

# IDENTIFICACIÓN Y ANALISIS DE LOS OJOS MEDIANTE DETECCIÓN DE OBJETOS EN CASCADA Y SEGMENTACIÓN DE IMAGEN

Ceballos Barrios Martin Camilo - Quintana Fuentes Jose Daniel

Procesamiento de Contenido Multimedia

Universidad del Magdalena

Santa Marta, Magdalena

[martinceballoscb@unimagdalena.edu.co](mailto:martinceballoscb@unimagdalena.edu.co)

[josequintanadf@unimagdalena.edu.co](mailto:josequintanadf@unimagdalena.edu.co)

**Resumen**—En este laboratorio se realizó el reconocimiento de cuando cerramos o abrimos los ojos por medio de video-imágenes, se sabe que primeramente se hace una lectura de los ojos y luego se define de qué manera están, si abiertos o cerrados, usamos MatLab para el desarrollo de este laboratorio.

**Palabras Clave:** *reconocimiento, ojos, definir.*

**Abstract**—In this laboratory we performed the recognition of when we close or open our eyes by means of video-images, it is known that first a reading of the eyes is made and then it is defined in which way they are, if open or closed, we used MatLab for the development of this laboratory.

**Keywords:** *recognition, eyes, define.*

## OBJETIVOS:

- Desarrollo de reconocimiento de ojos por medio de espacio de colores.
- Uso efectivo de la detección de cascada de objeto basado en Machine Learning.

## I. MODELO TEORICO

Las aplicaciones de vídeo muestran retos comunes pero complicados que precisan un análisis flexible y funcionalidades de proceso. Los productos MATLAB® y Simulink® admiten desarrollar soluciones para encontrarse los retos más habituales del procesamiento de vídeos, tales como estabilización del vídeo, creación de mosaicos de vídeos, detección de objetivos y seguimiento. El seguimiento de objetos es una parte fundamental de muchas aplicaciones, como evitación de peatones, seguridad y vigilancia o realidad aumentada. El procesamiento de vídeos se puede aprovechar para detectar y contar objetos que se mueven en secuencias de vídeo [1].

MATLAB® suministra materiales y algoritmos que permiten visualizar, analizar, leer y escribir vídeos. El procesamiento de vídeos puede resultar útil en aplicaciones como las siguientes:

- Reconocimiento de objetos con deep learning
- Cálculo del movimiento mediante flujo óptico
- Detección y seguimiento facial

El procesamiento de vídeo es esencial en áreas como deep learning, estimación del movimiento y conducción autónoma.

El reconocimiento de objetos es una técnica de visión artificial para identificar objetos en imágenes o vídeos. El reconocimiento de objetos compone una salida clave de los algoritmos de machine learning y deep learning. Cuando las personas miran una fotografía o ven un vídeo, detectan con rapidez personas, objetos, lugares y detalles visuales. El objetivo es dar lección a un ordenador a hacer lo que resulta nativo para los humanos: obtener cierto nivel de comprensión del contenido de una imagen.

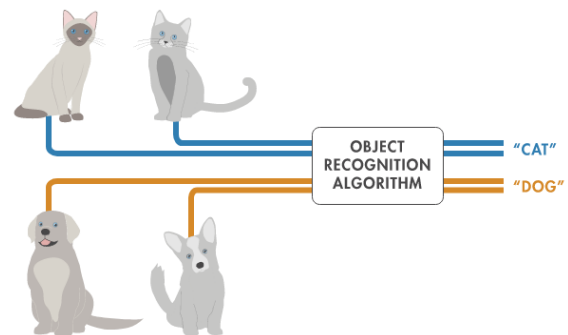


Fig. 1. Utilización del reconocimiento de objetos para identificar distintas categorías de objetos. Tomado de [1].

El reconocimiento de objetos es una tecnología importante presente en los vehículos sin conductor que les admite examinar una señal de stop o distinguir entre un peatón y una lámpara. Además, resulta útil en otras aplicaciones, tales como la identificación de enfermedades en las bioimágenes, la inspección industrial y la visión robótica.

La detección de objetos y el reconocimiento de objetos son técnicas similares para identificar objetos, pero varían en cuanto a su ejecución. La detección de objetos es el proceso de localizar objetos presentes en imágenes. En el caso de deep learning, la detección de objetos forma parte del reconocimiento de objetos, que no solo identifica el objeto, sino que lo localiza en una imagen. Esto permite identificar y localizar varios objetos en la misma imagen [1].

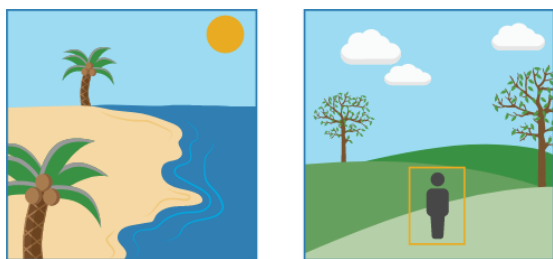


Fig. 2. Reconocimiento de objetos (izquierda) y detección de objetos (derecha). Tomado de [1].

Se pueden acoger diversos enfoques simultáneamente al reconocimiento de objetos. Últimamente, algunas técnicas de deep learning y machine learning se han transformado en enfoques generalizados para los problemas de reconocimiento de objetos. Ambas técnicas aprenden a identificar objetos en imágenes, pero difieren en su ejecución [1].

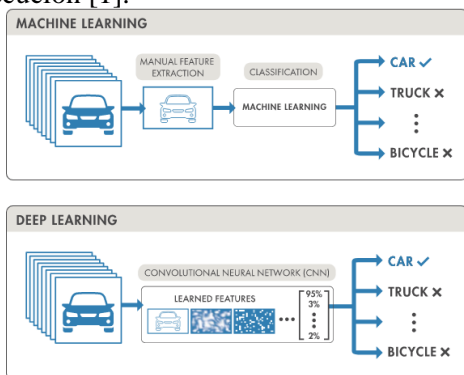


Fig. 3. Técnicas de machine learning y deep learning para el reconocimiento de objetos. Tomado de [1].

Deep Learning o aprendizaje profundo es una forma de aprendizaje automático, en el que una máquina intenta imitar al cerebro humano utilizando redes neuronales artificiales con más de tres capas que le permiten hacer predicciones con una gran precisión.

Las redes neuronales, y más concretamente las redes neuronales artificiales (ANN – Artificial Neural Network), imitan al cerebro humano utilizando un conjunto de algoritmos. A un nivel muy básico, una neurona, de una red neuronal, consta de cuatro componentes principales: entradas, pesos, un sesgo y una salida [2].

#### Modelo Simplificado



Fig. 4. Modelo simplificado de una neurona artificial. Tomada de [2].

Se le llama Deep Learning o aprendizaje profundo simplemente cuando una red neuronal tiene más de tres capas.

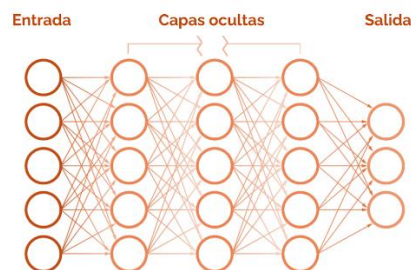


Fig. 5. Red neuronal profunda. Tomada de [2].

Los tipos más simples de redes neuronales son los previamente mencionados y son redes neuronales artificiales (ANN) o redes neuronales multicapa.

Hay dos tipos de redes neuronales que suelen ser muy utilizadas que son las redes neuronales convolucionales (CNN – Convolutional neural network) y las redes neuronales recurrentes (RNN – Recurrent Neural Networks).

#### Redes neuronales convolucionales (CNN)

Una red neuronal convolucional (CNN o ConvNet) es una arquitectura de red para Deep Learning que aprende directamente de los datos, sin necesidad de extraer características manualmente [3].

Estas redes son especialmente útiles para encontrar patrones en imágenes para reconocer objetos, caras y escenas. Además, resultan eficaces para clasificar datos sin imágenes, tales como datos de audio, series temporales y señales.

Las aplicaciones que utilizan reconocimiento de objetos y visión artificial, tales como las aplicaciones para vehículos autónomos y para reconocimiento facial, dependen en gran medida de CNN.

El empleo de CNN con Deep Learning es popular debido a tres factores importantes. Las CNN:

- Aprenden características directamente sin necesidad de extraerlas manualmente.
- Generan resultados de reconocimiento altamente precisos.
- Se pueden volver a entrenar para nuevas tareas de reconocimiento, lo que permite aprovechar las redes preexistentes.

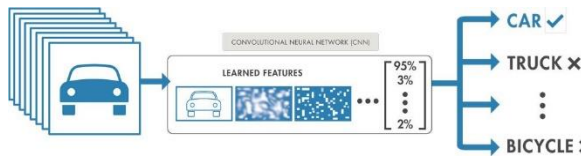


Fig. 6. Flujo de trabajo de Deep Learning. Las imágenes se envían a CNN, que aprende las características y clasifica los objetos automáticamente. Tomada de [3].

Las CNN proporcionan una arquitectura óptima para descubrir y aprender características principales en imágenes y datos de series temporales [3]. Las CNN son una tecnología clave en aplicaciones tales como:

- **Imágenes médicas:** las CNN pueden explorar miles de informes patológicos para detectar visualmente la presencia o ausencia de células cancerosas en las imágenes.
- **Procesamiento de audio:** la detección de palabras clave se consigue utilizar en cualquier dispositivo con un micrófono para detectar cuándo se pronuncia una palabra o frase determinada (como, por ejemplo: "Oye Siri"). Las CNN pueden aprender y detectar con precisión la palabra clave e ignorar todas las demás frases, independientemente del entorno.
- **Detección de señales de stop:** la conducción autónoma se fundamenta en CNN para detectar con exactitud la presencia de una señal u otro objeto y tomar decisiones establecidas en el resultado.
- **Generación de datos sintéticos:** utilizando redes generativas antagónicas (GAN), se pueden producir nuevas imágenes para su uso en aplicaciones de Deep Learning, tales como reconocimiento facial y conducción autónoma.

## Redes Neuronales Recurrentes (RNN)

Las redes neuronales demandantes trabajan con datos secuenciales o de series temporales. Se utilizan mucho para la traducción de idiomas, el procesamiento del lenguaje natural y el reconocimiento de voz. Es la base de muchas aplicaciones populares como Google Translate o Siri [2].

La gran diferencia de las otras redes neuronales es que tienen "memoria". Las salidas de las neuronas son utilizadas de nuevo la próxima vez que se corre el modelo [2].

Normalmente las redes neuronales profundas tradicionales asumen que las entradas y las salidas son independientes entre sí.

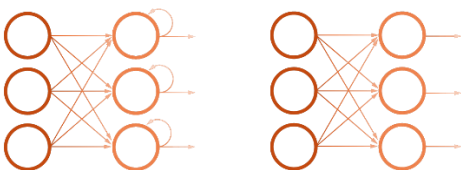


Fig. 7. Retropropagación. Tomado de [2].

A parte de la retropropagación normal igualmente tenemos la retropropagación en el tiempo para este tipo de red neuronal. Todos esos pasos de tiempo se suman rápidamente que crean capas adicionales creando una red neuronal muy profunda en muy poco tiempo.

## Arquitecturas RNN especializadas

Cuando estamos construyendo RNNs, el estado de las etapas anteriores se diluye con el tiempo. Esto puede ser un problema, por ejemplo, cuándo es aprendizaje de estructuras de frases y por ejemplo las iniciales palabras de la frase son muy importantes [2]. Esto se puede contrarrestar con células de memoria. Existe dos tipos de célula que puedes usar:

- **Célula LSTM:** Long short-term memory. Conserva estados separados a corto y largo plazo.
- **Célula GRU – Gated Recurrent Unit.** Célula LSTM resumida que funciona casi igual de bien.

Un espacio de color es un sistema de interpretación del color, es decir, una organización específica de los colores en una imagen o video. Depende del modelo de color en combinación con los dispositivos físicos que permiten las representaciones reproducibles de color, por ejemplo, las que se aplican en señales analógicas (televisión a color) o representaciones digitales. Un espacio de color puede ser arbitrario, con colores particulares asignados según el sistema y estructurados matemáticamente.

Un modelo de color es un modelo matemático abstracto que describe la forma en la que los colores pueden representarse como tuplas de números, normalmente como tres o cuatro valores o componentes de color (por ejemplo, RGB y HSV son modelos de color). Sin embargo, un modelo de color que no tiene asociada una función de mapeo a un espacio de color absoluto es más o menos un sistema de color arbitrario sin conexión a un sistema de interpretación de color.

Se puede crear un amplio rango de colores mediante pigmentos de colores primarios (cian (C), magenta (M), amarillo (Y), y negro (K)). Otra manera de crear los mismos colores es usando su matiz (eje X), su saturación (eje Y), y su brillo (eje Z). A esto se le llama modelo de color HSV.

El modelo de color RGB está implementado de formas diferentes, dependiendo de las capacidades del sistema utilizado. De lejos, la implementación general más utilizada es la de 24 bits, con 8 bits, o 256 niveles de color discretos por canal. Cualquier espacio de color basado en ese modelo RGB de 24 bits está limitado a un rango de  $256 \times 256 \times 256 \approx 16,7$  millones de colores. Algunas implementaciones usan 16 bits por componente para un total de 48 bits, resultando en la misma gama con mayor número de colores. Esto es importante cuando se trabaja con espacios de color de gama amplia (donde la mayoría de los colores se localizan relativamente juntos), o cuando se usan consecutivamente un amplio número de algoritmos de filtrado digital. El mismo principio se aplica en

cualquier espacio de color basado en el mismo modelo de color, pero implementado en diferentes profundidades de color.

En cuanto a la conversión del espacio de color es la traducción de la representación de un color de una base a otra. Esto ocurre normalmente en el contexto de convertir una imagen representada en un espacio de color a otro espacio de color, teniendo como objetivo que la imagen convertida se parezca lo más posible a la original [4]. Además, en Python y openCV se puede hacer un cambio de espacio de color como de RGB a gris, BGRA, YCrCb, XYZ, HSV, Lab, Luv, HLS, YUV usando la función `cvtColor`, esta es una función de openCV, con más de 150 espacios disponibles, que convierte imágenes de un espacio de color a otro [5].

La segmentación es uno de los problemas generales del campo de la visión artificial y consiste en dividir una imagen digital en varias regiones (grupos de píxeles) denominadas segmentos. Más concretamente, la segmentación es un proceso de clasificación por píxel que asigna una categoría a cada píxel de la imagen analizada. Este problema general se divide en problemas especializados, dando lugar por ejemplo a:

- segmentación por color
- segmentación por texturas
- superpíxel
- segmentación semántica

Además, cada problema especializado le otorga un significado propio a las categorías que se usan en la clasificación de los píxeles. Uno de los casos más elementales de segmentación es la umbralización, un tipo particular de segmentación por color con solo dos categorías: claro y oscuro. Cada píxel se clasifica como claro u oscuro comparando su intensidad con una intensidad de referencia dada denominada umbral. El objetivo de la segmentación es localizar regiones con significado. La segmentación se usa tanto para localizar objetos como para encontrar sus bordes dentro de una imagen. El resultado de la segmentación de una imagen es un conjunto de segmentos que cubren toda la imagen sin superponerse. Se puede representar como una imagen de etiquetas (una etiqueta para cada píxel) o como un conjunto de contornos [6].

La app Color Thresholder admite segmentar imágenes en color con umbrales para los canales de color en función de diferentes espacios de color. Con esta app, conseguiremos crear una máscara de segmentación binaria para una imagen en color.

Color Thresholder permite la segmentación en cuatro espacios de color, mostrados en la Tabla 1. En cada espacio de color, la app muestra la imagen, los tres canales de color y el valor de color de todos los píxeles

como puntos de una gráfica de espacio de color 3D. Puede seleccionar los colores contenidos en la máscara disponiendo en ventanas los valores del canal de color o trazando una región de interés (ROI) en la imagen o la gráfica de espacio de color 3D [7].

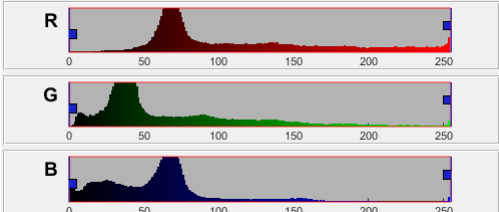
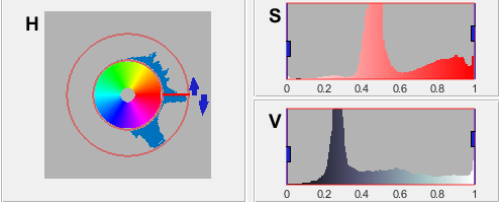
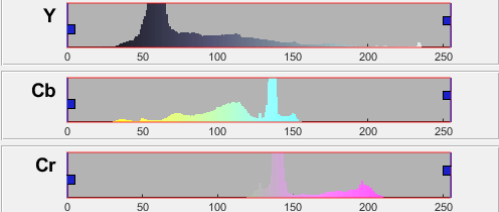
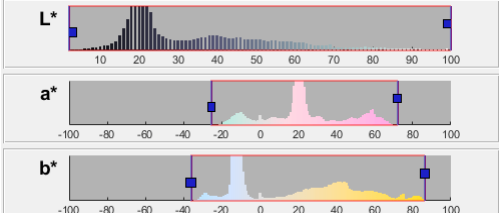
Espacio de color	Controles de umbral de canal de color
RGB	
HSV	
YCbCr	
L*a*b*	

Tabla 1. Espacio de color y Controles de umbral de canal de color. Tomado de [7].

## II. DESARROLLO

Se desarrollará el reconocimiento de los ojos, y además la detección cuando los ojos están abiertos o cerrados:

Inicializando la variable “n” en 1, donde “n” irá de 1 a 2 en el For; hasta 2 debido a que realizaremos dos fotos del video para la comparación.

```
clear all;
n=1;
```

Configurando la captura de video:

```
vid = videoinput('winvideo', 1, 'YUY2_1280x720');
src = getselectedsource(vid);
vid.ReturnedColorSpace = 'rgb';

start(vid)
```

For: Dos capturas de imágenes donde posteriormente se analizarán

```
for n=1:2;
% Inicializando las variables que contendrán la suma
% de la imagen binarizada
SUM1=0;
SUM2=0;

% Capturando una foto del video
I = getsnapshot(vid);
```

Uso de la detección de objetos en cascada para los ojos:

```
% Detección de los ojos mediante la detección de objetos en cascada
Detection= vision.CascadeObjectDetector('EyePairSmall');
```

```
% Pasando a espacio de color de grises
IEG = rgb2gray(I);
region = step(Detection, IEG);
r=region;
t=size(r);
```

Haciendo la delimitación del área detectada, en nuestro caso los ojos.

```
if t(1,1)==true
% Delimitando la imagen en los ojos
Iface=imcrop(IEG,r);
Rface=imcrop(I,r);

figure
bboxes = Detection(I);
Ifaces = insertObjectAnnotatation(I,"rectangle",bboxes,'Cara');
```

Haciendo uso de la aplicación “Color Thresholder” para segmentar la imagen y así identificar la caridad de la Esclera de los ojos. Esto se hizo más sencillo identificarlo en el espacio de colores de HSV.

```
% Uso de la aplicación "Color Thresholder"
% Convirtiendo la imagen RGB al espacio de color HSV
I = rgb2hsv(Rface);

% Definir umbrales para el canal 1 en función de la configuración
% del histograma del Matiz
channel1Min = 0.118;
channel1Max = 0.901;

% Definir umbrales para el canal 1 en función de la configuración
% del histograma de la Saturación
channel2Min = 0.000;
channel2Max = 0.238;

% Definir umbrales para el canal 1 en función de la configuración
% del histograma del Valor
channel3Min = 0.107;
channel3Max = 0.434;

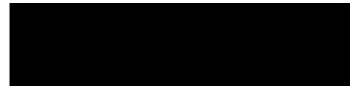
% Creando máscara basada en los umbrales de histograma elegidos
sliderBW = (I(:,:,1) >= channel1Min ) & (I(:,:,1) <= channel1Max) & ...
(I(:,:,2) >= channel2Min ) & (I(:,:,2) <= channel2Max) & ...
(I(:,:,3) >= channel3Min ) & (I(:,:,3) <= channel3Max);
BW = sliderBW;

% Inicializando la imagen enmascarada de salida en función de
% la imagen de entrada.
maskedRGBImage = Rface;

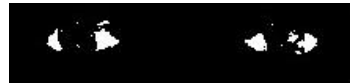
% Estableciendo los píxeles de fondo donde BW es falso en cero.
maskedRGBImage(repmat(~BW,[1 1 3])) = 0;
```

Imagen binarizada de las dos imágenes capturadas del video:

Ojos Cerrados:



Ojos Abiertos:



Dado que la imagen binarizada está compuesta de unos y ceros se puede hacer una sumatoria de estos y así identificar si el ojo está abierto o cerrado. Debido a que la suma de los ojos abiertos va ser mayor al de los ojos cerrados (normalmente la suma de estos es cero o muy baja).

Sumatoria de la variable que compone la imagen binarizada.

```
% Suma de imagenes binarizadas
if n == 1
SUM1 = sum(BW,'all');
else
SUM2 = sum(BW,'all');
end
% Identificación de ojos abiertos o cerrados haciendo uso de la
% suma anterior, teniendo en cuenta que la suma de los ojos
% cerrados será cero con respecto a los abiertos
if SUM1 > 1 || SUM2 > 1
imshow(Rface)
title("Ojos abiertos")
else
imshow(Rface)
title("Ojos cerrados")
end
end

n=n+1;
end

stop(vid)
```



## RESULTADOS:

```
48 % Create mask based on chosen histogram thresholds
49 sliderBW = (I(:,:,1) >= channel1Min) & (I(:,:,1) <= channel1Max)
50 (I(:,:,2) >= channel2Min) & (I(:,:,2) <= channel2Max)
51 (I(:,:,3) >= channel3Min) & (I(:,:,3) <= channel3Max)
52 BW = sliderBW;
53
54 % Initialize output masked image based on input image.
55 maskedRGBImage = RFace;
56
57 % Set background pixels where BW is false to zero.
58 maskedRGBImage(repmat(~BW,[1 1 3])) = 0;
59
60
61 if n == 1
62     SUM1 = sum(BW,'all');
63 else
64     SUM2 = sum(BW,'all');
65 end
66
67 if SUM1 > 1 || SUM2 > 1
68     imshow(RFace)
69     title('Ojos abiertos')
70 else
71     imshow(RFace)
72     title('Ojos cerrados')
73 end
74
```

Ojos cerrados



Ojos abiertos



## III. CONCLUSIONES

Finalmente, se pudo realizar el reconocimiento exitoso de los ojos abiertos y cerrados, para llegar a esto, nos apoyamos inicialmente en la identificaron los ojos mediante la deteccion de cascada de objetos la cual utiliza Machine Learning para identificar todo tipo de ojos. Luego de esto, para lograr la diferenciacion de los ojos abiertos y cerrados nos basamos en la claridad que la esclera del ojo produce respecto a este, por lo que se optó por realizar una segmentacion con espacio de colores, y así, con la imagen binarizada de esta segmentacion poder hallar la imagen en la cual la sumatoria sea mayor, la cual será la de ojos abiertos.

## Referencias

- [1] MathWorks, «Procesamiento de vídeos con MATLAB,» [En línea]. Available: <https://la.mathworks.com/solutions/image-video-processing/video-processing.html>.
- [2] Datademia, «¿Qué es Deep Learning y qué es una red neuronal?,» [En línea]. Available: <https://datademia.es/blog/que-es-deep-learning-y-que-es-una-red-neuronal>.
- [3] Mathworks, «Redes neuronales convolucionales,» [En línea]. Available: <https://la.mathworks.com/discovery/convolutional-neural-network-matlab.html>.
- [4] Wikipedia, «Espacio de color,» [En línea]. Available: [https://es.wikipedia.org/wiki/Espacio\\_de\\_color](https://es.wikipedia.org/wiki/Espacio_de_color).
- [5] kipunaEc, «Cambio de espacios de color openCV - python,» [En línea]. Available: <https://noemioocc.github.io/posts/Cambio-de-espacio-de-color-openCV-python/>.
- [6] Wikipedia, «Segmentación (procesamiento de imágenes),» [En línea]. Available: [https://es.wikipedia.org/wiki/Segmentación\\_\(procesamiento\\_de\\_imágenes\)](https://es.wikipedia.org/wiki/Segmentación_(procesamiento_de_imágenes)).