Laboratorio: Evaluación de la segmentación

# Introducción

A lo largo de los años los seres humanos hemos desarrollado diferentes herramientas conforme a los avances científicos, una de las más importantes se deriva de los dispositivos con cámaras e incluso equipos médicos como lo son las radiografías, los cuales nos permiten capturar un momento y guardarlo para ser utilizado en un futuro con un fin en específico.

Uno de los procesos más importantes que realizan las cámaras, es el de segmentación, ya que nos permite separar e identificar objetos de una imagen o video, eliminado las cosas que no son de relevancia para el análisis o proceso que se requiera realizar la segmentación en la imagen.

En el presente trabajo se utilizan 2 algoritmos para segmentar 2 imágenes y se realiza una comparativa para determinar en que situación un algoritmo funciona mejor que otro.

## Desarrollo

### Algoritmo 1

Una de las técnicas de segmentación más utilizadas, debido a que es una función similar a la visión en el ser humano (la detección de colores), es la segmentación por color, para realizar el presente trabajo se propuso el primer algoritmo, el cual se basa en las técnicas de segmentación por color HSV.

Para entender mejor como funcionan los canales HSV en la librería OpenCV, el autor Gastón Di Giuseppe nos proporciona la siguiente información de utilidad:

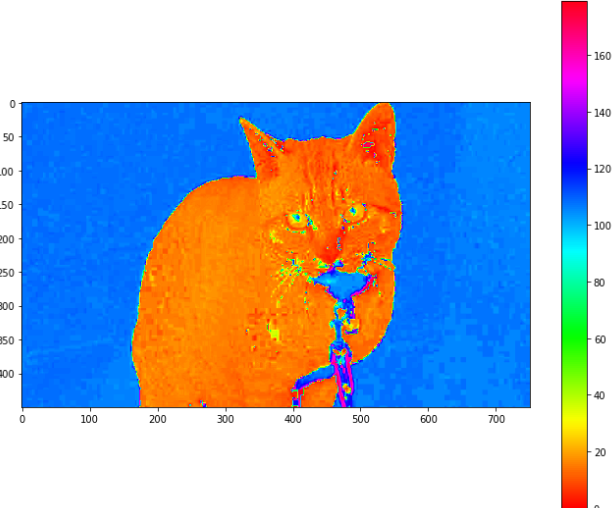
“En OpenCV, los canales de HSV varían así: [H:0–179, S:0–255, V:0–255]. Debemos entender la variación de H como “grados”, que en realidad varía de 0 a 360, como vemos en la imagen, pero en OpenCV lo simplifican en 180 grados. Vemos también que la saturación (S) varía desde el centro hacia afuera y el valor (V) desde lo más oscuro, la base, hasta lo más claro, el techo. “

Entendiendo los canales HSV, se propuso un algoritmo en Python, se implemento una clase en la cual estarán todos los métodos necesarios para realizar la correcta segmentación de los objetos, dentro de la clase ColorHSV se encuentran los siguientes métodos:

* convertirARGB(self, img): Este método sirve para convertir la imagen a una escala de grises.
* convertirAHSV(self,img): Se le pasa la imagen original y se convierte a un formato HSV.
* binRange(self,h,hmin,hmax): Para este método se realiza un proceso de obtención de rangos de los colores, como se observa en la figura 1, el rango para la imagen del gato fue de 5,30 y para el kiwi se aplico un rango de 31,44. Una vez propocionado los rangos, el método realizara un proceso de binarización de la imagen, tal y como se observa en la figura 2.

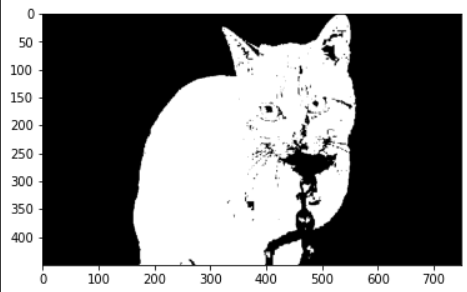
**Figura 1**

*Obtención de los rangos de matiz*



**Figura 2**

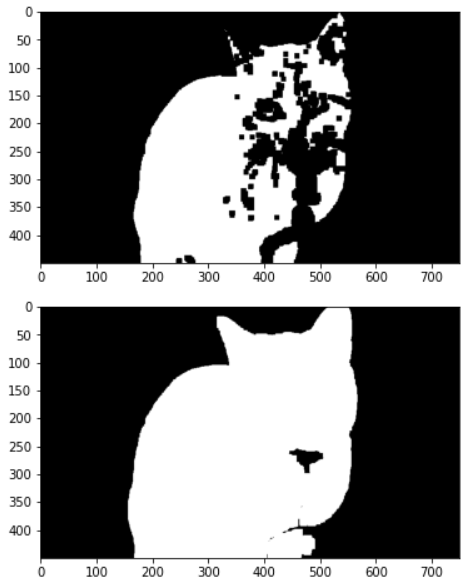
*Imagen después de aplicarle la función binRange(self,h,hmin,hmax)*



* filtradoMor(self,binimg): Con este método se realiza un proceso de apertura morfológica para eliminar el ruido de la imagen en zonas no deseas.

**Figura 3**

*Imagen final después de aplicar el filtrado de apertura*



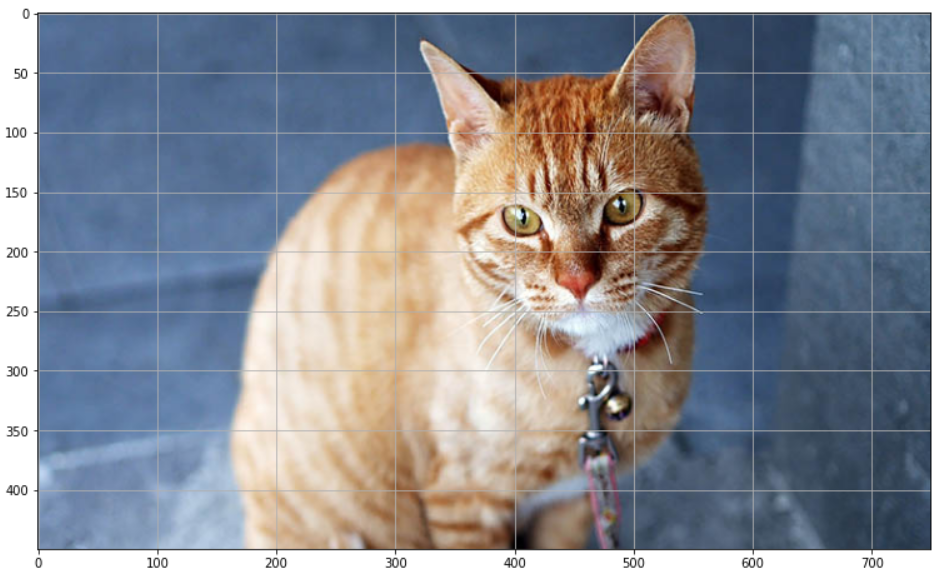
### Algoritmo 2

Este algoritmo se basa en la segmentación por colores RGB, el cual se maneja con 3 colores principales el rojo, verde y azul. Este espacio esta basado en un sistema cartesiano de coordenadas x,y.

Para este algoritmo se propuso la clase llamada ColorRGB, la cual esta constituida por los siguientes métodos:

* convertirARGB(self, img): Este método sirve para convertir la imagen a una escala de grises.
* binRange(self,img,h,l,h2,l2): Mediante esta función se extraerá se elige un rango para extraer el conjunto de pixeles mediante el cual se va segmentar por color. El rango se elige utlizando una foto con las coordenadas a elegir, en el caso de la foto del gato, se tomó las coordenadas (200,300,200,300) .

**figura 4**

*Ejemplo de elección del rango mediante coordenadas*

Los siguientes métodos se definieron para establecer los valores mínimos y máximos para cada uno de los colores rojo, verde y azul de la zona extraída, es por esto que a estas funciones se les pasa las coordenadas de la proporción elegida de la fotografía.

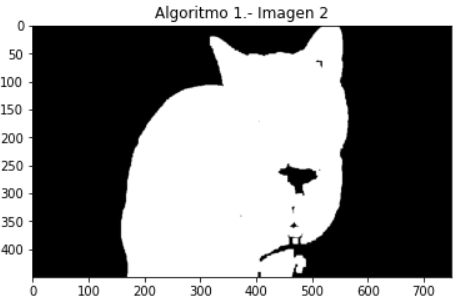
* rangeRmin(self,rI): extracción de la región de interés rango mínimo del color rojo
* rangeRmax(self,rI): extracción de la región de interés rango máximo del color rojo.
* rangeGmin(self,rI): extracción de la región de interés rango mínimo del color verde.
* rangeGmax(self,rI): extracción de la región de interés rango maximo del color verde.
* rangeBmin(self,rI): extracción de la región de interés rango mínimo del color azul.
* rangeBmax(self,rI): extracción de la región de interés rango maximo del color azul.
* uBajo(self,rmin,gmin,bmin): Mediante este método, se obtiene el umbralizado bajo y para esto le pasamos los valores de los rangos mínimos de los 3 colores RGB.
* uAlto(self,rmax,gmax,bmax): Mediante este método, se obtiene el umbralizado alto y para esto le pasamos los valores de los rangos máximos de los 3 colores RGB.
* binarizacion(self,imgRGB,umbral\_bajo,umbral\_alto): Función para realizar el proceso de binarizacion de la imagen, se le pasan los umbrales altos y bajos obtenidos previamente.
* filtradoMor(self,binimg): Con este método se realiza un proceso de apertura morfológica para eliminar el ruido de la imagen en zonas no deseas.

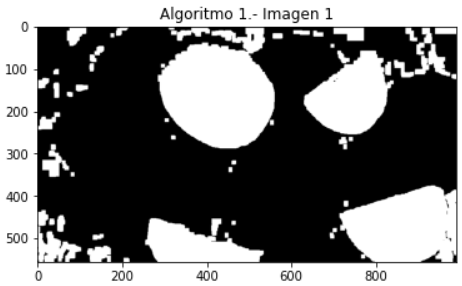
## Resultados

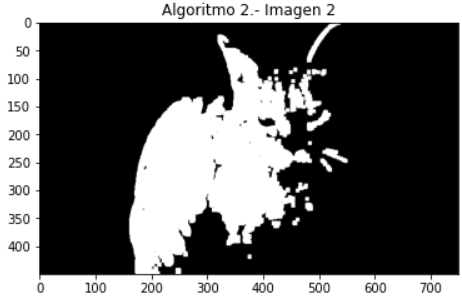
Se probaron los algoritmos con 2 imágenes diferentes, las cuales se muestran a continuación:

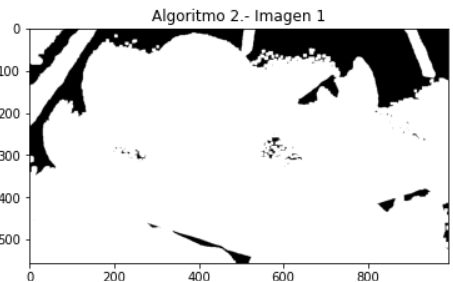
Imagen 1 Imagen 2



Para el algoritmo 1, segmentación por Color CSV los resultados fueron bastantes buenos tal y como se muestran en las siguientes imágenes.



Con el algoritmo 2 se obtuvieron los siguientes resultados en las imágenes.



Para evaluar que tan bueno fue el algoritmo, se utilizo una función llamada jaccard. La cual

# Bibliografía

F. Gimenez-Palomares, J. A.-M. (2016). *Universitat Politecnica de Valencia.* Universitat Polit`ecnica de Val`encia, Modelling in Science Education and Learning. Recuperado el 30 de 12 de 2022, de https://riunet.upv.es/bitstream/handle/10251/69639/4524-15767-1-PB.pdf

Mateu Villa, M. S. (30 de dic de 2017). *Universidad de guanajuato.* Obtenido de repositorio.ugto: http://repositorio.ugto.mx/handle/20.500.12059/4899

jm M. [@jmm5493] (14 de mayo del 2020). Contraste y reducción de ruido en imágenes [Archivo de video]. Youtube. https://www.youtube.com/watch?v=UhW8Kxsnu2g&t=588s

Proyectos JC. [@proyectos-jc] (23 de marzo del 2021). Filtros de imágenes en Python y OpenCv [Archivo de video]. Youtube. https://www.youtube.com/watch?v=b4h-IUfsuq8