FACULTAD DE CIENCIAS EXACTAS, INGENIERÍA Y AGRIMENSURA

Tecnicatura Universitaria en Inteligencia Artificial

Procesamiento de Imágenes I: Trabajo Práctico 2

Alumnos:
Facundo Geuna
Pedro Casado
Máximo Alva

Grupo: 5
Año: 2024

Introducción

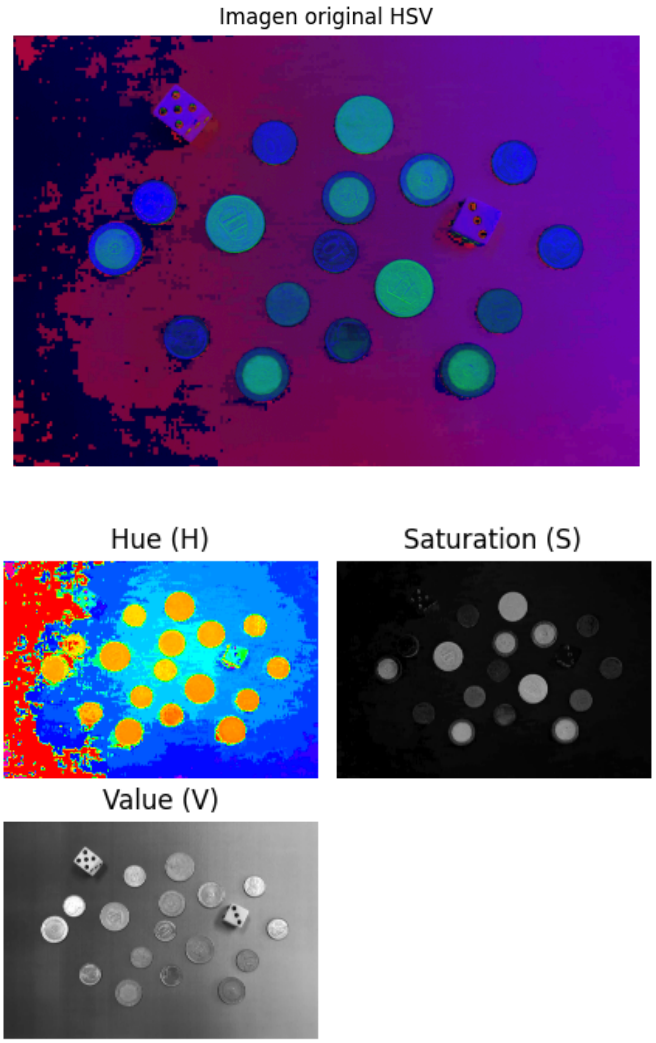
Con el propósito de resolver este trabajo práctico utilizando Python, implementamos las siguientes librerías:

- cv2
- numpy
- matplotlib

Problema 1 - Detección y clasificación de monedas y dados

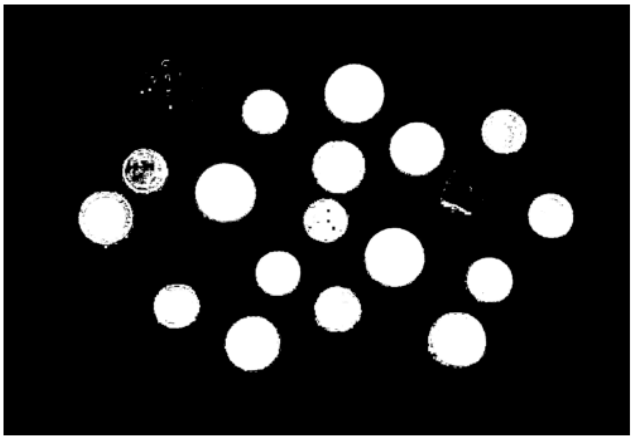
La resolución de esta problemática fue dividida en tres partes, abordando cada uno de los puntos requeridos en el trabajo práctico.

En primer lugar, con el objetivo de procesar la imagen de manera de segmentar las monedas y los dados de manera automática, transformamos la imágen a HSV y separamos los canales Hue (Tono), Saturation (Saturación) y Value (Brillo o Luminosidad).



Observamos que umbralizando el canal H con valores de entre 10 y 30, podríamos diferenciar las monedas fácilmente. A raíz de esto, tomamos la decisión de hacer las búsquedas de dados y monedas por separado.

H umbralado para monedas

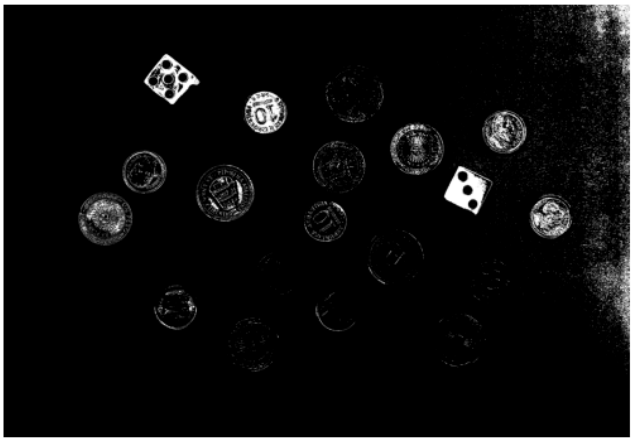


Con este resultado, realizamos la búsqueda de contornos y utilizando como filtro los tamaños de las bounding box de cada uno, eliminamos algunos ruidos menores. A partir de las coordenadas de los contornos resultantes, realizamos la segmentación de cada moneda en la imagen original, obteniendo recortes como este:

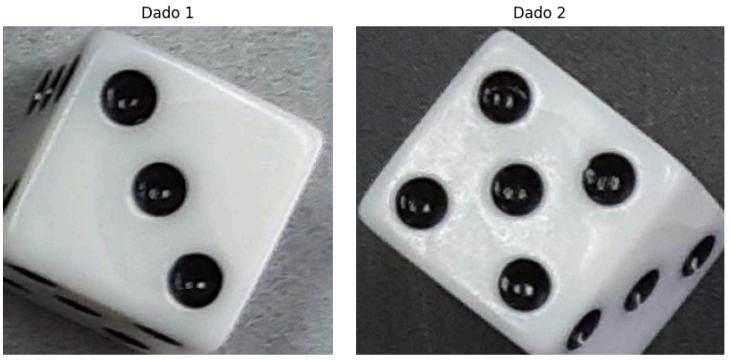
Moneda 1

Para continuar con la detección de los dados, umbralizamos el canal V con valores entre 180 y 220, que es donde observamos mayor contraste entre los dados y el fondo.

V umbralado para dados



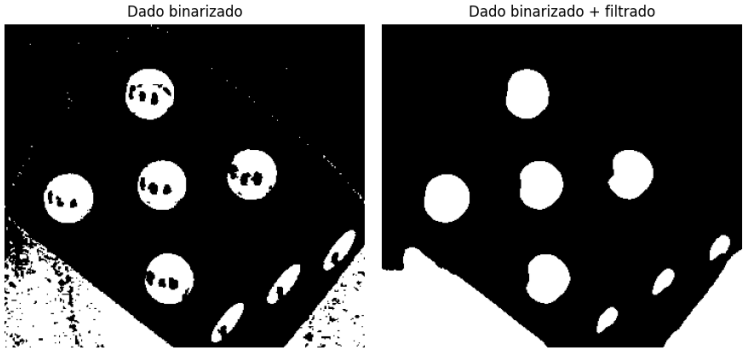
En esta imagen binaria obtenida, observamos resultados mucho más ruidosos que en el caso anterior, pero nuevamente filtrando por tamaño de las bounding box, ya que los dados son más grandes que cualquier otro contorno de la imagen, logramos segmentarlos:

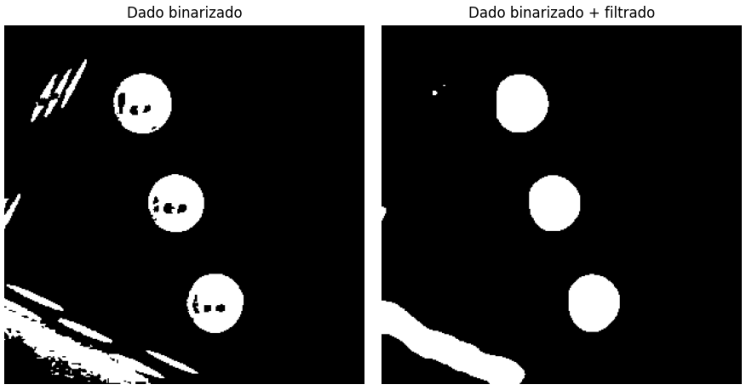


Siguiendo con el apartado B, que propone clasificar los distintos tipos de monedas y realizar un conteo automático, teniendo en cuenta que las monedas de cada tipo presentan distintos tamaños, recorrimos los recortes y a partir de sus áreas hicimos la clasificación. Además, sumamos la cantidad total de cada tipo y la cantidad total de dinero.

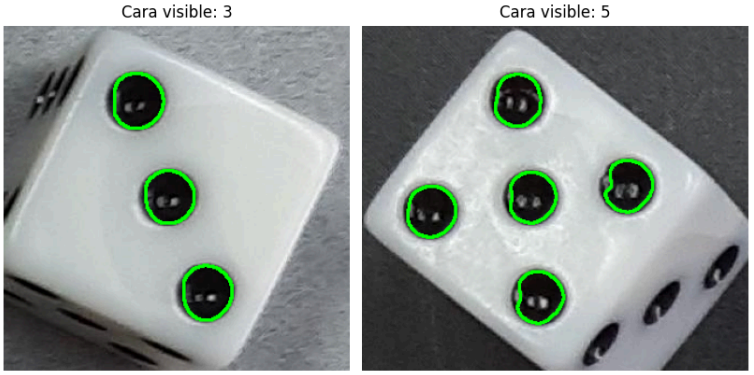
Cantidad de monedas de $1: ***5***
Cantidad de monedas de $0.5: ***3***
Cantidad de monedas de $0.1: ***9***
La suma total de dinero en monedas es de: ***$7.40***

Por último, para resolver el apartado C, que propone determinar el valor que representa la cara superior de cada dado y realizar un conteo automático, recorrimos los recortes de los dados obtenidos previamente y les aplicamos una umbralización, con el objetivo de obtener una imagen binaria con los puntos de los dados con valores de 255 y el todo el resto 0. Observamos cierto ruido dentro de los círculos, el cual resolvimos aplicando un filtro MedianBlur:





Continuamos hallando los contornos y filtrándolos por sus áreas logramos contar únicamente los puntos de la cara superior de cada dado. Finalmente, mostramos los resultados, realizamos el conteo y la suma del puntaje:



*Hay **2** dados.*
*La suma de los dados es de: **8***

Problema 2 - Detección de patentes

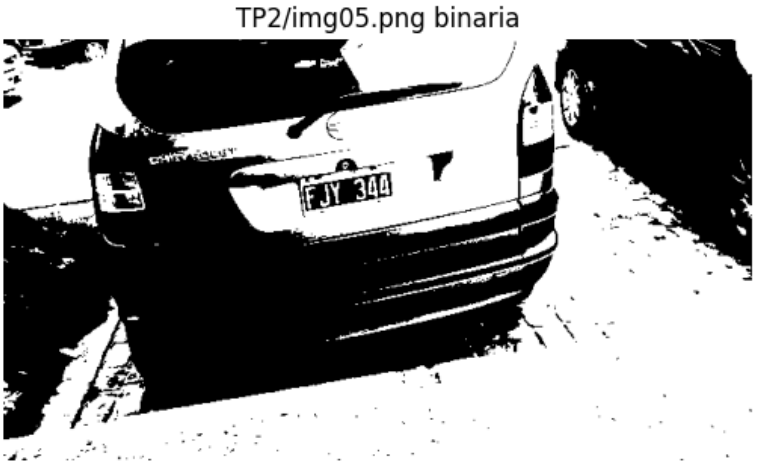
Para la resolución de esta problemática, se elaboró un algoritmo capaz de detectar automáticamente la placa de la patente y la propia segmentación de los caracteres.

Para procesar todas las imágenes de patentes de autos, se define una lista “autos” con todas las rutas a las imágenes y luego, a través de la estructura de control for, recorremos dicha lista.

A continuación se dejan detalles de cada una de las distintas etapas de procesamiento que se configuraron dentro del bucle for.

En primer lugar, cargamos la imagen y la convertimos de BGR a RGB para poder visualizarla correctamente a través de matplotlib. Luego de esto, la convertimos a escala de grises.

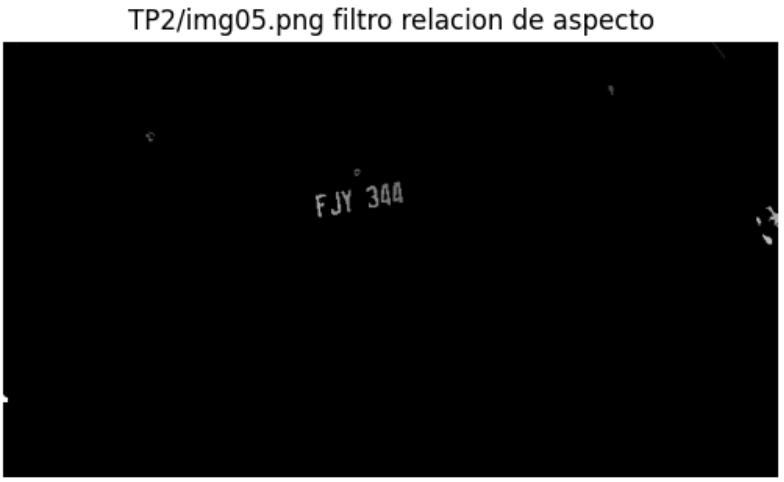
Después aplicamos una binarización en la cual los píxeles mayores o iguales a 124 se convierten en blanco, y los píxeles menores a dicho valor se convierten a negro, obteniendo una imagen binaria para una mejor utilización de la detección de componentes conectadas.



Siguiendo por este camino, a través de cv2.connectedComponentsWithStats, detectamos las componentes conectadas y aplicamos un filtro por área para solamente conservar las

componentes que posean un área entre 12 y 98 píxeles.

Para detectar componentes que tengan forma de caracteres verticales, se aplica un filtro basado en relación de aspecto a través del cálculo: alto/ancho. Luego, definimos que vamos a mantener las componentes donde el resultado del cálculo del aspecto se encuentre entre 1 y 2.5.



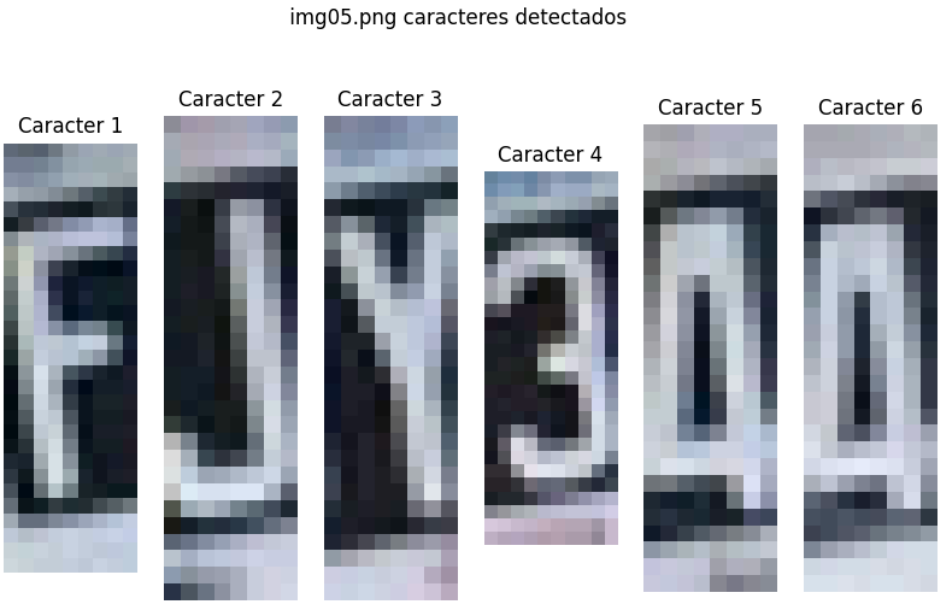
En el siguiente paso, identificamos los conjuntos alineados de tres caracteres. Para esto, definimos que los centros verticales deben estar dentro de un rango de 12 píxeles y la distancia entre los centros horizontales debe ser menor o igual que 30 píxeles.

TP2/img05.png trios

Habiendo ya identificado los caracteres de la patente, utilizamos la función cv2.boundingRect() para calcular el rectángulo de la imagen que contiene toda patente completa.

img05.png patente

Por último utilizamos la función cv2.findContours() para hallar los diferentes caracteres dentro de la patente y segmentarlos individualmente nuevamente con cv2.boundingRect().



En este ejercicio realizamos una detección y segmentación automática de las patentes de vehículos en imágenes. Para hacerlo, utilizamos técnicas de procesamiento de imágenes como convertir a blanco y negro, umbralización, encontrar las partes conectadas de la imagen y filtrar según tamaño, forma y posición. Así, logramos identificar la parte de la imagen donde está la patente. Después, buscamos los caracteres dentro de esa patente y los separamos uno por uno usando contornos y rectángulos.

Encontramos problemas debido a que las diferentes imágenes tenían diferentes condiciones en términos de iluminación, colores de los autos, ángulo de la foto, daños en la patente. Esto hizo que en algunos casos encontremos correctamente la patente con todos los caracteres y en otros la patente fue recortada incorrectamente y no todos los caracteres fueron detectados.