

Proyecto Final
Sistema de Reconocimiento de Cartas de Juego

Visión por Computador, MIR 2022
Ing. Fernando Gómez

Introducción

El proyecto consiste en el diseño de un sistema de procesamiento de imágenes para la lectura y reconocimiento de cartas de juego o naipes. A través de un algoritmo realizado en MATLAB, acompañado de una librería de adquisición y procesamiento de imágenes, se puede lograr la obtención del número de la carta y de su símbolo, mostrando los resultados en una GUI la cual detecte automáticamente la carta e inicie el proceso anteriormente descrito. Todo esto englobado con la teoría de Visión por Computador.

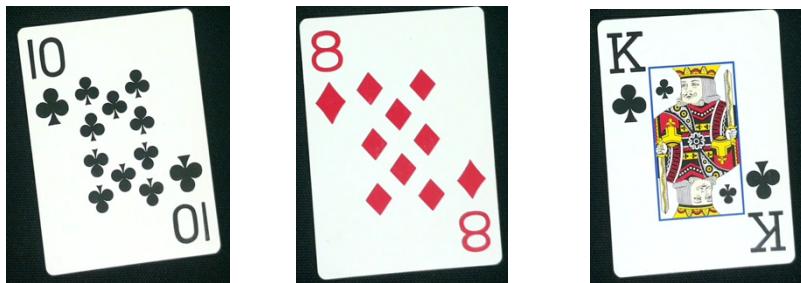


Imagen No.1 – Cartas de juego de distinto símbolo y color (autoría propia)

Todos los scripts, así como cartas modelo para la prueba del algoritmo pueden ser accedidas en el siguiente repositorio:

<https://github.com/fgomez1211/ComputerVision/tree/main/Proyecto>

Marco Teórico

Visión por Computador

Consiste en la extracción automatizada de información de las imágenes. Por información se entiende reconocimiento de objetos, agrupación y búsqueda de contenido y ejecución de acciones en base a los resultados obtenidos.

El procesamiento de imágenes consiste en 3 partes:

1. Adquisición de imágenes digitales
2. Pre-procesamiento de imágenes
3. Localización e identificación

La adquisición de imágenes considera los siguientes componentes:

- Cámara. Toma la información de luz de una escena y la convierte en información digital. Cuanto mayor sea la resolución, más datos recopila el sistema y con mayor precisión se pueden remover los demás componentes del entorno.
- Óptica. Consiste en que la fuente de luz se enfoque adecuadamente mediante lentes para que el sensor capture la imagen con la máxima claridad. Pueden ser lentes de resolución estándar, macro, telecéntricas.
- Iluminación. Uno de los factores más importantes, una imagen no es nada más que la luz con todo su espectro de colores que es reflejada sobre un objeto y capturada por un sensor. Esta debe de ser proporcional y uniforme en todas sus superficies.

Aplicación de Procesamiento de Imágenes (Image Processing)

Una imagen se puede definir como una información visual bidimensional que se almacena y muestra. El procesamiento de imágenes digitales tiene muchos avances como Medicina, Tecnologías de Información Geográfica, Ciencias Espaciales, Aplicaciones Militares, Seguridad y Aplicaciones Industriales según (Artificial, 2022).

Según MATLAB (*Procesado De Imágenes Digitales*, n.d.), el procesamiento de imágenes se realiza utilizando algoritmos informáticos para crear, procesar, comunicar y visualizar imágenes digitales. Los algoritmos se pueden emplear para:

12. Convertir señales de un sensor de imágenes en imágenes digitales
13. Mejorar propiedades como claridad, eliminar ruido y otros objetos.
14. Extraer el tamaño, escala y número de objetos de una imagen.
15. Preparar imágenes para su visualización

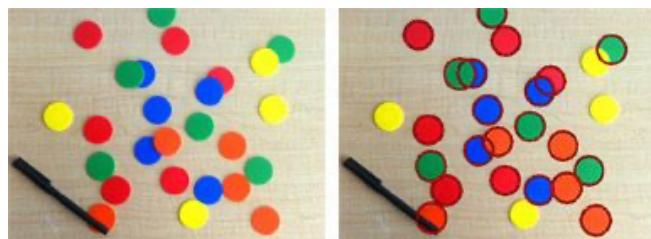


Imagen No.2 – Recuento de objetos circulares (*Procesado De Imágenes Digitales*, n.d.)

Para que una computadora pueda ver una imagen, debe de ser en términos de una función, por ejemplo: $I(x,y)$ o $I(x,y,z)$, donde I es la intensidad de un pixel y (x,y,z) representan coordenadas para la ubicación de cada pixel tanto en una imagen binaria, RGB(Red Green Blue) o en escala de grises respectivamente. Una computadora puede realizar distintas acciones dependiendo de la imagen obtenida. A continuación, se muestra cómo puede una computadora ver diferentes tipos de imágenes.

1. **Imagen en Escala de Grises.** Es una imagen de 8 bits, la cual puede tener 256 intensidades únicas donde cada pixel de 0 representa el negro, un pixel de 1 representa un blanco y los valores de por medio, diferentes tonalidades (Kundu, 2022).

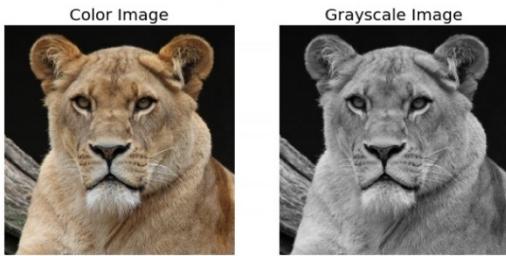


Imagen No.3 – Imagen original y en escala de grises (fuente: [Roberto Reif](#))

2. **Imagen en Color RGB.** Las imágenes utilizadas en el mundo actual son RGB o matrices de 16 como la reconocería una computadora. Pueden tener 65,536 diferentes combinaciones para cada pixel. RGB representa: Rojo, Verde y Azul (por su traducción en inglés) (Kundu, 2022).



Imagen No.4 – Imagen original y separada en cada uno de los canales RGB (fuente: [Chronicles of Fai](#))

3. **Imagen binaria.** Es una imagen cuyos pixeles solo cuentan con dos intensidades: 0 (representa negro) y 1 (representa blanco), razón por la cual son llamadas binarias. (Kundu, 2022) Estas se utilizan para resaltar la sección de una imagen a color, como, por ejemplo:

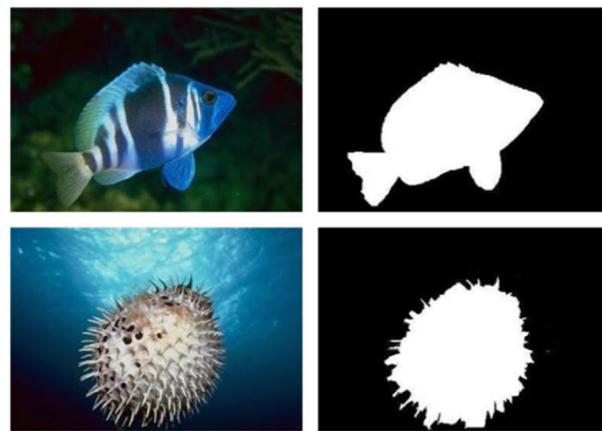


Imagen No.5 – Imagen original y binarizada (fuente: [V7 Labs](#))

Los pasos fundamentales para cualquier procesamiento según (Kundu, 2022) de imágenes digitales son los siguientes:

1. **Adquisición de imagen.** Es capturada por una cámara y digitalizada utilizando técnicas de conversión análogo a digital.
2. **Mejora de la Imagen.** Esta es manipulada para cumplir los requerimientos del trabajo a realizar. Aquí se realizan ajustes básicos de brillo, contraste, análisis con el histograma, etc.
3. **Restauración.** Mejora la apariencia de la imagen ya que el procesamiento y digitalización pueden causar cierta degradación. Aquí se realizan ajustes para quitar el ruido y la difuminación, conocida como blur.
4. **Procesamiento de color.** Ayuda a la computadora a manejar el procesamiento de colores. Se pueden presentar en RGB de 16 bits o RGBA entre otros.
5. **Procesamiento Morfológico.** Los componentes de una imagen son útiles en la representación y descripción de la forma que se requiere extraer para un posterior procesamiento. Este procesamiento cuenta con herramientas, generalmente operaciones matemáticas. Entre las funciones se encuentran: erosión, dilatación que permiten resaltar o difuminar bordes de objetos en una imagen respectivamente.
6. **Segmentación.** Permite dividir la imagen en diferentes secciones para simplificar el procesamiento y analizar solo lo necesario. También ayuda a la computadora a enfocarse en lo necesario permitiendo un mejor desempeño del algoritmo.
7. **Representación y Descripción.** En este paso se decide qué región es tomada como un borde o una región completa. La descripción extrae atributos que suele ser información cuantitativa y de interés para diferenciar un tipo de objeto con otro.
8. **Detección de Objetos y Reconocimiento.** Una vez segmentada la imagen, se puede crear un sistema automático para etiquetar el objeto y dejar ver al usuario que un objeto ha sido detectado.

Diseño

Entorno

El sistema de reconocimiento de cartas de juego se realiza en MATLAB de MathWorks®, el cual es una plataforma de programación y cálculos numéricos avanzados. Este, a través de múltiples librerías, permite realizar operaciones de procesamiento avanzado de imágenes y la creación de algoritmos. La versión utilizada es la R2022b (9.13.0.2049777).

Para el procesamiento, se utilizará una MacBook Pro de 14" con las siguientes características:



Procesador: Apple M1 Pro
Núcleos: 8 (6 de desempeño, 2 de eficiencia)
Memoria RAM: 16GB
OS: Ventura 13.0.1
SSD: Apple SSD NVMe Express

Imagen No.6 – MacBook Pro 14" M1 (Fuente: [Apple](#))

Para el ambiente de trabajo, se escogió una caja de luz la cual tiene las siguientes dimensiones: 30x30x40cm. Este cuenta con 70 piezas de LEDs de alta calidad, que proporcionan un efecto de iluminación constante. La intensidad de la iluminación es ajustable, lo cual permite evitar reflejos hacia la cámara. Esta será alimentada mediante un cable USB especial que cuenta con los controles de encendido e intensidad y una fuente de 5 VDC.

La luminosidad desde las tiras de luz hacia la ubicación de la carta ha sido medida en 1524 lx utilizando la aplicación para Android, Lux Meter.

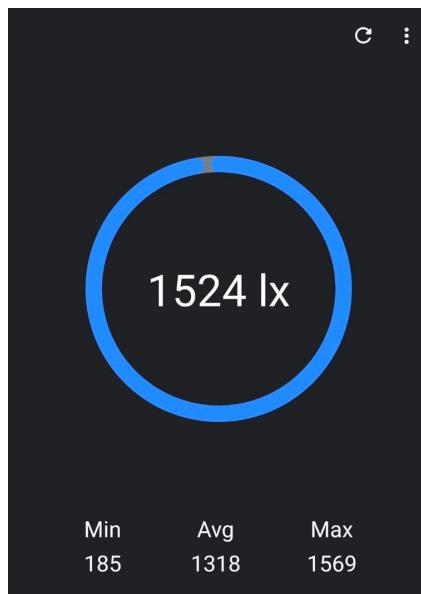


Imagen No.6 – MacBook Pro 14" M1 (fuente: Propia)

Se utiliza un fondo verde el cual permite un alto contraste entre la imagen y su entorno. Esto permite realizar un procesamiento que permita removerlo de una manera fácil y así trabajar únicamente con la carta. Se evitó trabajar con el blanco, negro, rojo y azul ya que las cartas a utilizar cuentan con objetos que contienen estos colores y a la hora de remover el fondo, tampoco remover estos objetos.



Imagen No.7 – Caja de Luz, fondos y conexión (fuente: [Guatemala Digital](#))



Imagen No.8 – Vista real del entorno de trabajo (fuente: propia)

La cámara se ha posicionado sobre el techo de la caja de luz. Por medio de ensayo de error se ha encontrado la posición óptica que permitiera observar el cuadro completo sin ninguno de los bordes internos de la caja. Se han fijado 3 puntos de silicón frío que funcionan como límites para obtener siempre la misma posición.

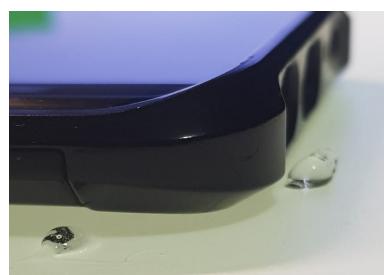


Imagen No.9 – Bordes de silicón frío actuando como posicionadores (fuente: propia)

La distancia focal es de 25cm. Originalmente se trabajó sobre la superficie de la caja, a 30cm, pero el detalle de la imagen era bajo. Esto ocasionaba un error en el momento de utilizar las propiedades de los objetos y realizar las respectivas comparaciones. Se utilizó un objeto el cual contaba con 5cm de altura y los detalles mejoraron. Por lo tanto, se decidió por trabajar a una altura de 5cm obteniendo una distancia focal de 25cm.

Se realizó una medición horizontal sobre lo que observa la cámara dando como resultado 31cm. Esto permitió calcular un FOV Horizontal de aproximadamente 63.6° . Como se observa en la Imagen No.8, se ha elevado la carta por 5cm para obtener mejores detalles de los objetos. Para evitar colores y fondos no deseados en la escena, se ha forrado el objeto con el mismo color verde y cerrados los bordes laterales para evitar que la iluminación externa tenga un efecto sobre la escena.

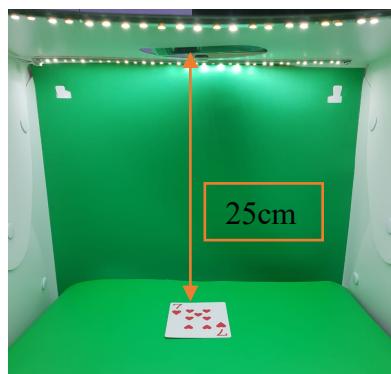


Imagen No.10 – Distancia entre cámara y carta (fuente: propia)

Captura de Imágenes

El proceso de captura de imágenes se realiza mediante la librería 'Image Acquisition Toolbox' (IAT). Este proporciona funciones y bloques para conectar cámaras con MATLAB. Permite detectar y configurar de forma interactiva las propiedades del hardware.

Como dispositivo de captura, se utilizó un Samsung Galaxy Note8 conectado vía puerto USB-C. Para obtener las imágenes, se ha utilizado el software llamado 'Iriun'. Este funciona como conexión entre la cámara y la computadora y debe estar instalado tanto en la computadora como en el dispositivo móvil. En esta ocasión, se ha utilizado un móvil con Android OS.

Las características de la cámara principal, según el fabricante son:

Resolución: 12MP

Apertura: Wide Angle: f/1.7, Telephoto: f/2.4

Zoom Óptico: 2X, Dual Pixel Autofocus

Lente: 26mm

Grabación de video:

4K @ 30fps

1080p @ 60fps

720p @ 240fps



Imagen No.11 – Cámara Samsung Galaxy Note8 (fuente: [Samsung](#))

La razón final por la cual se escogió esta cámara y no una webcam tradicional, es por la calidad en el reconocimiento no solo en detalles sino en colores. Asimismo, la función de enfoque automático permite tener el enfoque 100% sobre la carta. Se realizaron múltiples pruebas con el enfoque manual y para cada reconocimiento de carta se debía ajustar nuevamente. Ya que esta cámara no cuenta con un lente externo, no es posible ajustar el foco de forma fija, por lo que se ha optado por un enfoque automático. Las pruebas realizadas han sido exitosas, como se mostrará en los resultados.

A través de la librería de IAT, se obtuvieron las propiedades adicionales del dispositivo de captura:

General Settings:

```
DeviceID = 1
DiskLogger = []
DiskLoggerFrameCount = 0
EventLog = [1×0 struct]
FrameGrabInterval = 1
FramesAcquired = 0
FramesAvailable = 0
FramesPerTrigger = 1
Logging = off
LoggingMode = memory
Name = YCbCr422_1920x1080-macvideo-1
NumberOfBands = 3
PreviewFullBitDepth = off
Previewing = off
ROIPosition = [0 0 1920 1080]
Running = off
Tag =
Timeout = 10
Type = videoinput
UserData = []
VideoFormat = YCbCr422_1920x1080
VideoResolution = [1920 1080]
```

Color Space Settings:

```
BayerSensorAlignment = grbg
ReturnedColorSpace = rgb
```

Callback Function Settings:

```
ErrorFcn = @imaqcallback
FramesAcquiredFcn = []
FramesAcquiredFcnCount = 0
StartFcn = []
StopFcn = []
TimerFcn = []
TimerPeriod = 1
TriggerFcn = []
```

Trigger Settings:

```
InitialTriggerTime = []
TriggerCondition = none
TriggerFrameDelay = 0
TriggerRepeat = 0
TriggersExecuted = 0
TriggerSource = none
TriggerType = immediate
```

Acquisition Sources:

```
SelectedSourceName = default  
Source = [1x1 videosource]
```

Debido a la sencillez del algoritmo realizado, no se ha necesitado cambiar muchos de los parámetros permitidos por el programa.

Ejecución del Programa

El objetivo del algoritmo es capturar una imagen, procesarla y extraer características de los objetos detectados. A través de estas características, realizar comparaciones con mediciones previamente obtenidas para poder clasificar e identificar la carta. Y a su vez, dar el resultado: número/letra y símbolo.

El algoritmo trabajado en MATLAB cuenta con 5 archivos los cuales se explicarán a detalle:

1. Proyecto_Final.m
2. Script_main.m
3. Script0.m
4. Script1.m
5. Script2.m

Script: 'Proyecto_Final.m'

El script 'Proyecto_Final.m' contiene el código principal para la ejecución del programa. Es importante aclarar que se necesitan los 5 archivos para poder operar el algoritmo.

El código inicia cerrando todas las ventanas y borrando todas las variables creadas previamente (Workspace) y se hace la carga de la estructura 'adt.mat'

Esta contiene un rango de áreas máximas y mínimas en las cuales se puede encontrar el área del símbolo de la carta a analizar. Esta estructura fue creada a partir de 50 mediciones realizadas para cada tipo de carta. En base a estas, se tomaron los valores máximos y mínimos y se realizaron las respectivas calibraciones para poder diferenciar cuando se trata de una carta de Rombos, Tréboles, Espadas o Corazones.

ch	nombre	Amin	Amax
	'Corazones'	3000	3330
	'Treboles'	3331	3520
	'Rombos'	2000	2999
	'Espadas'	3521	4500
	'J'	1500	1900
	'K'	2300	4000

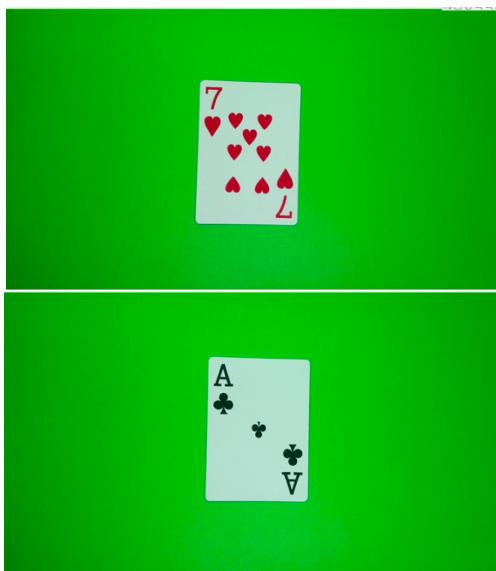
Imagen No.12 – Estructura 'adt.m' (fuente: propia)

Una vez realizada la comparación, se obtiene como resultado el nombre correspondiente.

Se inicia una figura y luego la respectiva secuencia para la captura de imágenes. Debido a que no se ha podido realizar un ciclo de captura de forma continua, se ha creado un ciclo para realizar la captura y procesamiento de 'n' cantidad de imágenes. Esta puede ser ajustada en la variable 'scan'. Por ejemplo, si se define scan=20, el sistema hará 20 ciclos o, en otras palabras, procesará e identificará 20 imágenes.

Este ciclo realiza las siguientes funciones:

1. Ejecuta el ciclo 'scan' cantidad de veces.
2. Ejecuta el Script_main.m (encargado de captura y procesar las imágenes).
3. Almacena la imagen de la carta analizada en la estructura 'resultados.carta'.
4. Almacena el resultado de la carta analizada en la estructura 'resultados.resultado'.
5. Almacena el tiempo en que ha durado el procesamiento del algoritmo, desde la toma de la imagen hasta el despliegue del resultado y se almacena en la estructura 'resultados.tiempo'.
6. Se realiza una pausa de 5 segundos en lo que el usuario cambia la carta.
7. Una vez finalizado el ciclo, se imprimen los resultados obtenidos (solo el resultado, no la carta)
8. Se realiza un promedio de todos los tiempos almacenados en la estructura y se muestra como 'Tiempo promedio de reconocimiento'.



Imagenes 12 y 13 – Capturas por medio del Image Acquisition Toolbox de MATLAB (fuente: propia)

Script: 'Script0.m'

Esta sección tiene como objetivo llamar realizar la captura de imágenes. Se inicia buscando todas las fuentes de video y seleccionando la adecuada. En esta oportunidad, se utilizó la siguiente

```
videoinput('macvideo','1','YCbCr422_1920x1080')
```

La imagen capturada es almacenada en la variable 'frame' la cual se utilizará en el resto del programa. Dentro de este script (líneas 17 y 19) se ha comentado una sección del código para que pueda ser utilizado con imágenes externas y no necesariamente desde la cámara. Las instrucciones se encuentran en el script. Este algoritmo se ha diseñado para procesar una carta a la vez.

Script: 'Script_main.m'

En este script, ocurre el procesamiento de las imágenes. Antes de iniciar el procesamiento, este ejecuta el script anterior (Script0.m) y una vez almacenada la imagen en la variable 'frame', se realiza todo el procesamiento.

Este algoritmo realiza las siguientes funciones:

1. Inicia el temporizador mediante la función 'tic' para medir el tiempo que le toma capturar, procesar la imagen y presentar los resultados.
2. Ejecutar Script0.m y obtener imagen



Imagen No.14 – Imagen Original (frente: propia)

3. Convertir la imagen 'frame' en escala de grises.



Imagen No.15 – Imagen en escala de grises (frente: propia)

4. Se aplica la erosión utilizando como elemento morfológico el diamante con intensidad 0.



Imagen No.16 – Aplicación de la erosión (frente: propia)

5. Binarización y complemento de la imagen.

Como se observa en la siguiente imagen, la umbralización esta bien definida. Esta imagen permite identificar todos los objetos de color negro, los cuales son necesarios para la identificación de la carta.

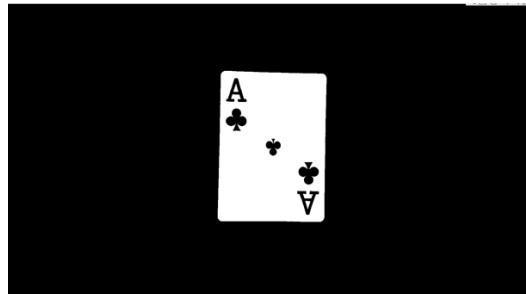


Imagen No.17 – Imagen Binarizada (fruente: propia)

La imagen complementada sirve para obtener la ubicación exacta de la carta (objeto negro) dentro de la imagen original. Esto servirá para realizar una captura más detallada de la imagen original, la cual será mostrada como parte de los resultados. Esta ubicación será almacenada dentro de la estructura 'stats2'.



Imagen No.18 – Imagen Complementada (fruente: propia)

6. Debido a que pueden existir pixeles o grupos de pixeles pequeños los cuales todavía pueden ser identificados como objetos, se procese a filtrar la imagen nuevamente y se remueven los grupos de 3 pixeles. En base a todas las mediciones realizadas, no se encontraron agrupaciones mayores a 3 pixeles. Seguidamente se ha vuelto a binarizar la imagen y a complementarla nuevamente.
7. Se realiza la búsqueda de objetos dentro de la carta binarizada y se obtienen las siguientes propiedades:
 - a. **BoundingBox**. Muestra las coordenadas del objeto dentro de la imagen.
 - b. **Área**. Muestra la cantidad de pixeles negros dentro del objeto encontrado.
 - c. **EulerNumber**. Este parámetro hace un conteo de agujeros o huecos dentro del objeto y da como resultado un número característico que es útil para identificar cierto tipo de objetos.

- d. Estas 3 propiedades han sido seleccionadas para la identificación de las cartas, habiendo muchas más propiedades por considerar.
- e. Las propiedades son almacenadas dentro de la estructura 'stats'.

	Area	BoundingBox	EulerNumber
	1860840	[0.5000,0.5000,1920,1080]	0
	3541	[799.5000,384.5000,76,82]	0
	2536	[800.5000,273.5000,75,87]	0
	1910	[946.5000,494.5000,54,63]	1
	2539	[1.0615e+03,687.5000,80,87]	0
	3516	[1.0665e+03,580.5000,74,83]	0

Imagen No.19 – Estructura 'stats' (frente: propia)

Cuando se realiza el conteo de los objetos, debido a que el fondo es color negro se considera como uno también. Para realizar el conteo real de los objetos, se hace la resta de 1 y se almacena en objetos. Por ejemplo, un As tiene 6 objetos, restando 1 se obtienen 5 objetos como se observa en la Imagen No.20.

- 8. Se crea un ciclo para crear BoundingBoxes alrededor de los objetos encontrados. Estas se crearán a partir de las propiedades encontradas en el inciso anterior. Esta información es visual, permite observar si los objetos encontrados corresponden a los requeridos y que no se hayan encontrado objetos en forma de agrupación de pixeles dentro de la imagen que puedan afectar al conteo real de objetos.

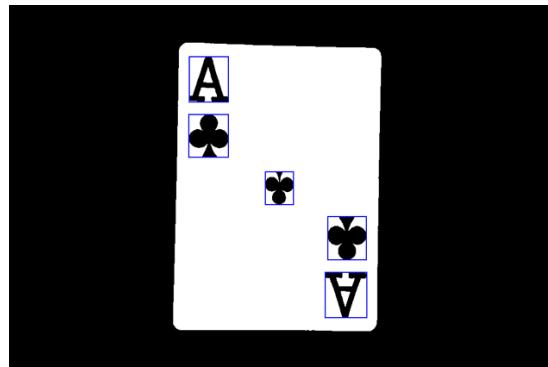


Imagen No.20 – Bounding Boxes alrededor de los objetos encontrados, 5. (frente: propia)

- 9. En base a la posición de la carta, distancia entre carta y cámara e iluminación, el algoritmo diseñado trabaja bajo la suposición de que el tercer elemento a detectar en la estructura 'stats' es el símbolo de la carta. Es importante mencionar que en algunas posiciones la detección no siempre se da igual. Según mediciones realizadas funciona a un ángulo máximo de 26°C sobre la horizontal. En algunos casos se ve afectada cuando la imagen se encuentra sin rotación, puede considerarse como una debilidad.

Una vez identificado el tercer objeto, se procede a extraer el área y realizar la respectiva comparación.

10. Una vez obtenida el área del objeto 3, se procede a realizar la respectiva comparación dentro de la estructura 'adt.mat'. Si el área obtenida se encuentra dentro de alguno de los 3 rangos, el ciclo dará como resultado el nombre del símbolo (adt.nombre). Por ejemplo, si el área obtenida es de 3,350, esta se encuentra dentro del rango de 3,331 y 3520 por lo tanto corresponde al símbolo de 'Tréboles'. El resultado se almacena en la variable 'simbolo'.
11. La siguiente sección se encarga de reconocer los números y letras de la carta. Cuando se procesan cartas del 1 al 10, debido al diseño de estas, se ha identificado que estas contienen desde 5 hasta 16 objetos. Por ejemplo, la siguiente carta (As de Tréboles) tiene 5 elementos, 3 tréboles y 2 letras. Y el patrón se mantiene hasta la carta 10. Se toma la cantidad de objetos y se hace una resta de 4, dando como resultado del número real de la carta.



Imagen No.21 – Objetos encontrados y resultado 5. (frente: propia)

Por ejemplo: La carta anterior (As de Tréboles) tiene 5 elementos, por lo tanto, la carta es un '1'. Dentro de este algoritmo, se requiere que cuando el resultado de la resta sea igual a 1, se cambie el '1' por 'As'.

Cuando se captura la imagen de una carta 10, se cuentan 16 objetos, por lo cual la resta de 4 no es suficiente debido a que el resultado es de 12. En este caso se hace una nueva excepción en el algoritmo, que cuando el resultado sea igual a 12 (16-4), se cambie el '12' por el '10'.

Esta sección se encuentra dentro de un ciclo el cual se encarga de realizar las respectivas validaciones.

'Script1.m'

Si el número de objetos totales se encuentra entre 5 y 17, se ejecuta 'Script1.m'. Este tiene como objetivo identificar el número de la carta, del As hasta el 10. El resultado del número lo almacena en la variable 'ncarta'.

'Script2.m'

Si el número de objetos se encuentra arriba de 17, se ejecuta 'Script2.m'. Este script tiene como objetivo identificar la letra de la carta a través del área. Esto se realiza considerando que el programa identifica el segundo objeto como la letra de la carta.

Se obtiene el área del segundo objeto y se almacena en la variable 'area2'. Esta se compara en la estructura 'adt.mat'. Si el área se encuentra dentro de la región, se obtiene como resultado la letra de la carta y se almacena en la variable 'lcarta'. Esto es válido para las cartas J y K. Para la letra Q, por el diseño de la carta se hace la validación por medio del Número de Euler.



Imagen No.22 – Carta Q de Tréboles, validación por Número de Euler. (fruente: propia)

Como se observa en la imagen, dentro de la letra 'Q' se observan dos regiones blancas las cuales el sistema identifica como huecos dentro del mismo objeto. Se hace la resta del objeto (1) menos los 2 agujeros dando como resultado el Número de Euler. En este caso: $1-2 = -1$. Por tanto, cuando se realice la validación dentro de las propiedades del objeto, si se encuentra que el objeto 2 de la carta tiene un Número de Euler = -1 se obtiene como resultado 'Q'. Este se almacena igualmente en la variable 'lcarta'.

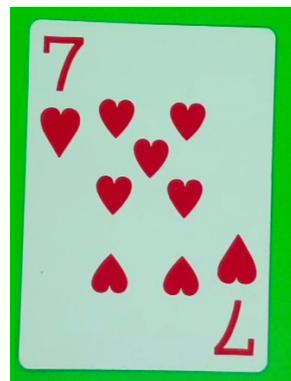


Imagen No.23 – Carta 7 de Corazones, resultados. (fruente: propia)

Cuando termina de validar el 'Script1.m' se concatena el resultado de la operación dentro de la variable 'Resultado'. Por ejemplo, si se toma la Imagen No.23 (7 de Corazones) se obtiene la siguiente información:

Resultado = [ncarta ' de' simbolo]
Resultado = 7 de Corazones

Cuando termina la validación del 'Script2.m' se concatena el resultado nuevamente en la variable 'Resultado'. Por ejemplo, si se toma la Imagen No.22 (Q de Tréboles) se obtiene la siguiente información:

```
Resultado = [lcarta ' de' simbolo]
Resultado = Q de Tréboles
```

12. Se despliegan los resultados. Con la ventana 'figure' abierta, se despliegan 4 imágenes:
 - a. Posición 1,1 – Imagen Original
 - i. Título: 'Imagen Original'
 - b. Posición 1,2 – Imagen en Escala de Grises
 - i. Título: 'Imagen en Escala de Grises'
 - c. Posición 2,1 – Imagen Binarizada
 - i. Título: 'Imagen Procesada'
 - d. Posición 2,2 – Imagen de la carta obtenida de la Imagen Original
 - i. Título: 'Resultado'
13. Al final del proceso se finaliza el temporizador del proceso que se inició en el inciso 1 con la función 'toc'. Este tiempo se almacena en la variable 'tiempo'. Este se utilizará como métrica para validar la rapidez del reconocimiento de las cartas.

A continuación, se muestran los resultados obtenidos:

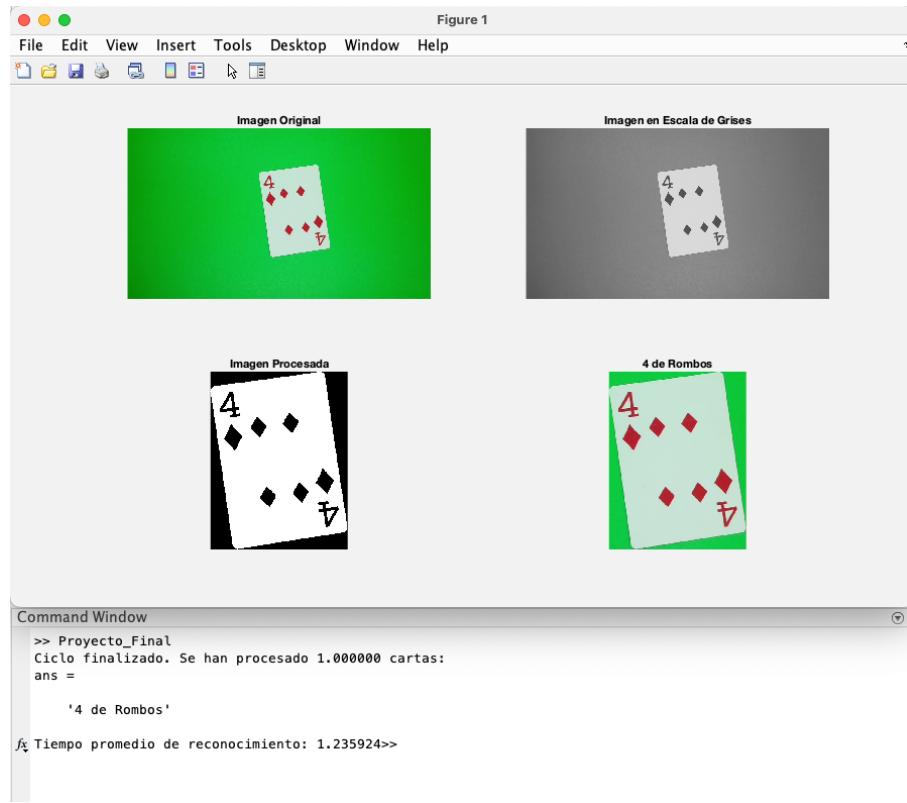


Imagen No.24 – Carta 4 de Rombos. (fruente: propia)

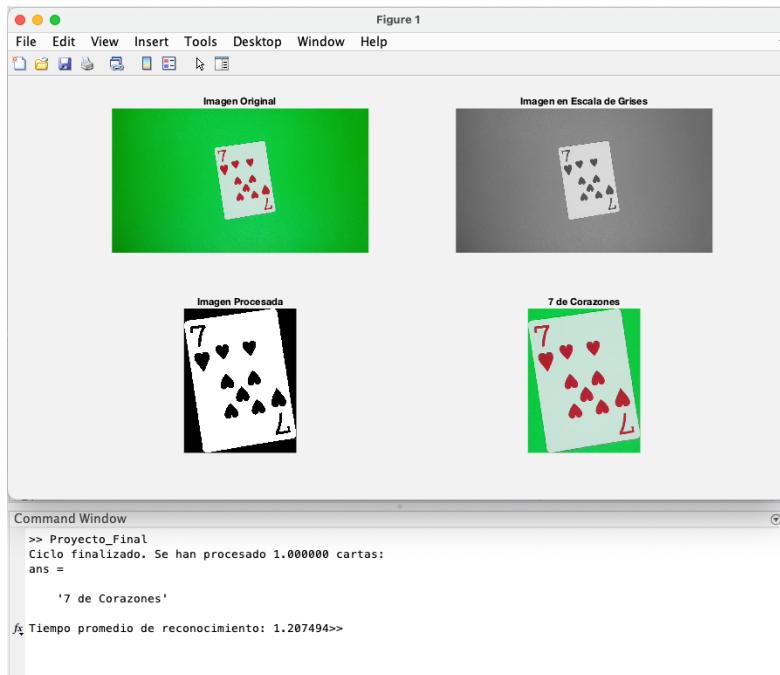


Imagen No.25 – Carta 7 de Corazones. (fruente: propia)

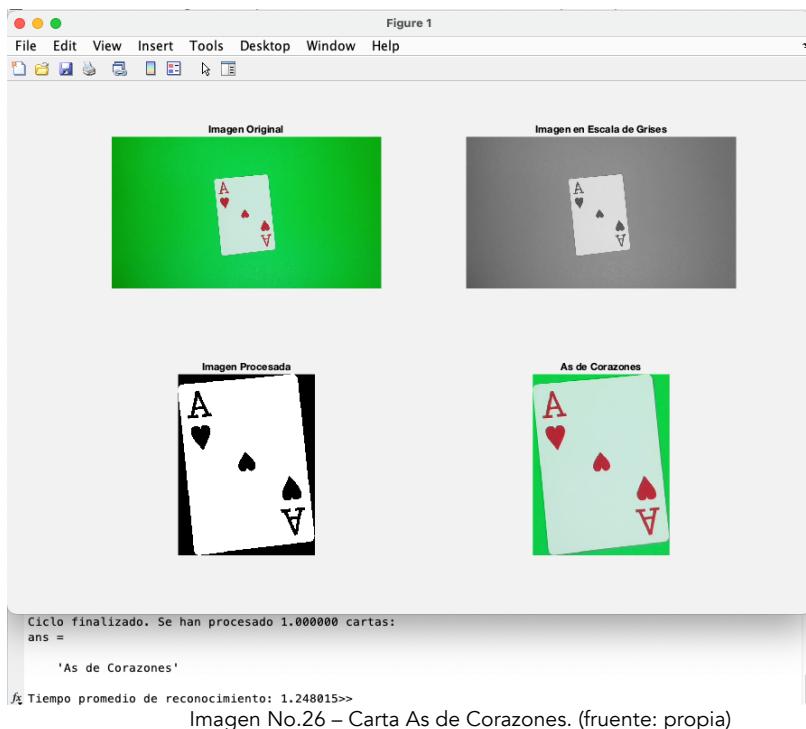


Imagen No.26 – Carta As de Corazones. (fruente: propia)

Cuando se hace la validación de una carta, el tiempo de proceso se encuentra dentro de los 1.2 a 1.3 segundos promedio. Esto ya que el algoritmo debe de reconocer la fuente de video y cargar los archivos necesarios para el procesamiento. Cuando se procesan múltiples cartas, a partir de la segunda el tiempo de proceso se reduce entre 0.6 y 0.7 segundos, dando como resultado un promedio de todos los tiempo entre 0.7 y 0.8 segundos.

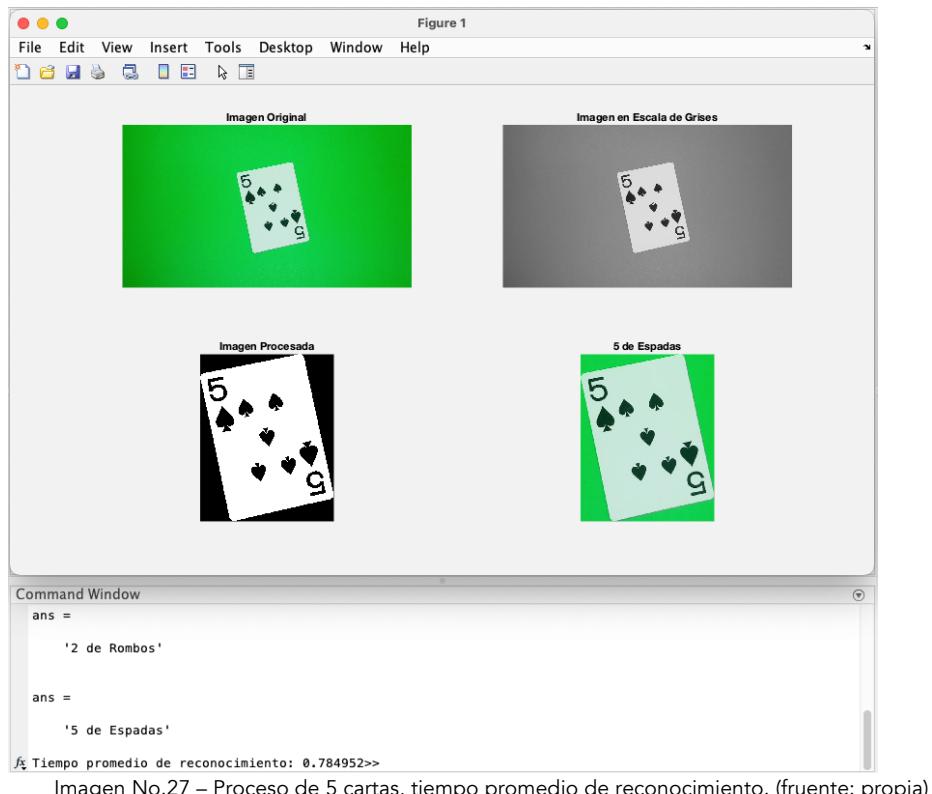


Diagrama de bloques

Debido a la dimensión del proceso, la imagen ha sido almacenada en el [repositorio personal de Github](#) el cual está disponible en formato 'PDF' o 'drawio' para su visualización. De ser requerido puede ser insertada por secciones para la entrega final.

Código Fuente

El programa completo consiste en 5 archivos:

1. Proyecto_Final.m
2. Script_main.m
3. Script0.m
4. Script1.m
5. Script2.m

Código de 'Proyecto_Final.m'

```
%Proyecto_Final.m
%Este módulo tiene como objetivo identificar cartas de juego (número y
%símbolo). Este archivo es el principal del proyecto, que a su vez necesita
%los archivos 'Script_main.m', 'Script0.m', 'Script1.m' y 'Script2.m'.
```

```
%Se cierran las ventanas, se borran las variables y cerca el objeto que
%contiene los parametros de la cartas.
```

```
close all;
clear all;
```

```
%La estructura adt.mat contiene máximos, mínimos que servirán para realizar
%la comparación entre los objetos para lograr el reconocimiento.
```

```
load('adt.mat');
```

```
%Se abre una figura para la proyección de las capturas a realizar y los
%respectivos resultados.
```

```
figure;
```

```
%El siguiente código realiza un ciclo para procesar una cierta cantidad de
%imágenes. Esta cantidad esta almacenada en la variable 'scan'. Una vez el
%ciclo alcance el valor establecido se finalizará la captura y
%reconocimiento de imágenes.
```

```
scan=5;
```

```
for l=1:scan;
    Script_main
    resultados(l).carta=carta;
    resultados(l).resultado=Resultado;
    resultados(l).tiempo=tiempo;
    pause(5);
    if l==scan;
        fprintf('Ciclo finalizado. Se han procesado %f cartas:',scan)
        for m=1:length(resultados)
            resultados(m).resultado
        end
        tpromedio=mean([resultados.tiempo]);
        fprintf('Tiempo promedio de reconocimiento: %f',tpromedio);
    end
end
```

Código de 'Script_main.m'

```
%Inicia la medición del tiempo del ciclo.  
tic  
  
%Ejecuta el Script0. Este se encarga de realizar la captura de las  
%imagenes.  
Script0  
  
%Se convierta la imagen a escala de grises.  
ImgGray=im2gray(frame);  
  
% Se aplica el filtro de erosión usando como elemento estructurante (strel)  
% un diamante con intensidad 0.  
SE=strel('diamond',0);  
Imorph=(imerode(ImgGray,SE));  
  
%Binarización de la Imagen (Quita la escala de grises y lo deja en terminos  
%de 1 y 0.  
BW=imbinarize(Imorph);  
  
%Complemento de la imagen (intercambia los blancos y negros)  
J=imcomplement(BW);  
  
%Se eliminan los puntos que tengan formen un conjunto menor a 3 pixeles y  
%se vuelve a complementar la imagen.  
BW2=bwareaopen(J,3);  
J=imcomplement(BW2);  
  
%Genera las propiedades de todos los objetos dentro la carta y numera cada  
%uno de los objetos, almacenandolos en L,num. Como también detecta el fondo  
%como un objeto, el conteo de objetos reales se almacena en 'objetos'.  
stats=regionprops(BW2,'BoundingBox','Area','EulerNumber');  
[L,num]=bwlabel(BW2,4);  
objetos=num-1;  
  
  
%Generacion de propiedades en la imagen binarizada. Se realiza para  
%encontrar la ubicación de la carta completa y no solo los objetos dentro  
%de la misma. Se utilizará para obtener la imagen de la carta dentro de la  
%imagen original.  
stats2=regionprops(J,'BoundingBox','Area');  
carta=imcrop(frame,stats2(1).BoundingBox);  
J2=imcrop(J,stats2(1).BoundingBox);
```

```
%Encuentra el numero y el objeto. Crea un BoundingBox alreder de los  
%mismos.
```

```
for k = 2:length(stats)  
    thisBB = stats(k).BoundingBox;  
    rectangle('Position',thisBB,'EdgeColor','b','LineWidth',1)  
end
```

```
%Se busca el valor del área del objeto 3 dentro de las propiedades el cual  
%corresponde al simbolo de la tarjeta.
```

```
area3=stats(3).Area;
```

```
%Se realiza una validación dentro de la estructura 'adt.m'. Se toma el  
%valor de 'area3' y se compara con los valores máximos y mínimos y de  
%acuerdo a donde se encuentre, retornará el símbolo de la carta.
```

```
for i=1:4  
    amin=adt(i).Amin;  
    amax=adt(i).Amax ;  
    if (amin<=area3) && (area3<=amax)  
        simbolo=adt(i).nombre;  
    end  
end
```

```
%Para diferenciar entre las cartas del 1 al 10 y las J, Q y K, se utiliza  
%el parámetro 'num'. Una carta del 1 al 10 tendrá como máximo 17 objetos.  
%Por tanto, cualquier tarjeta con un 'num' arriba de 17, será una J, Q o K.  
%Se realizaron 2 scripts:
```

```
%Script1 valida cartas del 1 al 10.
```

```
%Script2 valida las cartas J, Q y K.
```

```
if or(area3 <2000, area3>4500)  
    fprintf('No es posible detectar el simbolo de la carta')  
else  
    if num<=17 && num>=5  
        Script1  
        ncarta=num2str(ncarta)  
        Resultado=[ncarta ' de' simbolo];  
    else  
        Script2  
        Resultado=[lcarta ' de' simbolo];  
    end  
end
```

```
%Realiza un plot de la imagen original almacenada en 'frame' con el título
%'Imagen Original' y como subplot la imagen recortada donde se encuentra la
%carta con el título de 'Resultado'.
subplot(2,2,1), imshow(frame);
title('Imagen Original');
subplot(2,2,2), imshow(ImgGray);
title('Imagen en Escala de Grises');
subplot(2,2,3), imshow(J2);
title('Imagen Procesada');
subplot(2,2,4), imshow(carta);
title(Resultado);

%Finaliza el tiempo de medición del tiempo del ciclo.
tiempo=toc;
```

Código de 'Script0.m'

```
%Se obtiene la fuente de Video.
vid = videoinput('macvideo','1','YCbCr422_1920x1080');
vid.ReturnedColorspace = 'rgb';
src = getselectedsource(vid);
vid.FramesPerTrigger = 1;

%Se crea la ventana y se realiza la captura, almacenandola en 'frame'
for i=0:1
    frame=getsnapshot(vid);
    pause(0.01);
end

% %Se se utilizar con imagenes previamente capturadas, comentar la sección
% %anterior y descomentar las siguientes 2 líneas
% frame=imread('');
```

Código de 'Script1.m'

```
%Este ciclo identifica la carta en base a la cantidad de objetos.  
for n = 1:num  
    if n==num  
        ncarta=objetos-4;  
        if ncarta==1  
            ncarta = 'As';  
        elseif ncarta==12  
            ncarta = 10;  
        end  
    end  
end
```

Código de 'Script2.m'

```
%Obtiene el Número de Euler del objeto 2, que corresponde a la letra de la  
%carta. Asimismo, el área del mismo objeto. Como resultado se obtiene la  
%letra de la carta
```

```
neuler=stats(2).EulerNumber;  
area2=stats(2).Area;  
  
if num>17  
    if neuler==1  
        lcarta='Q';  
    else  
        for i=5:6  
            val3=adt(i).Amin;  
            val4=adt(i).Amax;  
            if (val3<=area2) && (area2<=val4)  
                lcarta=adt(i).nombre;  
            end  
        end  
    end  
end
```

Diagrama de Procesos

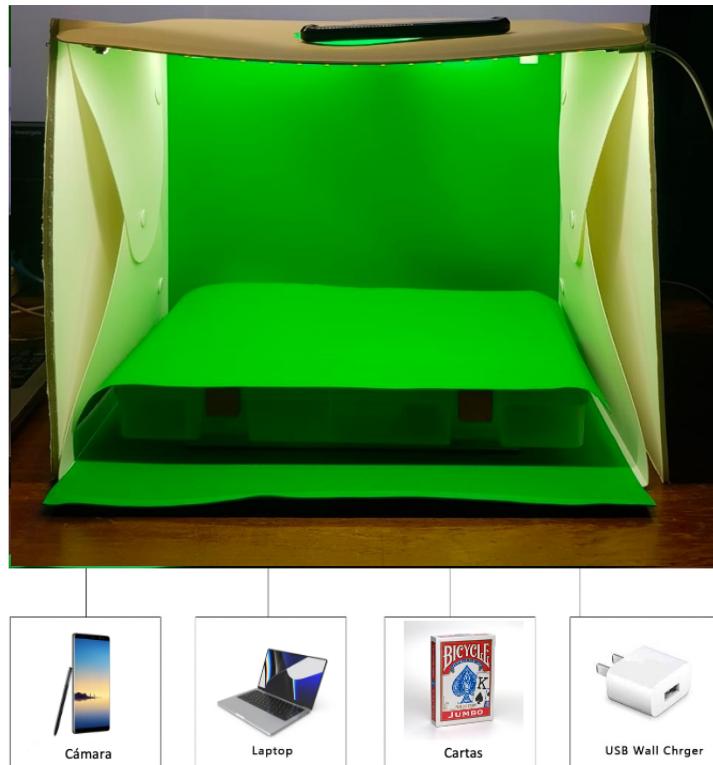
Es importante mencionar que tanto el código como el diagrama de procesos pueden obtenerse en el repositorio:

<https://github.com/fgomez1211/ComputerVision/tree/main/Proyecto>

Planos

Se presenta un esquema general del sistema a implementar. Este cuenta con los siguientes componentes:

- Caja de Luz. Ambiente controlado de iluminación y color donde se realizará la captura de imágenes.
- Cámara. Samsung Note8.
- Laptop. MacBook Pro 14" M1
- Cartas. Baraja de cartas grandes Jumbo.
- Cable y conector USB para control de iluminación.
- Uniformidad en la superficie inferior, también en color. ****Mejora****



Limitaciones

1. Únicamente puede procesar una carta a la vez, no funciona con múltiples cartas.
2. No tiene función de captura y detección automática. Este algoritmo ha sido diseñado para ingresar la cantidad de cartas que se quieren procesar, con una pausa de 5 segundos entre cada iteración.
3. Este algoritmo funciona bajo la suposición de que MATLAB reconocerá ciertos objetos en cierto orden. Una vez cambie el orden según lo previsto, cambiarán los resultados. Esto puede variar según la orientación de la carta.
4. Las cartas pueden tener una ligera inclinación de +18° sobre la horizontal. Mas allá de esta inclinación y el algoritmo reconocerá los objetos en un orden distinto afectando el resultado. En varias ocasiones el algoritmo a procesado imágenes fuera de esta inclinación, pero no se pueden garantizar los resultados.

Oportunidades de Mejora

1. El área entre las cartas de tréboles y espadas son muy similares. Esto se puede solucionar elevando aún más la carta para obtener un mejor detalle de los objetos y poder diferenciar mejor. Aunque esto reduciría el FOV y disminuiría las posibilidades de poder procesar múltiples cartas en algún momento.
2. Cambio en el proceso de identificación. Trabajar por medio del orden de detección de objetos puede afectar en el desempeño del algoritmo. Se debe de buscar implementar todo tipo de validaciones para darle una mayor fiabilidad al proceso.
3. Mejorar la sujeción de la cámara a la estructura.

Conclusiones

Visión por Computador es una herramienta versátil para el procesamiento de escenas y a través de resultados ejecutar ciertas acciones. Es sumamente importante lograr controlar todos los parámetros de la escena como iluminación, enfoque etc. Una imagen bien capturada representa un 50% del trabajo.

Las escenas siempre serán diferentes al igual que los elementos a reconocer. Es importante ser flexible a la hora de realizar los algoritmos y tener muy en cuenta que es lo que se requiere procesar, que características o propiedades se quieren obtener.

MATLAB es un software que cuenta con todas las funciones para la captura, preprocesamiento y obtención de información de imágenes, sin embargo, tiene limitaciones en temas de velocidad de procesamiento. Para aplicaciones industriales, se deben utilizar cámaras y de alta velocidad y lenguajes de programación aptos para procesos rápidos, como lo es 'C'.

Bibliografía

1. Briega, L. R. E. (s. f.). *Visión por computadora - Libro online de IAAR*.
<https://iarbook.github.io/vision-por-computadora/>
2. *Image Acquisition Toolbox*. (s. f.). MATLAB. <https://www.mathworks.com/products/image-acquisition.html>
3. *Image Processing Toolbox*. (s. f.). MATLAB. <https://www.mathworks.com/products/image.html>
4. Qualitas Editorial Team. (2022, 6 septiembre). *Image Acquisition Components*. Qualitas Technologies. <https://qualitastech.com/image-acquisition/image-acquisition-components/>
5. *Procesado de imágenes digitales*. (n.d.). MATLAB & Simulink. <https://la.mathworks.com/discovery/digital-image-processing.html>
6. Artificial, B. I. (2022, January 21). *Procesamiento de imágenes digitales con MATLAB*. Brita Inteligencia Artificial. <https://brita.mx/procesamiento-de-imagenes-digitales-con-matlab>