



Universidad Nacional de Rosario  
Facultad de Ciencias Exactas, Ingeniería y Agrimensura  
Tecnicatura Universitaria en Inteligencia Artificial  
Procesamiento de Imágenes I - IA 4.4

## TRABAJO PRÁCTICO N°2

GRUPO 6

### INTEGRANTES:

Cima, Nancy Lucía  
Longo, Gonzalo  
Sumiacher, Julia



# Índice

<b>Índice</b>	<b>1</b>
<b>Problema 1</b>	<b>2</b>
Descripción y Enunciado	2
Desarrollo	3
1. Preprocesamiento de la Imagen	3
2. Segmentación y Clasificación de Monedas	4
3. Segmentación y Clasificación de Dados	7
Conclusión	13
<b>Problema 2</b>	<b>14</b>
Descripción y Enunciado	14
Desarrollo	15
Conclusión	22



## Problema 1

### Descripción y Enunciado

El objetivo del primer problema es generar un programa que detecte y clasifique entre monedas y dados. Además, el programa debe realizar un conteo automático de monedas según su denominación y determinar la suma de los valores mostrados en los dados.

En la Figura 1 se muestra la imagen del archivo monedas.jpg que es la imagen de entrada del programa. Esta consiste en un fondo de intensidad no uniforme sobre el cual se hallan dados y monedas de distinto valor y tamaño.



Figura 1: Imagen de entrada (monedas.jpg)

Se debe elaborar un algoritmo capaz de procesar dicha imagen y resolver los siguientes puntos sobre ella:

- A. Procesar la imagen de manera de segmentar las monedas y los dados de manera automática.
- B. Clasificar los distintos tipos de monedas y realizar un conteo automático.



C. Determinar el número del valor que representa la cara superior de cada dado y realizar un conteo automático.

En cada punto, el script elaborado debe informar y mostrar los resultados en cada una de las etapas de procesamiento.

A continuación, se detalla el desarrollo y las etapas implementadas.

## Desarrollo

Dado que para este problema debemos tener en cuenta distintas cosas, decidimos crear funciones para modularizar nuestro código. Hablaremos un poco de cada una y de cada etapa de desarrollo.

Además, es importante considerar las librerías de Python usadas. En este caso necesitamos de Numpy, Matplotlib.pyplot y OpenCV.

### 1. Preprocesamiento de la Imagen

El primer paso consistió en preparar la imagen para su análisis mediante técnicas de preprocesamiento. En particular, la imagen original fue convertida a escala de grises utilizando cv2.cvtColor(), facilitando la aplicación de filtros y métodos de segmentación posteriores. Luego, se aplicó un filtro de mediana (cv2.medianBlur()) con un tamaño de kernel de 7 píxeles para eliminar posibles ruidos y suavizar la imagen sin perder bordes importantes.



Figura 2: Imagen preprocesada



Una vez hecho esto, podemos empezar a detectar y clasificar entre monedas y dados. Para esto creamos dos funciones llamadas “monedas” y “dados” de las que hablaremos en más detalle a continuación.

## 2. Segmentación y Clasificación de Monedas

Para este paso, creamos la función “monedas” que recibe una imagen de monedas y dados (la imagen original y aplica el preprocesamiento internamente), segmenta las mismas, calcula e informa el total y cuántas monedas de cada tipo hay. Devuelve la máscara de la segmentación y una imagen con la identificación de cada moneda. Se puede especificar la visualización del informe y el procesamiento de la imagen por separado, utilizando flags.

La identificación de las monedas se realizó utilizando el método de detección de círculos de Hough (`cv2.HoughCircles()`), permitiendo localizar las monedas según sus contornos circulares. Aquí se configuran parámetros importantes, como el rango de radios (`minRadius` y `maxRadius`) y los umbrales para la detección de bordes (`param1`, umbral para la detección de bordes (`Canny`) y acumuladores (`param2`, umbral para el acumulador de la transformada de Hough). Esto permite identificar los círculos correspondientes a las monedas.

Según el radio de cada círculo detectado, las monedas fueron clasificadas en tres categorías (10 centavos, 50 centavos y 1 peso).

Para esto primero creamos varias imágenes auxiliares:

- Una máscara binaria (`mask_monedas`) donde se dibujan los círculos detectados.
- Una imagen RGB (`tipos_monedas`) para clasificar las monedas con colores según su tipo.
- Una copia de la imagen original (`identificacion`) para marcar cada moneda con su tipo y dibujar un rectángulo alrededor de ellas.

También se inicializan contadores para las monedas de 10 centavos, 50 centavos y 1 peso.

Luego, pasamos al dibujo y clasificación de las monedas. En este paso, se recorren los círculos detectados y se dibuja un círculo blanco en la máscara binaria. Según el tamaño del radio, se clasifica la moneda según los siguientes criterios:



- Radios entre 170 y 190 corresponden a monedas de 50 centavos (color rojo).
- Radios entre 150 y 170 corresponden a monedas de 1 peso (color verde).
- Radios entre 120 y 150 corresponden a monedas de 10 centavos (color azul).

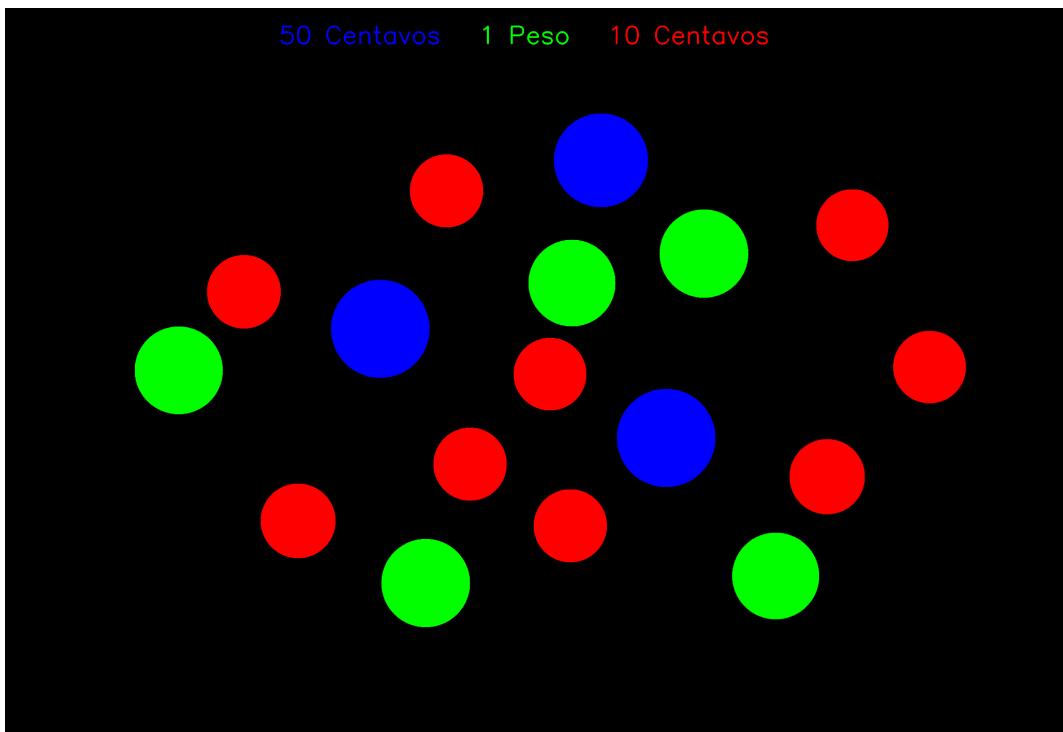


Figura 3: Imagen con monedas detectadas y clasificadas por colores sobre la máscara binaria

Además, se dibuja un rectángulo (bounding box) alrededor de cada moneda en la imagen preprocesada y se agrega un texto indicando su denominación para explicar qué tipo de moneda representa cada una.

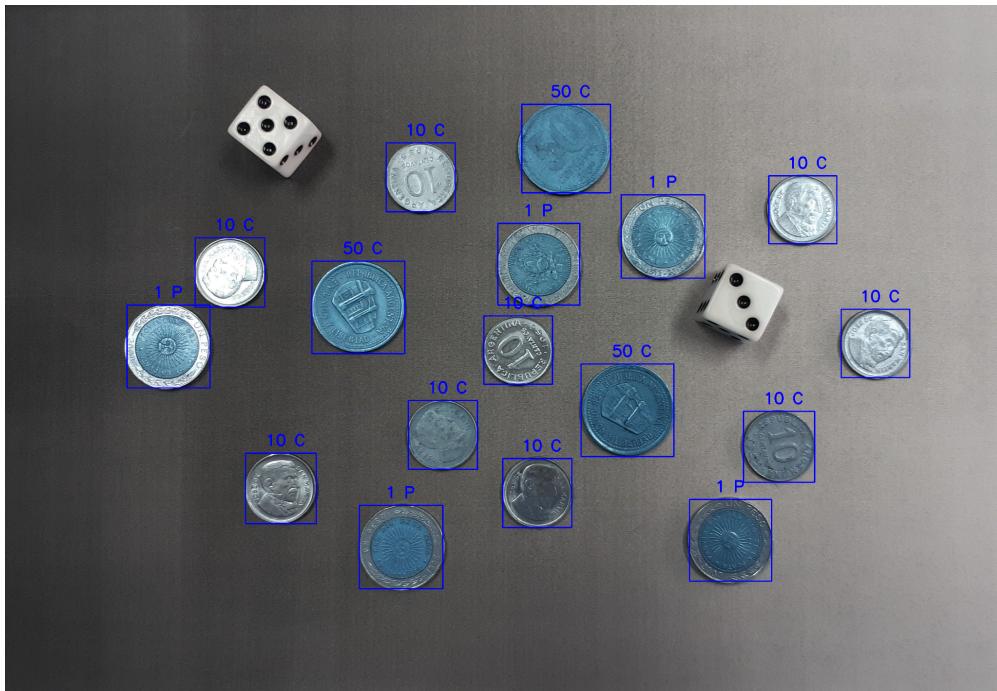


Figura 4: Imagen con monedas identificadas sobre la imagen preprocesada

Si está activada la opción de generar informe, se imprime en consola el total de monedas detectadas, el número de cada tipo y la cantidad de dinero total. Esto facilita analizar los resultados del procesamiento.

```
-----Informe-----  
  
Cantidad de monedas : 17  
Cantidad de monedas de 10 centavos : 9  
Cantidad de monedas de 50 centavos : 3  
Cantidad de monedas de 1 peso : 5  
Dinero total : $ 7.4
```

Figura 5: Imagen del informe obtenido para las monedas

Si la opción de visualización está activada, se muestran varias imágenes:

- La imagen original y la imagen en escala de grises con desenfoque.
- La máscara binaria con los círculos detectados.
- La imagen con las monedas clasificadas por colores.
- Una imagen con cada moneda identificada por texto y marcada con rectángulos.



Finalmente, el código devuelve dos imágenes: la máscara binaria con las monedas detectadas y la imagen con las monedas identificadas. Sumado a esto, contabilizó la cantidad de monedas de cada tipo y se reportaron los resultados. El algoritmo también calculó el total acumulado de las monedas y la cantidad de dinero total.

### 3. Segmentación y Clasificación de Dados

Para este paso, creamos otra función, la función “dados” que recibe una imagen de monedas y datos (la imagen original y aplica el preprocesamiento internamente) y una máscara binaria con las monedas detectadas en la imagen. Utiliza estos para segmentar los datos de la imagen y calcular e informar la cantidad de datos junto al número de la cara superior de cada dato. Luego, devuelve una máscara de la segmentación y una imagen con la identificación de cada dato y el resultado. Al igual que en la función “monedas”, se puede especificar la visualización del informe y el procesamiento de la imagen por separado.

A continuación, se detalla paso a paso cómo funciona esta función.

Primero, se utiliza la máscara generada en la etapa anterior para "borrar" las monedas de la imagen. Esta máscara es dilatada (`cv2.dilate()`) utilizando un kernel elíptico de tamaño 40x40. Esto asegura que las áreas correspondientes a las monedas sean suficientemente grandes para cubrirlas completamente. Luego, los valores de los píxeles correspondientes a las monedas en la imagen original se asignaron a cero, dejando solo los datos visibles.

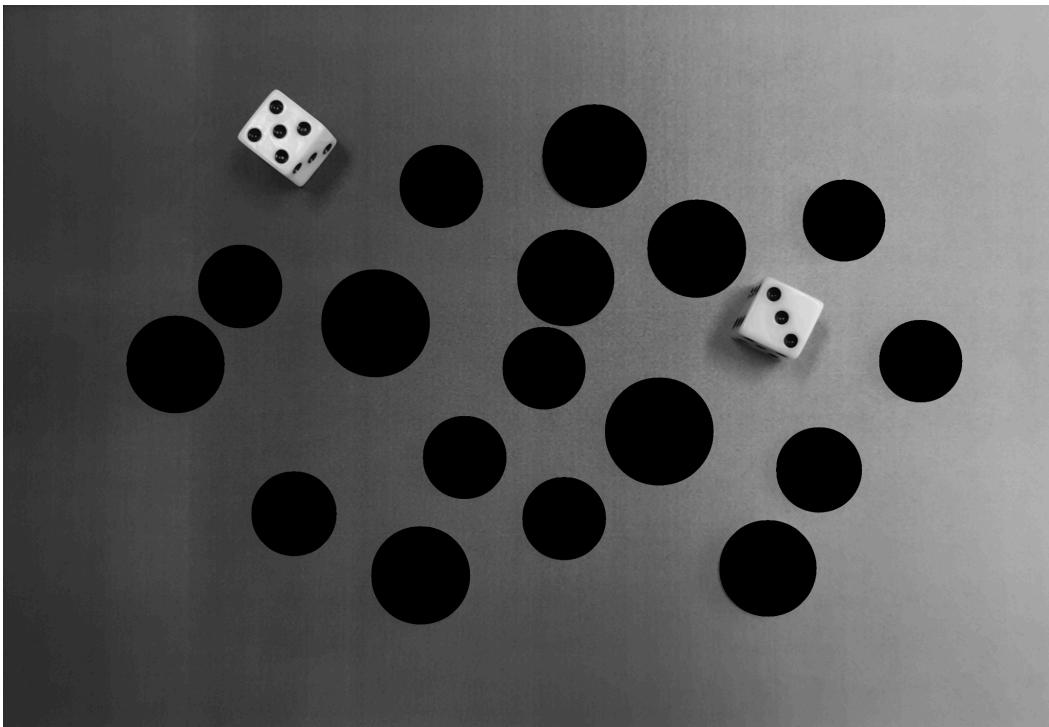


Figura 6: Imagen con las monedas removidas

Habiendo hecho esto, se aplica un umbral ( $\text{threshold} = 185$ ) para generar una máscara binaria (`mask_dados`) que segmenta las regiones más brillantes, correspondientes a los dados. Las áreas blancas en la máscara representan las caras superiores de los dados.

Luego, se utilizan componentes conectadas (`cv2.connectedComponentsWithStats()`) para identificar las áreas correspondientes a los dados. Los objetos más pequeños que un área mínima (4000 píxeles) fueron descartados como ruido.

Con la función `cv2.connectedComponentsWithStats`, se identifican las regiones conectadas en la máscara de los dados. Esto separa cada dado individual en la imagen, calculando estadísticas como área, centroides y coordenadas.

Para unificar posibles fragmentaciones en las caras de los dados, se aplicó una operación de cierre morfológico (`cv2.morphologyEx()`), utilizando un kernel elíptico más grande (80x80). Esto "rellena" los huecos dentro de las caras de los dados y mejora la segmentación.

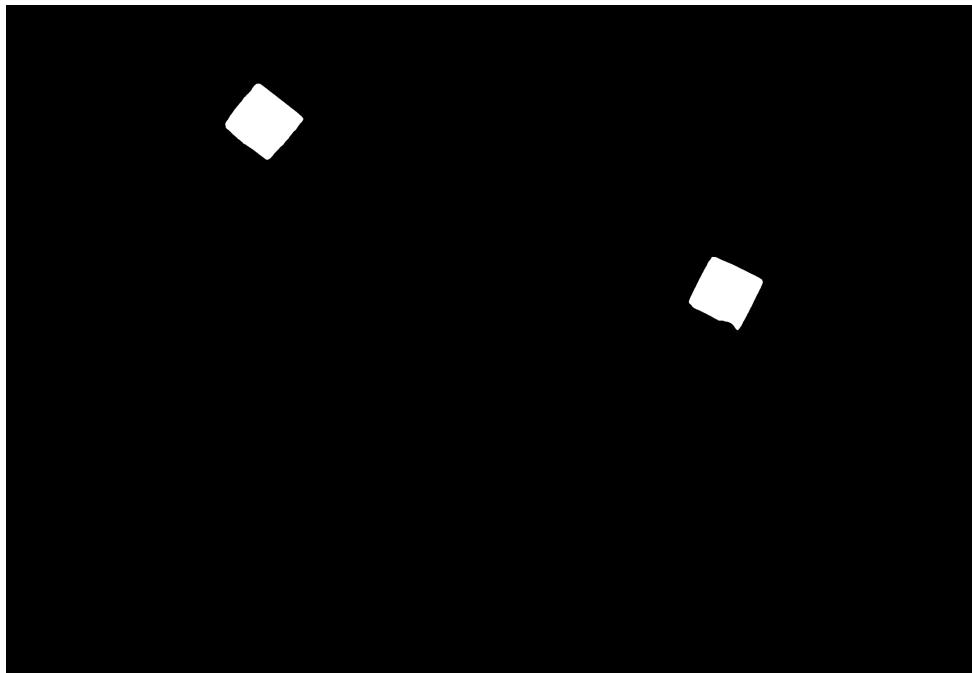


Figura 7: Imagen con datos segmentados y clausura aplicada

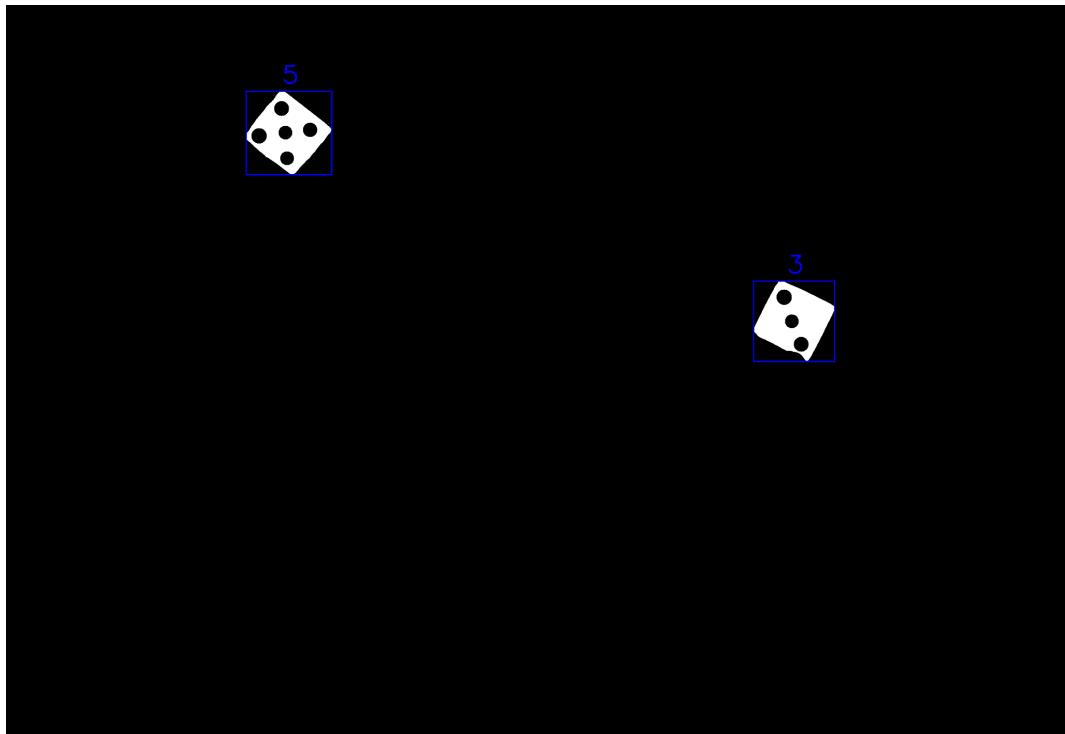
Ahora, pasamos a la detección de los puntos en los dados. Primero, se extrae la región de interés (ROI) de cada dado detectado en la imagen. Luego, los círculos representando los puntos de las caras de los dados fueron detectados nuevamente con cv2.HoughCircles(). Se suma el valor de las caras superiores de todos los dados para obtener el resultado final. Este resultado se puede mostrar como texto en la imagen procesada, dibujando un rectángulo alrededor de cada dado, y etiquetando el número en las imágenes img\_color e identificacion.

Si el parámetro views está activado, se visualizan las siguientes imágenes:

- La imagen original.
- La imagen en escala de grises con desenfoque.
- La máscara binaria con los dados detectados.
- La imagen con los dados clasificados.
- Una imagen con cada dado identificado por texto y marcada con rectángulos.



Universidad Nacional de Rosario  
Facultad de Ciencias Exactas, Ingeniería y Agrimensura  
Tecnicatura Universitaria en Inteligencia Artificial  
Procesamiento de Imágenes I - IA 4.4



Figuras 8 y 9: Imagen con los dados clasificados.



Si el parámetro informe está activado, se imprime en la consola:

- El número total de dados detectados.
- La suma de las caras superiores de todos los dados.

```
-----Informe-----  
Cantidad de dados : 2  
Suma de todos los dados : 8  
-----
```

Figura 10: Imagen del informe obtenido para los datos

La función devuelve dos imágenes:

- mask\_dados: La máscara binaria que representa las caras superiores de los dados.
- identificacion: La imagen de entrada con los dados identificados y etiquetados con sus valores.

En resumen, este código es útil para analizar imágenes con dados, segmentarlos correctamente, determinar sus caras superiores y calcular la suma total. Esto se logra mediante la combinación de operaciones morfológicas, segmentación y detección de círculos.



Universidad Nacional de Rosario  
Facultad de Ciencias Exactas, Ingeniería y Agrimensura  
Tecnicatura Universitaria en Inteligencia Artificial  
Procesamiento de Imágenes I - IA 4.4

Como resultado final de aplicar ambas funciones obtenemos:

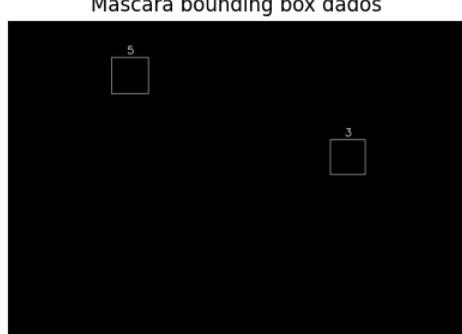
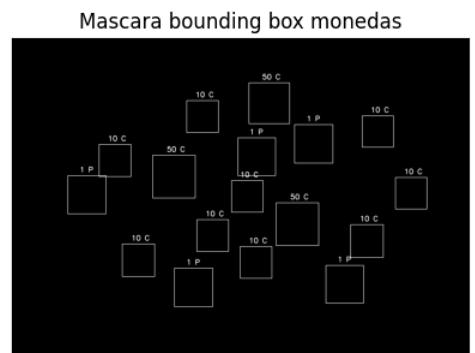


Figura 11: Imagen del resultado final



## Conclusión

El algoritmo desarrollado cumplió con éxito los objetivos planteados:

- Segmentó automáticamente las monedas y los dados presentes en la imagen.
- Clasificó las monedas según su denominación y realizó un conteo preciso.
- Determinó los valores de las caras superiores de los dados y calculó su suma total.

El uso de técnicas de procesamiento de imágenes como detección de bordes, operaciones morfológicas y componentes conectadas permitió obtener buenos resultados, incluso en presencia de ruido y superposiciones parciales en los objetos.

En futuras implementaciones, se podrían explorar técnicas basadas en aprendizaje automático para mejorar la robustez ante variaciones en la iluminación y otros factores ambientales para hacer que este código sea aplicable a imágenes similares con éxito.



## Problema 2

### Descripción y Enunciado

El objetivo del segundo problema es generar un algoritmo que detecte de patentes.

En la Figura 12 se muestra una de las doce imágenes (archivos img<id>.png), las cuales representan la vista anterior o posterior de diversos vehículos. En cada una de ellas se puede visualizar las correspondientes patentes.

Se debe elaborar un algoritmo capaz de procesar cada una de estas imágenes y resolver los siguientes puntos en cada uno de ellas:

- A. Detectar automáticamente la placa patente y segmentar la misma.
- B. Implementar un algoritmo de procesamiento que segmente los caracteres de la placa patente detectada en el punto anterior.

Informar las distintas etapas de procesamiento y mostrar los resultados de cada etapa para ambos puntos.



Figura 12 : Ejemplo de imagen de entrada (imagen05.png)

A continuación, se detalla cómo se resuelve este problema.



## Desarrollo

Lo primero que hacemos es cargar y visualizar las imágenes de entrada. Para esto, las 12 imágenes de vehículos se cargan y se almacenan en una lista. Cada imagen es leída con OpenCV (cv2.imread) y convertida al formato RGB (cv2.COLOR\_BGR2RGB). Las imágenes se muestran utilizando la función personalizada 'subplot12', que organiza las imágenes en una cuadrícula de 3x4, asignando un título descriptivo a cada una.

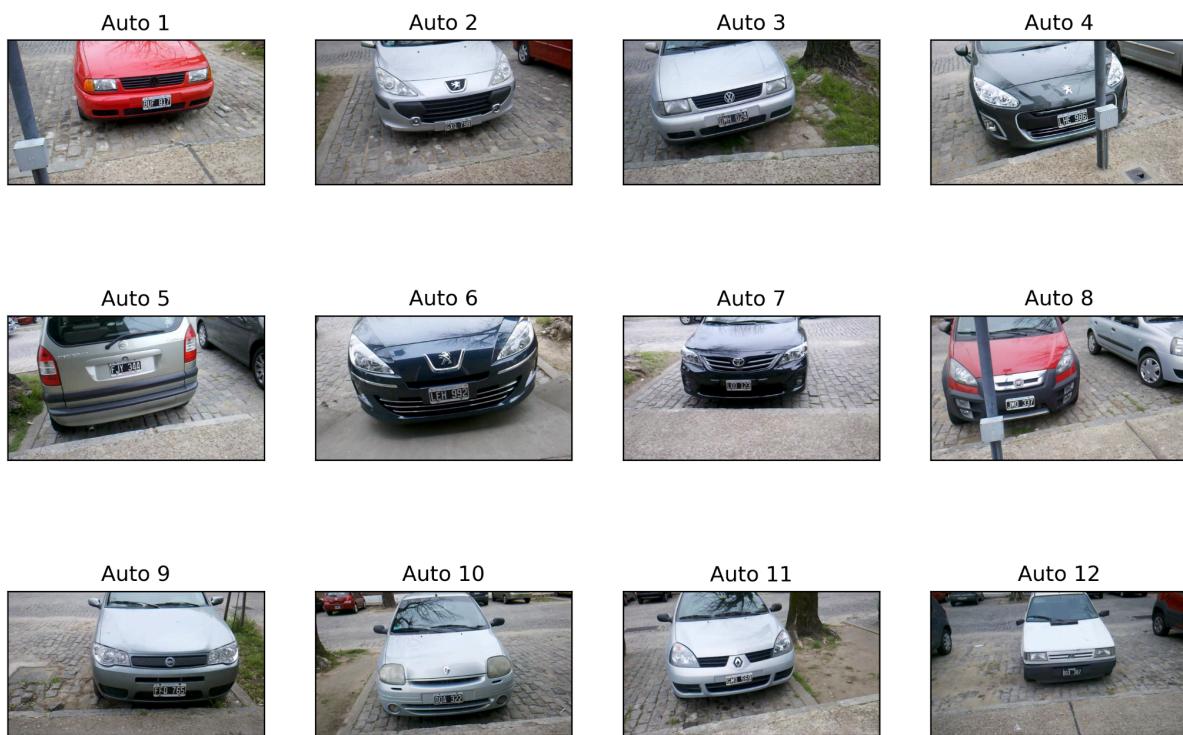


Figura 13: Imágenes de entrada

A continuación, las imágenes originales se convierten a escala de grises utilizando cv2.cvtColor con el argumento cv2.COLOR\_RGB2GRAY. Este paso reduce la complejidad de procesamiento al eliminar la información de color, centrándose únicamente en la intensidad de los píxeles. Nuevamente, se utilizan funciones personalizadas para mostrar las imágenes procesadas en escala de grises.



Universidad Nacional de Rosario  
Facultad de Ciencias Exactas, Ingeniería y Agrimensura  
Tecnicatura Universitaria en Inteligencia Artificial  
Procesamiento de Imágenes I - IA 4.4

Escala de grises Auto 1



Escala de grises Auto 2



Escala de grises Auto 3



Escala de grises Auto 4



Escala de grises Auto 5



Escala de grises Auto 6



Escala de grises Auto 7



Escala de grises Auto 8



Escala de grises Auto 9



Escala de grises Auto 10



Escala de grises Auto 11



Escala de grises Auto 12



Figura 14: Imágenes de entrada en escala de grises

Realizada esta etapa de preprocesamiento, pasaremos a recortar las áreas de interés (ROIs) de las imágenes. Para esto, definimos una función 'recortes' que recorta las regiones donde se encuentran las patentes en las imágenes. Las coordenadas de las regiones de interés (ROIs) se establecen manualmente para cada imagen.

Los recortes se generan sobre las imágenes en escala de grises, centrando el área de análisis en las patentes únicamente y facilitando la segmentación de caracteres en pasos posteriores.



Universidad Nacional de Rosario  
Facultad de Ciencias Exactas, Ingeniería y Agrimensura  
Tecnicatura Universitaria en Inteligencia Artificial  
Procesamiento de Imágenes I - IA 4.4



Figura 15: Patentes recortadas

Luego, aplicamos un filtro de realce de bordes (high-boost) para destacar los caracteres dentro de las patentes. Este filtro utiliza un kernel personalizado con parámetros ajustables, donde el valor  $A=2$  controla el nivel de realce. El filtro se aplica a las imágenes en escala de grises completas y a las regiones recortadas para comparar los resultados.

También, se aplica una umbralización binaria (`cv2.threshold`) para segmentar los caracteres de las patentes, separando píxeles claros y oscuros en función de un valor de umbral ( $TH1=145$ ). Este proceso genera imágenes binarias donde los caracteres se destacan como regiones blancas sobre un fondo negro.



Universidad Nacional de Rosario  
Facultad de Ciencias Exactas, Ingeniería y Agrimensura  
Tecnicatura Universitaria en Inteligencia Artificial  
Procesamiento de Imágenes I - IA 4.4

High boost Patente 1



High boost Patente 2



High boost Patente 3



High boost Patente 4



High boost Patente 5



High boost Patente 6



High boost Patente 7



High boost Patente 8



High boost Patente 9



High boost Patente 10



High boost Patente 11



High boost Patente 12



Figura 16: Patentes recortadas con filtro high-boost y umbralización binaria

Seguido de esto, calculamos la segmentación en componentes conectados mediante cv2.connectedComponentsWithStats. Mediante estos, aplicaremos filtros basados en:

- Área de los caracteres: Sólo se conservan componentes con áreas entre 18 y 90 píxeles.
- Relación de aspecto (RA): Los caracteres deben cumplir una proporción de altura y ancho entre 1.3 y 2.4.

Así, se eliminan elementos no relevantes (ruido) y se conservan los caracteres de las patentes. Obtenemos la siguiente imagen:



Universidad Nacional de Rosario  
Facultad de Ciencias Exactas, Ingeniería y Agrimensura  
Tecnicatura Universitaria en Inteligencia Artificial  
Procesamiento de Imágenes I - IA 4.4

Auto 1

BUF 817

Auto 2

GXA 790

Auto 3

WKA 024

Auto 4

LHE 985

Auto 5

FJY 344

Auto 6

LEH 992

Auto 7

LDD 123

Auto 8

JMD 337

Auto 9

FFD 765

Auto 10

DOD 322

Auto 11

GMO 569

Auto 12

DDX 387

Figura 17: Imágenes de los caracteres de cada patente

Luego, se realiza un análisis adicional para agrupar caracteres en conjuntos válidos de tres. Se verifican distancias horizontales y verticales entre caracteres consecutivos:

- Distancia horizontal máxima (dist\_max\_h): 13 píxeles.
- Distancia vertical máxima (dist\_max\_v): 5 píxeles.

En particular, la distancia horizontal máxima define cuánto pueden separarse horizontalmente dos caracteres consecutivos para considerarse parte del mismo grupo y la distancia vertical máxima establece cuánto pueden estar desplazados verticalmente dos caracteres consecutivos para formar parte del mismo grupo. Si la distancia vertical supera este umbral, los caracteres probablemente no pertenecen al mismo grupo, lo que evita errores en casos como texto en varias líneas.

Así, los caracteres que no forman parte de un grupo válido de tres componentes se eliminan.

Notamos que la imagen resultante conserva bien los caracteres para las imágenes 1 a 10, pero en el caso de las imágenes 11 y 12 vemos que elimina los grupos de caracteres numéricos. Esto puede deberse a que no cumplen los criterios antes nombrados, por ejemplo, puede que los caracteres de estas patentes estén



separados por una distancia mayor a la permitida, y aunque pertenezcan al mismo grupo, no se considerarán relacionados.



Figura 18: Imágenes resultantes de la agrupación de caracteres

Finalmente, el algoritmo procesa automáticamente las imágenes para detectar y segmentar las patentes de los vehículos, mostrando los resultados en cada etapa. Este flujo incluye desde la carga inicial de imágenes hasta la agrupación de caracteres para formar las patentes completas.

Para tratar estos casos, usaremos los parámetros definidos como punto de inicio e iremos variando hasta obtener los óptimos.

Primero, se copian las imágenes procesadas previamente, y para cada una, se calcula el número de componentes conectadas usando `cv2.connectedComponentsWithStats`. Esto genera estadísticas como coordenadas y tamaño de cada componente, que luego se ordenan horizontalmente (de izquierda a derecha) para facilitar el análisis.

Una vez ordenadas, se filtran los grupos de componentes que estén cercanos tanto en posición horizontal como vertical. Para ello, se establecen umbrales de distancia (`dist_max_h` y `dist_max_v`). El algoritmo busca tres componentes consecutivos que cumplan con estas condiciones de proximidad. Si no se encuentran suficientes



grupos válidos, se reduce gradualmente el umbral horizontal y se repite el proceso varias veces.

Finalmente, se eliminan las componentes que no pertenecen a los grupos seleccionados, quedando solo con las que cumplen las condiciones de cercanía, lo que ayuda a aislar los caracteres relevantes de la patentes.



Figura 19: Imágenes resultantes de la agrupación de caracteres iterativa

Finalmente, podemos detectar cada patente y sus caracteres sobre la imagen original, dibujando las bounding boxes.

Para eso usamos nuevamente cv2.connectedComponentsWithStats para identificar áreas conectadas, que en este paso son los caracteres de las patentes. Esto también nos ayuda a calcular las coordenadas de los recuadros (bounding boxes) para enmarcar los caracteres y la patente completa.



Universidad Nacional de Rosario  
Facultad de Ciencias Exactas, Ingeniería y Agrimensura  
Tecnicatura Universitaria en Inteligencia Artificial  
Procesamiento de Imágenes I - IA 4.4

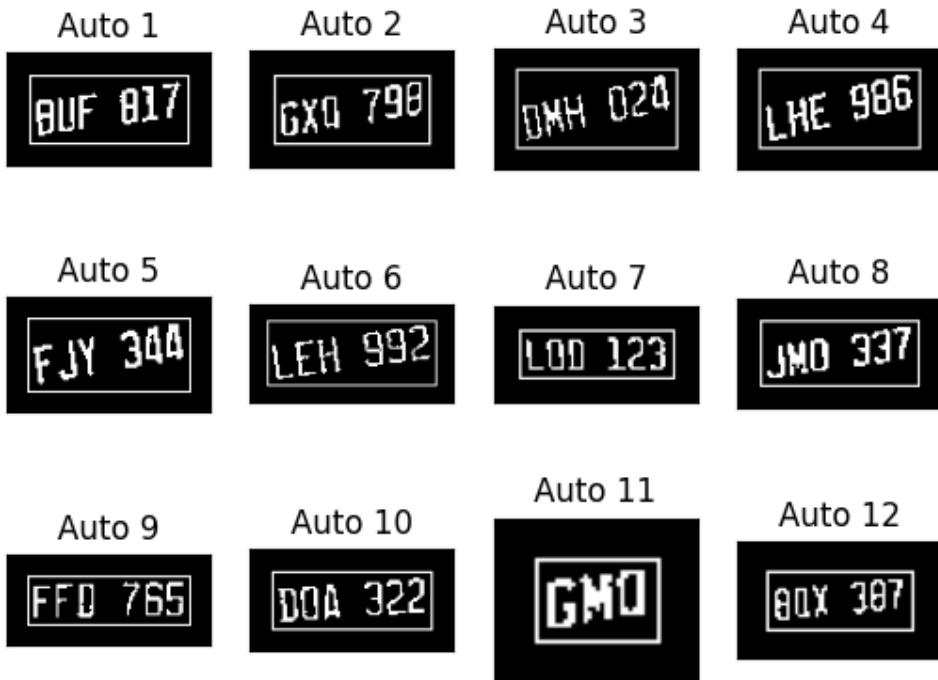


Figura 20: Imágenes de las patentes detectadas con sus recuadros.

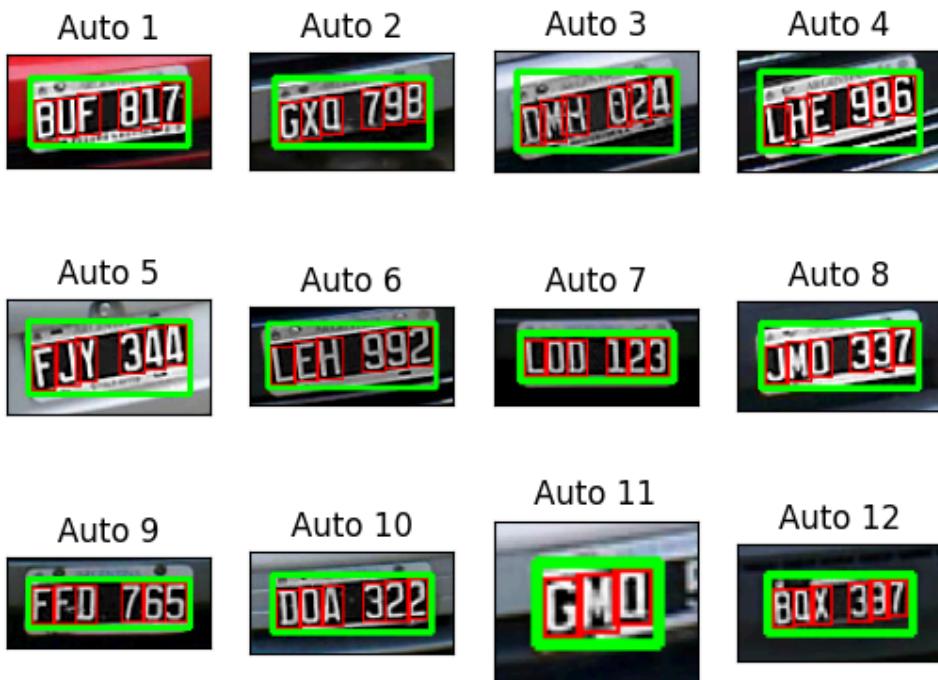


Figura 21: Imágenes de las patentes detectadas y sus caracteres.



## Conclusión

Este código logra su objetivo de detectar y segmentar patentes de automóviles utilizando técnicas de procesamiento de imágenes. A lo largo de los pasos, se demuestra una metodología efectiva para trabajar con este tipo de problemas, logrando buenos resultados al resaltar y aislar las áreas de interés. Sin embargo, el sistema presenta ciertas limitaciones en cuanto a su capacidad de generalización, ya que depende de parámetros ajustados manualmente que podrían no ser aplicables a otros conjuntos de datos o a imágenes de patentes que no cumplan con ciertas características contempladas al ajustar los parámetros. Por lo que se creó un algoritmo iterativo para definir los parámetros a utilizar. A pesar de esto, creemos que el trabajo realizado constituye una buena base para futuros desarrollos que podrían integrar algoritmos de aprendizaje automático o técnicas más avanzadas de visión por computadora, mejorando tanto la precisión como la adaptabilidad del sistema.