de procesamiento - Dirección de estudios analíticos estadísticos.

Monk, S. (2016). *Raspberry Pi Cookbook.* O'Reilly Media.

OpenCV Organization. (2014). *Geometric Image Transformations*. Recuperado el 06 de enero de 2017, de OpenCV 2.4.13.2 documentation: http://docs.opencv.org/2.4/modules/imgproc/doc/geometric_transformations.html

Raikwal, J., & Saxena, K. (2012). Performance Evaluation of SVM and K-Nearest Neighbor Algorithm over Medical Data set. *International Journal of Computer Applications (0975 – 8887), 50*(14).

Raspberry Pi Foundation. (2016). *Raspberry Pi*. Recuperado el 06 de enero de 2017, de Raspberry Pi - Teach, Learn, and Make with Raspberry Pi: https://www.raspberrypi.org/

Secretaría del Buen Vivir. (2017). *El Sumak Kawsay*. (Secretaría del Buen Vivir - Gobierno Nacional de la Republica del Ecuador) Recuperado el 06 de enero de 2017, de Secretaría Buen Vivir | Ecuador: http://www.secretariabuenvivir.gob.ec/el-sumak-kawsay/

Sudha, L., & Bhavani, R. (2012). Performance Comparison of SVM and kNN in Automatic Classification of Human Gait Patterns. *INTERNATIONAL JOURNAL OF COMPUTERS, 6*.

Suzuki, S., & Abe, k. (1985). Topological Structural Analysis of Digitized Binary Images by Border Following. *CVGIP, 30*(1), 32-46.

Szeliski, R. (211). *Computer Vision Algorithms and Applications.* London: Springer-Verlag. doi:10.1007/978-1-84882-935-0

Taylor, A. (2015). *Getting Started*. Recuperado el 06 de enero de 2017, de python-for-android 0.1 documentation: https://python-for-android.readthedocs.io/en/latest/quickstart/

Tello, S. (2 de julio de 2014). *Los no videntes cuentan con más dispositivos para desenvolverse*. (Grupo El Comercio) Recuperado el 06 de enero de 2017, de El Comercio: http://www.elcomercio.com/tendencias/no-videntes-cuentan-mas-dispositivos-desenvolverse.html

Wikipedia. (28 de diciembre de 2016). *Wikipedia La enciclopedia libre*. Recuperado el 06 de enero de 2017, de Cuarenta (juego): https://es.wikipedia.org/wiki/Cuarenta_(juego)

Zapata, A., Pérez, S., & Mora, J. (2014). Método basado en clasificadores k-NN parametrizados con algoritmos genéticos y la estimación de la reactancia para localización de fallas en sistemas de distribución. *Rev. Fac. Ing. Univ. Antioquia*, 220-232.