# Uso de visión para corrección de lecturas de indicadores de 7 segmentos

Ricardo Díaz-Arroyo, Eduardo Font
rickydiaz05@estudiantec.cr, Eduardofontc@hotmail.com
Área de Ingeniería Mecatrónica
Tecnológico de Costa Rica

**Resumen**

En el presente documento se realiza el informe del planteo y solución al miniproyecto del curso de Sistemas de visión. Se crearon funciones y diversas operaciones básicas que fueron implementadas dentro de una aplicación que identifica los dígitos de un display de 7 segmentos y realiza su lectura o indica si hay alguno dañado. La aplicación fue creada en el lenguaje de programación Python y con la ayuda de las librerías de scikit, numpy, opencv y mathplotlib.

**Palabras clave**

Segmentación, Detección de bordes, Filtrado de imágenes, Distractores

## I. Introducción

Mediante los conocimientos adquiridos sobre sistemas de visión industrial del curso de Sistemas de visión se diseñó una aplicación con interfaz gráfica que lee las cifras mostradas por un dispositivo indicador de cifras de tipo LED de 7 segmentos y que pueda avisar si hay algún dígito con problemas o defectuoso y dando la lectura de cada dígito. Esta aplicación fue probada con diferentes imágenes para lograr los requerimientos a cumplir. Se realizó la interfaz gráfica de la forma más simple para que otras personas puedan usarla.

## II. Requerimientos

### *II-A. Entorno*

Se necesita un dispositivo que muestre cifras en formato LED de 7 segmentos, donde al menos se muestren 4 cifras (por ejemplo, un reloj). El dispositivo debe funcionar de tal manera que se puedan generar diversos números y de tal manera que se puedan generar cifras donde "falle" al menos uno de los 7 segmentos (que tiene que poder ser cualquiera). El fallo puede ser real o bien mediante ocultación de algún segmento con un elemento indistinguible del entorno [1].

### *II-B. Contexto*

La toma de imágenes se debe hacer de tal forma que tiene que haber distractores en el fondo de la imagen. Esto es: en ninguna circunstancia se puede establecer la posición de las cifras y la eliminación del fondo por medio de una única inspección visual, al menos son necesarias dos. El objeto debe poder desplazarse por el campo visual de la cámara y debe ser tolerante a rotaciones a izquierda o derecha de aproximadamente 15 grados [1].

### *II-C. Desarrollo de la aplicación*

No se pueden usar funciones que ejecuten de manera monolítica funciones principales de la metodología que hayan diseñado ustedes. Es decir, no se pueden usar funciones que lean una cifra en un 7 segmentos o que realicen funciones complejas de filtrado sin que haya un diseño previo por su parte. Si pueden usar funciones básicas de visión, ya implementadas (binarizaciones, filtrados, detección de bordes, identificación de objetos en imágenes binarias, etc.) [1].

### *II-D. Salida*

El programa debe devolver las coordenadas de la pantalla LED de 8 segmentos (medida desde un punto arbitrario pero siempre el mismo) así como la lectura de las cifras, advirtiendo de si hay algún error, y devolviendo la lectura corregida) [1].

## III. METODOLOGÍA

### *III-A. Descripción de la solución*

Para la solución que se planteó se diseñó primeramente utilizando un diagrama de bloques con los procesos que se consideraron para poder obtener una solución adecuada. El diagrama de bloques que se planteó se observa en la figura 1

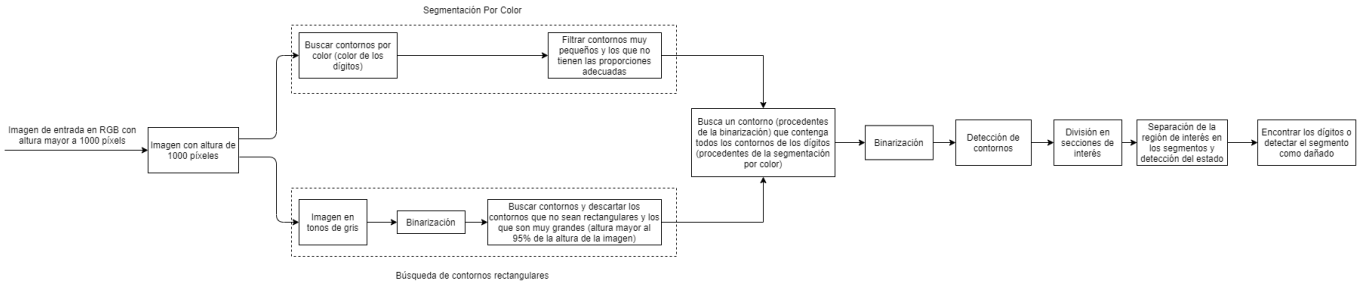


*Figura 1: Diagrama de bloques de la solución planteada. Elaboración propia.*

Para lograr cumplir con los requerimientos se planteó primeramente que la entrada de la aplicación va a ser una imagen de más de 1000 píxeles de altura ya que está hecha para usar con imágenes tomadas con cámaras digitales de un teléfono inteligente que usualmente toman imágenes con una resolución mayor a 2 mega-píxeles.[2]

Luego de esto la imagen es cambiada para que tenga una altura de 1000 píxeles para facilitar el manejo de la misma. A partir de tener esta imagen de 1000 píxeles de altura se realizan 2 funciones básicas de visión para realizar 2 métodos distintos de inspección visual. Uno de estos es mediante la segmentación por color, que primero busca contornos por color usando el color de los dígitos y luego filtra los contornos muy pequeños y los que no tienen las proporciones adecuadas. El otro método es el de búsqueda de contornos rectangulares que convierte la imagen en tonos de gris, la binariza y luego busca los contornos y descarta tanto los que no son rectangulares así como los que sean mayores al 95 % de la altura de la imagen. Para finalizar esta parte del proceso se busca un contorno proveniente de la binarización que tenga los dígitos de la segmentación por color. Para la siguiente parte del proceso luego de localizar los dígitos dentro de la imagen se binariza la imagen para realizar una detección de contornos para dividir las regiones de interés y así poder separar cada región de interés, que en este caso sería cada dígito, y a partir de estas separaciones encontrar los dígitos analizando los segmentos o encontrar algún dígito dañado

### *III-B. Entorno*

Para poder manejar un dispositivo más versátil se utilizó un teléfono ***Xiaomi Redmi 9*** el cual enseña la aplicación ***Gran Reloj Digital*** en la cual se puede seleccionar lo que se quiere representar en pantalla. Para el uso que se le dio se seleccionó solamente el despliegue de la hora y los minutos en color rojo con fondo negro cono se puede ver en la figura 4

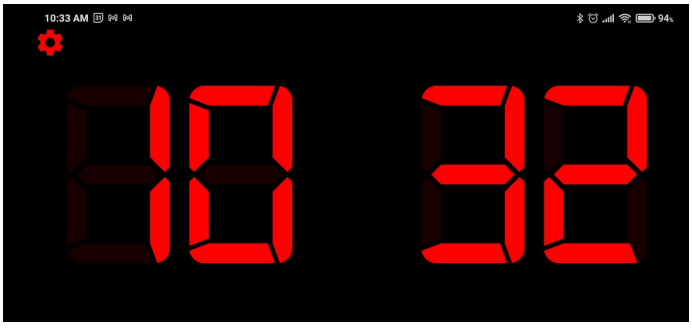


*Figura 2: Diagrama de bloques de la solución planteada. Elaboración propia.*

### *III-C. Contexto*

Para validar el uso de la aplicación y cumplir con los requerimientos se utilizaron objetos como ***Distractores*** en el fondo de la imagen. Estos distractores fueron escogidos para intentar çonfundir.ª la aplicación y así generar una aplicación robusta, tomando en cuenta que el display usado es un rectángulo con fondo negro y dígitos en color rojo. Los distractores usados fueron objetos que puedan confundir en la búsqueda de objetos rectangulares así como objetos rojos y negros para confundir a la segmentación por color, además se incorporaron distractores que también fueran números de color rojo contenidos en un rectángulo. En la figura 3 se puede observar tanto el dispositivo usado como los distractores.


*Figura 3: Diagrama de bloques de la solución planteada. Elaboración propia.*

### *III-D. Desarrollo de la aplicación*

La pantalla principal de la aplicación se muestra en la Figura 4. Para lograr que la aplicación funcionara se utilizaron principalmente 2 métodos distintos de funciones básicas de visión. Cómo se mencionó anteriormente, estas 2 fueron una segmentación por color y una detección de bordes. Para la segmentación por color se buscaron los contornos con el color rojo de los display de 7 segmentos que en ***RGB*** es cercano a ***220,70,70***, adicional a esto se filtraron los contornos muy pequeños y los que tuvieran las dimensiones erróneas a las relacionadas con las dimensiones de los dígitos. Para la detección de bordes se tomó la imagen binarizada en tonos de gris y se buscan los contornos que sean rectangulares, además se descartan los que tengan una altura muy grande, por encima del 95 % del total de la altura de la imagen. En el siguiente link se encuentra el vídeo mostrando el funcionamiento de la misma: **https://drive.google.com/file/d/1nBj4-SiIqsIMQlcukDkigo4yoQIHIeO_/view?usp=sharing**.

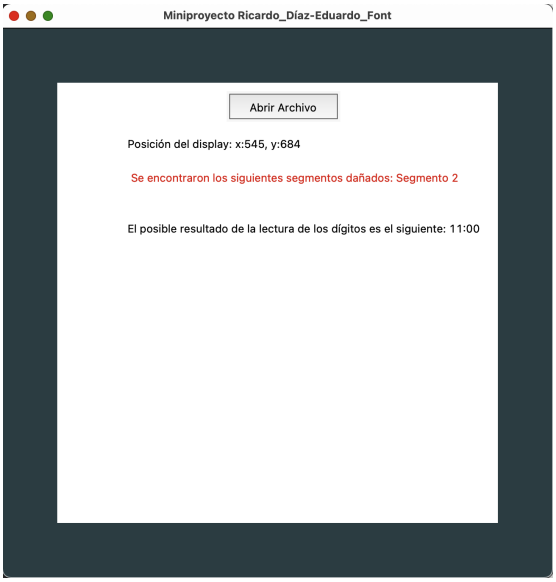


*Figura 4: Pantalla principal de la aplicación. Elaboración propia.*

### *III-E. Salida*

El programa debe devolver las coordenadas de la pantalla LED de 8 segmentos (medida desde un punto arbitrario pero siempre el mismo) así como la lectura de las cifras, advirtiendo de si hay algún error, y devolviendo la lectura corregida

La aplicación toma el contorno rectangular obtenido de la búsqueda de contornos que contiene los contornos de lo dígitos obtenidos de la binarización por color, este contorno es un contorno rectangular. Las coordenadas que se muestran se toman de izquierda a derecha para el ***eje X*** y de arriba hacia abajo para el ***eje Y***, estas coordenadas indican el centroide del rectángulo como se muestra en la figura 5


*Figura 5: Imagen con coordenadas de la región de interés de contorno rectangular. Elaboración propia.*

La lectura de las cifras se realiza tomando cada dígito y analizándolo al colocarle un contorno rectangular para que en lugar de tener una sola región de interés como la de la figura 5 se puedan tener cuatro como se puede observar en la figura 6 y así realizar un análisis por cada uno de los dígitos. La región de interés de cada dígito se binariza y se separa en las 7 regiones de los segmentos. Cada región se analiza y si tiene del 45 % o más de sus píxeles en blanco entonces se considera que el segmento está encendido. Esto se compara con un banco en el que se le dice que numero es según los segmentos que estén encendidos y así se obtiene el valor de cada dígito.

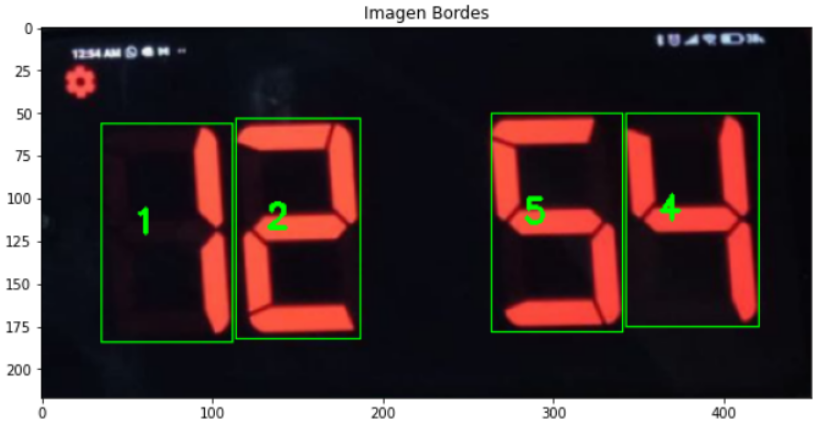


*Figura 6: Resultados de la lectura de dígitos. Elaboración propia.*

## IV. Resultados y Análisis

### *IV-A. Toma de imágenes*

Para un funcionamiento adecuado de la aplicación, es fundamental que las imágenes se tomen correctamente, es significa que se debe procurar que estas tengan una iluminación adecuada y evitar que las posibles superficies reflectoras en la imagen reflejen luz. En la Figura (Figura 7) se muestra una ejemplo de una imagen inadecuada, en la cual hay reflejo de luz sobre el

estuche del celular. En este caso, la aplicación no puede detectar el contorno del teléfono y por lo tanto falla en la detección de rectángulos, ya que no se genera un contorno cerrado al binarizar la imagen (Figura 8).


*Figura 7: Imagen con reflejo de luz sobre el estuche del celular. Elaboración propia.*

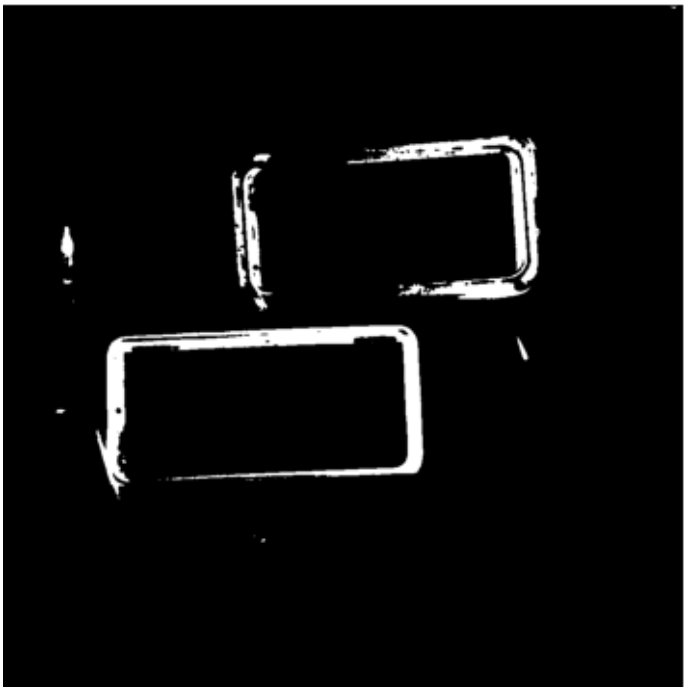

*Figura 8: Binarización de la imagen mal iluminada con contornos abiertos debido al reflejo. Elaboración propia.*

### *IV-B. Segmentación por color y reconocimiento de rectángulos*

En las Figuras 9 a 15 se muestran los resultados obtenidos para la prueba utilizando la imagen de prueba 5. En la Figura 10 se muestra el primer paso que se realiza al cargar una imagen a la aplicación, la segmentación por color. En este caso se detectaron los píxeles con color cercano al color deseado (el color al que brillan los segmentos). En la Figura 11 se muestran los resultados obtenidos al buscar contornos en el resultado anterior (la segmentación por color). Luego, en paralelo (o sea, no

se realiza sobre la imagen obtenida anteriormente, si no sobre otra copia de la original), la aplicación realiza la binarización de la imagen, con el objetivo de encontrar rectángulos. En la Figura 12 se muestra el resultado de que obtuvo para esta prueba y en la Figura 13 se muestran los contornos obtenidos a partir de la imagen binarizada (se muestran los resultados sobre la imagen original). Finalmente, en la Figura 14, se muestra el resultado obtenido para la combinación de los dos resultados obtenidos anteriormente. Esto se hace buscando el rectángulo obtenido que contenga la mayor cantidad de contornos rojos que se obtuvieron en el primer paso. Finalmente, en la Figura 15 se muestra la región del display extraída para la siguiente fase (la región de interés). En el caso de que el display esté ligeramente rotado (hasta aproximadamente 15º), se aplica un operador morfológico para alinear el display horizontalmente para que pueda ser procesado con facilidad en las etapas posteriores. Un ejemplo de esto se ve en la imagen de prueba 7 (Figura 16), la cual tiene el display rotado. En la Figura 17 se muestra la alineación del display.


*Figura 9: Imagen de prueba 5. Elaboración propia.*

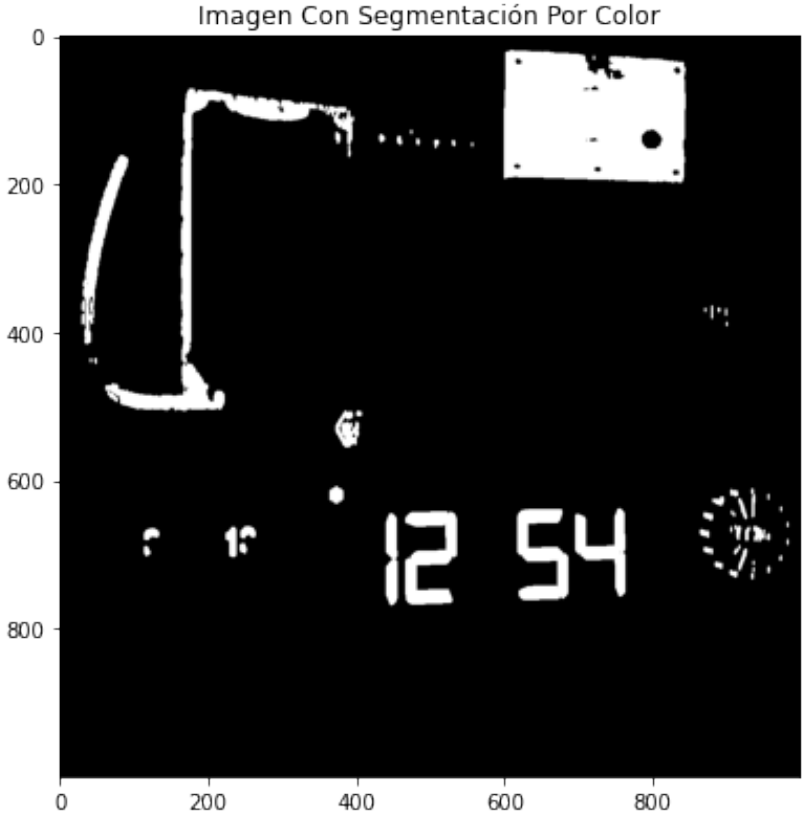


*Figura 10: Resultado de la segmentación por color en la imagen de prueba 5. Elaboración propia.*

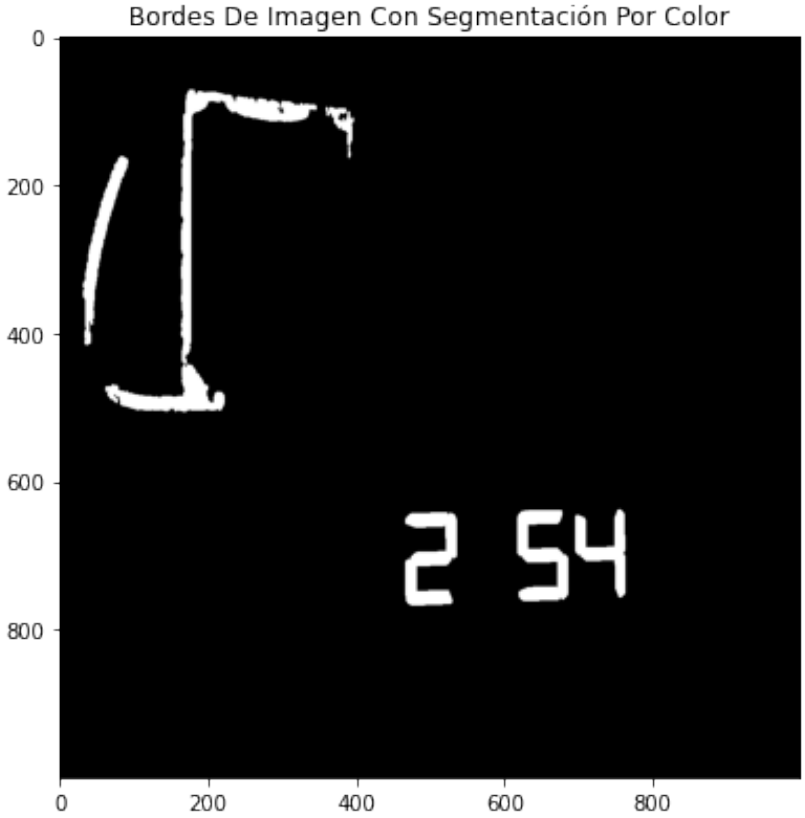


*Figura 11: Contornos encontrados para la segmentación por color de la imagen de prueba 5. Elaboración propia.*

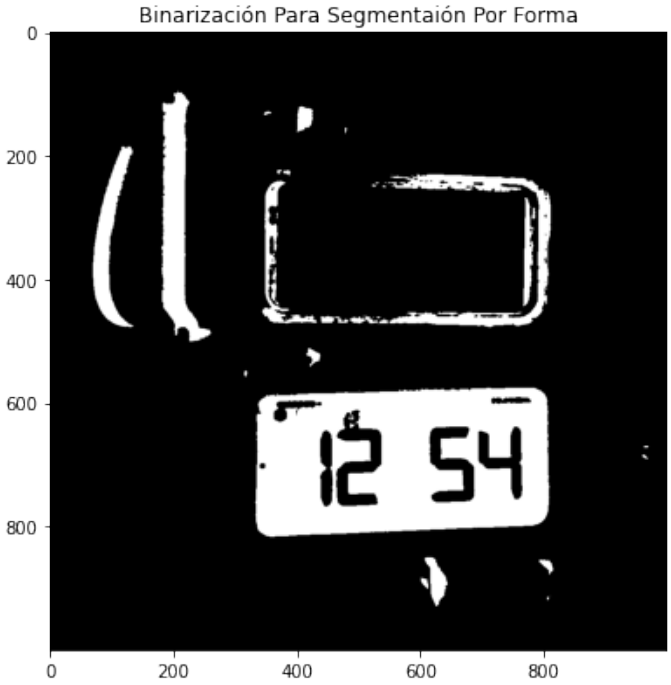


*Figura 12: Binarización de la imagen de prueba 5. Elaboración propia.*


*Figura 13: Bordes encontrados en la binarización de la imagen de prueba 5 (se muestran resaltados sobre la imagen original). Elaboración propia.*


*Figura 14: Display encontrado en la imagen de prueba 5. Elaboración propia.*


*Figura 15: Región de interés obtenida en el procesado de la imagen de prueba 5. Elaboración propia.*


*Figura 16: Imagen de prueba 7, la cual tiene el display rotado. Elaboración propia.*


*Figura 17: Región de interés obtenida en el procesado de la imagen de prueba 7. Elaboración propia.*

### *IV-C. Reconocimiento de dígitos*

A partir de los resultados obtenidos anteriormente (la región de interés del display), la aplicación encuentra las regiones de interés de los dígitos para poder leerlos. En la Figura 18 se muestran las regiones de interés detectadas para el display de la imagen de prueba 5. Esto se hizo binarizando la imagen, buscando contornos y seleccionando los que tuvieran las dimensiones estimadas para los segmentos (hecho por medio de prueba y error hasta encontrar el mejor caso general). Una vez que se tiene la región de cada dígito, se divide cada una de estas regiones en 7 partes donde van a estar ubicados aproximadamente los 7 segmentos del dígito y se detecta si el segmento está encendido o apagado para formar una tupla con los estados de los segmentos y compararlo con el diccionario que contiene cada dígito con su respectiva tupla de estados [3]. Una vez que la aplicación obtiene la tupla de estados de los segmentos, busca los dígitos y los muestra. En la Figura 19 se muestra l resultado final obtenido para la imagen de prueba 5. Se puede ver que se detectaron correctamente los dígitos para esta prueba.

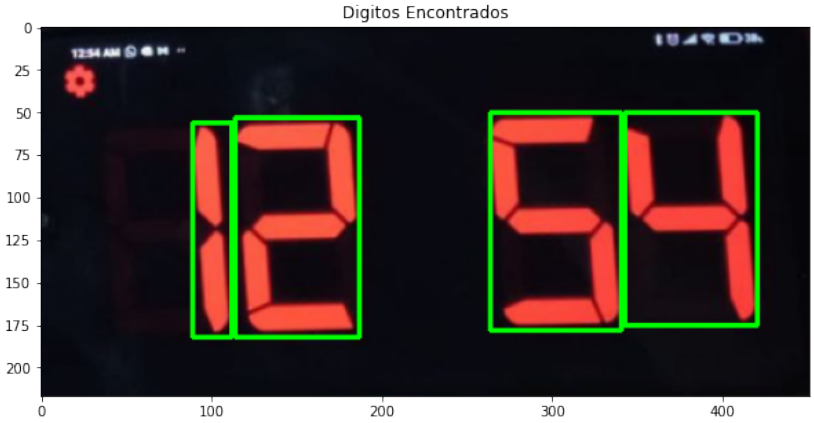


*Figura 18: Región de interés obtenida en el procesado de la imagen de prueba 7. Elaboración propia.*

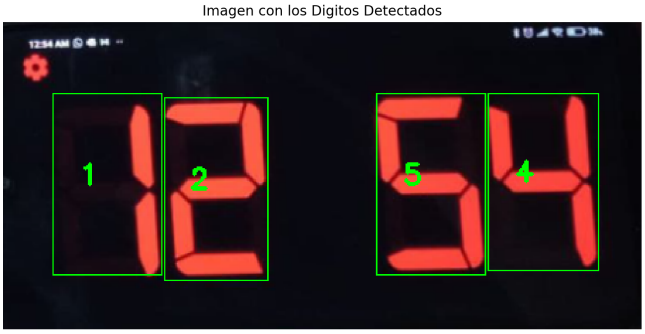


*Figura 19: Resultado final obtenido para la imagen de prueba 5. Elaboración propia.*

### *IV-D. Reconocimiento de segmentos dañados*

En el caso de que se llegara a tener una tupla de los estados de los segmentos donde no calzara con ninguna de las definiciones de dígitos dada, se considera que el dígito tiene segmentos dañados. Para estos casos, la aplicación intenta adivinar el dígito más cercano. Para el caso de la imagen de prueba 9 (Figura 20), la cual contenía un segmento dañado, se obtuvo el resultado mostrado en la Figura 21. En esta prueba se obtuvo como resultado en el tercer dígito una “X”, lo cual es la conducta programada para el programa en estos casos para indicar al usuario que hay problemas en ese dígito.

Luego de encontrar algún dígito dañado, se programó a la aplicación para intentar reconocer el dígito que debería mostrarse. Esto no se puede dar perfectamente en todos los casos debido a ambigüedades, pero si se puede lograr exitosamente en algunos casos. El resultado que imprimió la ventana de la aplicación al ingresar la imagen de prueba 6 fue 10:49, lo cual es correcto ya que en el momento en que se tomó la foto eran las 10:49 p.m. En algunos casos, la aplicación se encontraba dos valores para los dígitos que podrían ser correctos, puramente por cuántos segmentos eran diferentes entre la respuesta obtenida y el posible resultado, pero para reducir las opciones, se agregó un segundo paso en el cual al aplicación reducía la probabilidad de escoger un cierto dígito si en la imagen se detectaba uno o varios segmentos encendidos que en la tupla de estados de los segmentos que definen al dígito debería estar apagado, aumentando la probabilidad de encontrar el dígito correcto.


*Figura 20: Imagen de prueba 9, la cual contiene un segmento "dañado"(tapado con tape eléctrico al confundirse al fondo de la aplicación). Elaboración propia.*

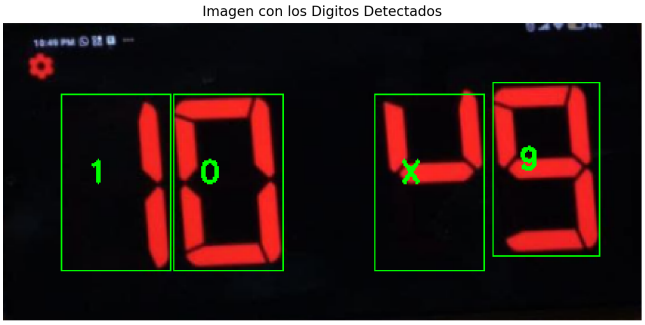


*Figura 21: Resultado obtenido al ingresar a la aplicación la imagen de prueba 9, la cual contiene un segmento dañado en el tercer dígito. Elaboración propia.*

En el caso de segmentos que al dañarse mostraran un dígito que si es posible, como en la imagen de prueba 14 (Figura 22) que se tapó el segmento superior derecho del último dígito, el cual era un 9, y lo hace parecer un 5. Cuando sucedió esto, la imagen detectaba el dígito como si fuera correcto (Figura 23). Esto es en realidad un resultado esperado porque, incluso un humano al ver esa imagen, no podría decir con certeza si el dígito tiene un segmento dañado. Por esto mismo, se consideró que la aplicación detectaba y corregía correctamente (dentro de lo posible) los errores en los dígitos.


*Figura 22: Imagen de prueba 14, la cual contiene un segmento "dañado"(tapado con tape eléctrico al confundirse al fondo de la aplicación) pero que da lugar a otro posible dígito correcto. Elaboración propia.*


*Figura 23: Resultado obtenido al ingresar a la aplicación la imagen de prueba 14, la cual contiene un segmento dañado en el cuarto dígito pero este segmento dañado igual da lugar a un dígito posible. Elaboración propia.*

## V. Conclusiones y Recomendaciones

- En aplicaciones de reconocimiento de imágenes, es de suma importancia no trabajar con valores exactos o cerrados para segmentar o realizar algún tipo de reconocimiento, si no trabajar con rangos ya que reducir mucho los intervalos de valores sobre los cuales se extrae información puede dar a lugar a que la aplicación descarte demasiada información, incluyendo información que es necesaria para el funcionamiento correcto de la aplicación.

- Una forma de hacer que un sistema de visión sea robusto es utilizando procesos y extracción de información en paralelo para luego combinar estos resultados. Esto permite que el sistema sea menos susceptible a distractores y que pueda reconocer en una mayor cantidad de casos la información importante para la aplicación deseada.
- La correcta iluminación en un sistema de visión es fundamental, ya que una aplicación de procesado de imágenes o vídeo a la cual se le ingrese información con mala iluminación, sin importar que tan robusta sea, no va a dar resultados correctos dentro de lo esperado. Por lo tanto, en especial para aplicaciones en tiempo real, es importante asegurar que, no solo se tenga un buena fuente de luz, si no que además esta esté posicionada correctamente y que no hayan reflejos u otros efectos que causen daños a la información fundamental que se desea extraer de las fotos o vídeos.
- Es recomendable mostrar las diferentes etapas de procesado de la foto mientras se desarrollan este tipo de aplicaciones (pero no necesariamente al final ya que puede saturar la interfaz de usuario), esto con el propósito de poder ver el flujo de trabajo del programa y poder detectar posibles errores y espacios para mejora.
- En la toma de imágenes se recomienda el uso de distractores con características similares de la información que se quiere extraer, para que se le pueda dar validez a los criterios usados para la búsqueda.

## REFERENCIAS

[1] J. Crespo, Miniproyecto Sistemas de Visión. 2021.
[2] G. Calatrava, .EVOLUCIÓN DE LA CÁMARA DEL IPHONE, DESDE EL IPHONE 1 AL IPHONE X", Naturpixel, 2017. [Online]. Available: https://naturpixel.com/2017/09/13/evolucion-de-la-camara-del-iphone-desde-el-iphone-1-al-iphone-x/. [Accessed: 20- Apr- 2021].
[3] A. Rosebrock, Recognizing digits with OpenCV and Python - PyImageSearch", PyImageSearch, 2017. [Online]. Available: https://www.pyimagesearch.com/2017/02/13/recognizing-digits-with-opencv-and-python/. [Accessed: 16- Apr- 2021]