



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 - Año 2024 - 2º Semestre

Alumnos:

Antuña, Franco A-4637/1

Gallardo, Jonatan G-5970/6

Orazi, Roberto O-1815/5

Problema 1 - Detección y clasificación de monedas y datos.....	3
Descripción del problema.....	4
Implementación.....	4
Resultado.....	8
Problema 2 - Detección de patentes.....	10
Descripción del problema.....	11
Implementación.....	11
Resultados.....	17
Conclusión Final.....	18

Problema 1 - Detección y clasificación de monedas y dados

En la figura 1 se muestra la imagen del archivo *monedas.jpg*, la cual consiste en un fondo de intensidad no uniforme sobre el cual se hallan dados y monedas de distinto valor y tamaño.

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.

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



Figura 1: Imagen con monedas y dados del archivo *monedas.jpg*.

Descripción del problema

El objetivo principal del proyecto es desarrollar un algoritmo capaz de procesar una imagen denominada *monedas.jpg*, la cual contiene monedas de distintos valores y tamaños, así como dados con caras visibles en un fondo de intensidad no uniforme. Se busca segmentar automáticamente estos objetos, clasificar las monedas según su valor nominal, determinar el número visible en la cara superior de cada dado y realizar un conteo total. Para cada paso, el script debe generar resultados visuales y numéricos.

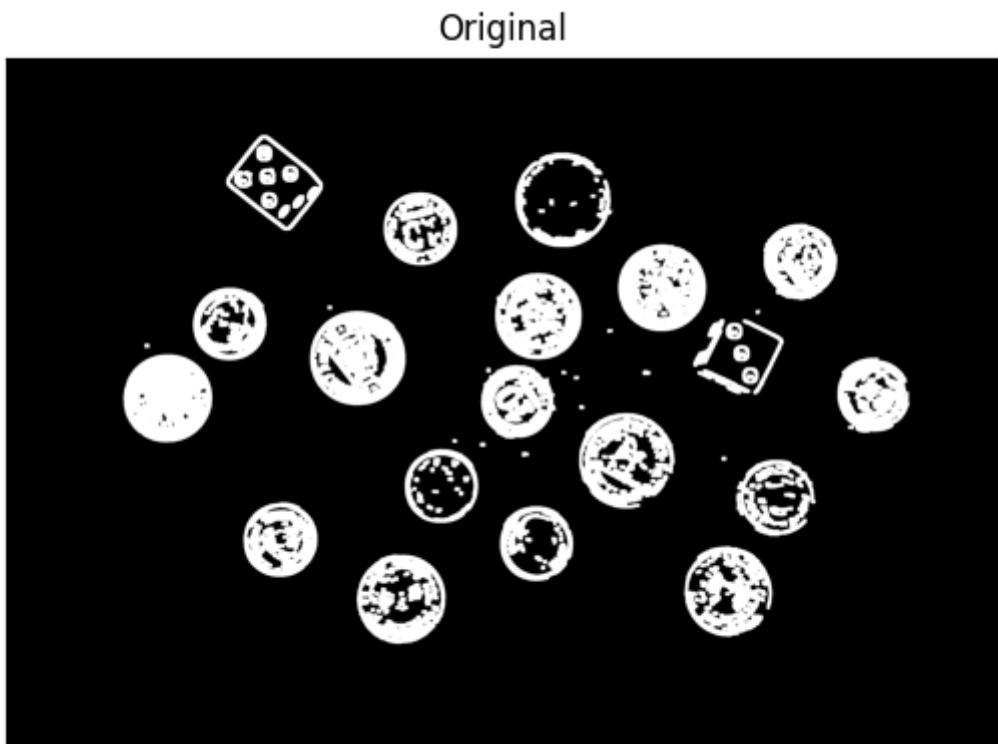
Implementación

La implementación del proyecto se llevó a cabo mediante un conjunto de etapas diseñadas para procesar y resaltar las características relevantes de la imagen original. En primer lugar, se adquirió la imagen de entrada y se preparó adecuadamente mediante su conversión a escala de grises, lo que permitió simplificar el análisis eliminando la información de color. Posteriormente, se aplicó un filtro Gaussiano para reducir el ruido presente en la imagen, mejorando así la definición de los bordes en las etapas posteriores.

Para identificar los contornos principales, se utilizó el operador *Canny*, el cual generó un mapa inicial de bordes basado en las diferencias de intensidad. Este mapa fue refinado mediante una operación de dilatación que amplió los bordes detectados, haciéndolos más prominentes y continuos. Finalmente, se llevó a cabo un cierre morfológico que permite llenar pequeñas discontinuidades en los bordes, logrando una representación más uniforme y preparada para su análisis.

La figura 1 muestra el resultado intermedio obtenido tras aplicar la operación de dilatación sobre la imagen procesada. En esta etapa, los bordes detectados previamente mediante el operador *Canny* se amplían utilizando un kernel de tamaño 13 x 13. Esto permite resaltar las áreas delimitadas por los contornos, haciéndolas más prominentes y uniformes. La dilatación es especialmente útil para mejorar la continuidad de los bordes y facilitar etapas posteriores de procesamiento, como la segmentación o el análisis de formas.

Figura 1. Resultado intermedio después de la dilatación aplicada a la imagen original.



Para reducir el ruido y preservar las características esenciales de los objetos, se implementa un filtro contraharmónico. Este filtro es eficaz para eliminar ruido de tipo sal sin afectar los bordes, lo cual resulta fundamental en imágenes donde los objetos tienen bordes suaves o finos. Una vez limpia la imagen, se procede a llenar huecos en las regiones segmentadas mediante un procedimiento de reconstrucción morfológica iterativa. Este paso garantiza que las monedas y dados queden completamente definidos, incluso si inicialmente presentan interrupciones en sus bordes.

El desarrollo del proyecto incluyó la creación de varias funciones personalizadas para abordar tareas específicas en el procesamiento de la imagen. A continuación, se presenta un resumen de las mismas:

La función ***imreconstruct*** implementa la reconstrucción morfológica a partir de un marcador y una máscara. Este proceso iterativo combina dilataciones sucesivas con intersecciones, permitiendo restaurar o reconstruir regiones de interés según las restricciones impuestas por la máscara. Es fundamental para operaciones como la restauración de formas específicas en imágenes binarias.

La función ***imfillhole*** se encarga de llenar los huecos en las regiones de una imagen binaria. Utilizando la reconstrucción morfológica como núcleo, esta función invierte la imagen y utiliza los bordes como marcadores para identificar y completar las áreas internas que corresponden a huecos. Al final, el resultado es invertido nuevamente para devolver la imagen con los huecos rellenos.

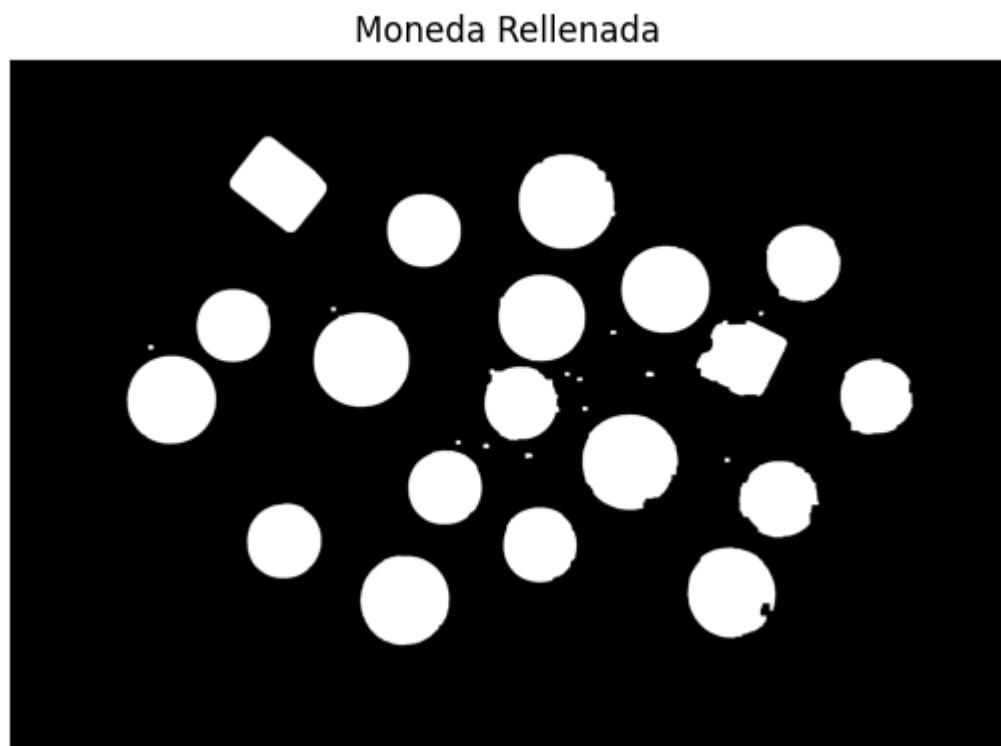
La función ***menor_factor_de_forma*** permite identificar el objeto con el menor factor de forma dentro de un conjunto de registros. Esto es útil en aplicaciones donde se analiza la forma de los objetos para clasificar o filtrar elementos específicos.

Finalmente, la función ***menor_area*** localiza el objeto con el área más pequeña dentro de un conjunto de objetos. Esto se utiliza para determinar características relevantes relacionadas con el tamaño de los elementos en la imagen, permitiendo discriminar entre diferentes regiones de interés.

Posteriormente, se aplicó un proceso de relleno de huecos en la imagen procesada. Para ello, se utilizó la función personalizada ***imfillhole***. Este proceso asegura que los objetos detectados no presenten discontinuidades internas, facilitando la identificación precisa de sus características.

La Figura 2 muestra el resultado de este paso, donde los huecos dentro de las formas detectadas han sido completamente llenados, destacando así las regiones sólidas correspondientes a las monedas y otros objetos en la imagen procesada.

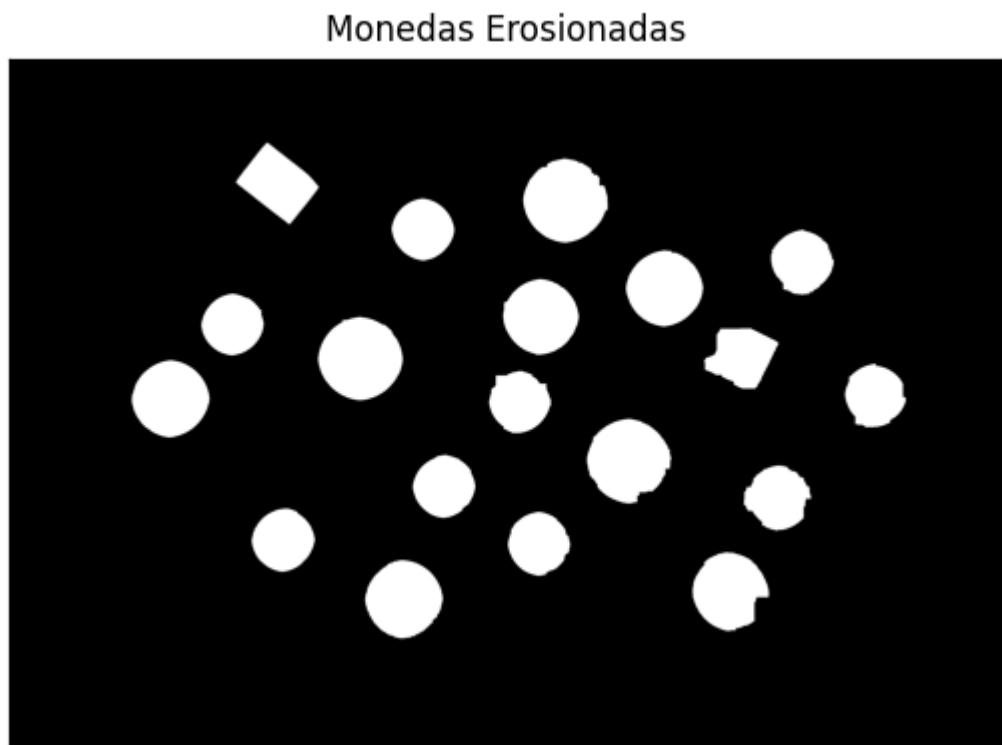
Figura 2. Imagen binaria tras el proceso de relleno de huecos aplicado a los bordes detectados.



A continuación, se realizó un proceso de erosión sobre la imagen resultante del relleno de huecos. Este paso tiene como objetivo reducir el tamaño de las regiones detectadas, eliminando posibles irregularidades en los bordes externos y refinando las

formas. Para ello, se utilizó un kernel de 41x41, lo que permite aplicar una erosión significativa, ajustada al tamaño de los objetos presentes en la imagen.

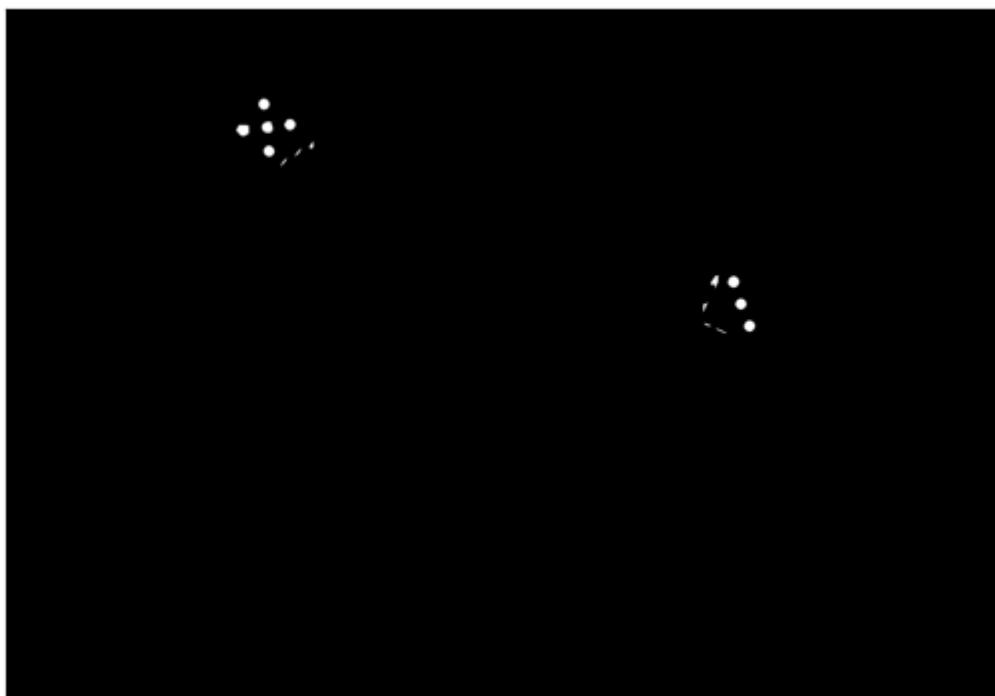
Figura 3. Imagen binaria tras el proceso de erosionado



Por otra parte, se aisló la región correspondiente a los datos mediante un procesamiento adicional. A estas regiones se les aplicó un refinamiento morfológico, que incluyó erosiones sucesivas y relleno de huecos. Esto permitió mejorar la precisión en la segmentación de los datos, eliminando detalles redundantes y asegurando la uniformidad de las regiones extraídas.

La Figura 4 muestra el resultado de esta etapa, donde los datos han sido erosionados y procesados para una mejor visualización y análisis.

Dados erosionados



En la última fase del procesamiento, se llevó a cabo la identificación y etiquetado de las monedas y los dados mediante un análisis más preciso de las regiones segmentadas. Para ello, se utilizaron las estadísticas generadas por la función **cv2.connectedComponentsWithStats** para los objetos identificados como dados. Se filtraron los componentes con un área superior a 1000 píxeles, lo que permitió enfocar la atención únicamente en las regiones relevantes.

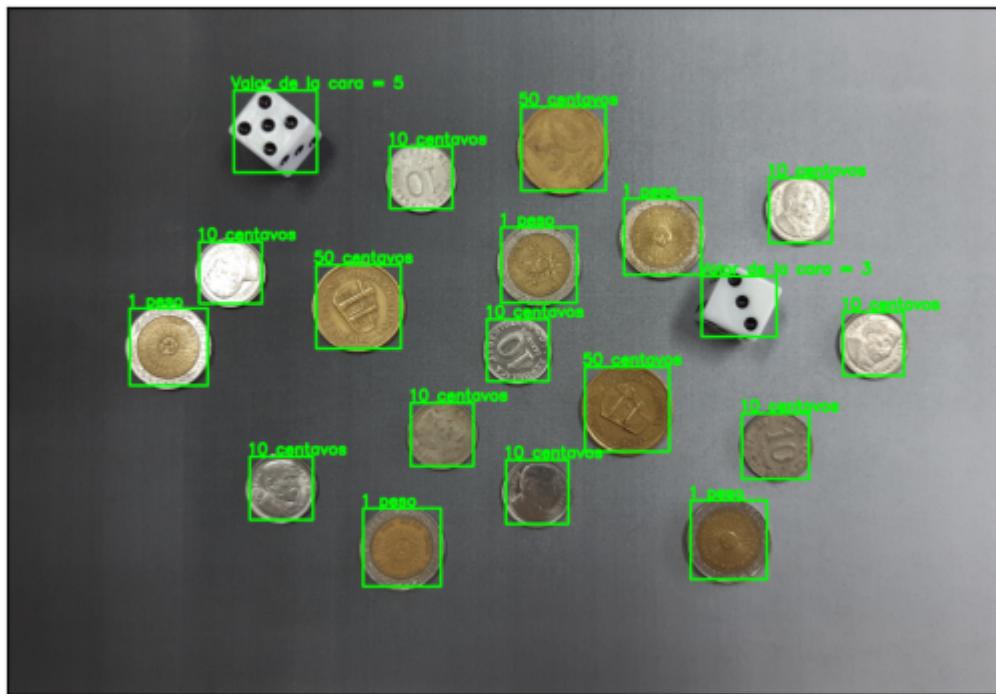
Además, se analizaron las caras de los dados, y a partir de la cantidad de componentes conectados en estas áreas, se asignaron valores a cada cara. Esta información se utilizó para etiquetar los dados con su valor correspondiente, que fue incorporado a la imagen.

Por otro lado, las monedas fueron etiquetadas y clasificadas en tres categorías según su tamaño. Se dibujan rectángulos alrededor de cada objeto, y se añadieron etiquetas con los valores correspondientes de las monedas, como "10 centavos", "1 peso" y "50 centavos". De manera similar, los dados también fueron etiquetados y rodeados por rectángulos, mostrando el valor de la cara detectada en cada uno.

Resultado

La figura 5 muestra la imagen final, donde tanto las monedas como los dados han sido etiquetados y destacados mediante rectángulos, con las correspondientes etiquetas visualizadas sobre cada objeto.

Dados y monedas etiquetados por valor



Se logró implementar un sistema eficaz para la detección, clasificación y etiquetado de monedas y dados a partir de imágenes utilizando técnicas de procesamiento de imágenes. El proceso comenzó con la segmentación de las regiones de interés mediante la conversión a escala de grises, el filtrado, y la detección de bordes. Posteriormente, se aplicaron operaciones morfológicas como la dilatación y la erosión para refinar la imagen y eliminar el ruido, mejorando la segmentación de los objetos.

El uso de componentes conectados permitió identificar y clasificar los objetos en la imagen según su área y forma, lo que facilitó la distinción entre las monedas y los dados. Las monedas fueron etiquetadas por valor, y se asignaron valores a las caras de los dados mediante el análisis de las componentes conectadas.

Se logró un etiquetado claro y preciso de los objetos detectados, destacando las monedas de diferentes denominaciones y los valores de los dados, lo que demuestra la efectividad de las técnicas de procesamiento de imágenes empleadas.

Problema 2 - Detección de patentes

En la figura 2 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. Informar las distintas etapas de procesamiento y mostrar los resultados de cada etapa
- B. Implementar un algoritmo de procesamiento que segmenta los caracteres de la placa patente detectada en el punto anterior. Informar las distintas etapas de procesamiento y mostrar los resultados de cada etapa.



Figura 2: Ejemplo de una de las imágenes (archivo *img05.png*) de la carpeta *Patentes*.

Descripción del problema

El propósito principal del script es procesar imágenes de vehículos para identificar y extraer posibles regiones que correspondan a las patentes, utilizando técnicas de procesamiento digital de imágenes y segmentación.

Implementación

El primer paso es recorrer cada imagen en un conjunto específico, representado por los nombres *img01.png*, *img02.png*, ..., *img12.png*. Cada imagen es leída usando la función *cv2.imread*, que permite leer el archivo en formato BGR (Blue, Green, Red), el formato utilizado comúnmente en OpenCV. Posteriormente, se convierte la imagen de BGR a RGB utilizando *cv2.cvtColor*, facilitando una mejor visualización en procesos posteriores, especialmente en herramientas gráficas como matplotlib.

Después de la conversión a RGB, se convierte la imagen a escala de grises utilizando *cv2.cvtColor*, convirtiendo así la imagen en un solo canal (intensidad de brillo), simplificando el procesamiento y facilitando la detección de patrones y contornos. Esta transformación es fundamental para operaciones como la detección de contornos y áreas relevantes.

Escala de grises



El siguiente paso es ajustar dinámicamente un umbral para separar las características de interés en la imagen. Este proceso es iterativo y personalizado para cada imagen.

Inicialización del Umbral: Se inicia con un umbral de valor 49. La bandera bandera controla si el proceso continúa aumentando el umbral.

Iteración del umbral: Mientras la bandera sea True, el umbral se incrementa en 1. Este umbral es usado en la función *cv2.threshold* para convertir la imagen a una versión binaria.

binaria



Detección de Características: Al aplicar el umbral, la imagen binarizada (*thresh*) resalta las áreas que superan el umbral establecido. Los píxeles que superan el umbral son convertidos en blanco (255), mientras que los que no cumplen con la condición se convierten en negro (0).

Detección Final: Si el valor del umbral alcanza 250, la iteración se detiene para evitar un procesamiento excesivo. Este enfoque permite ajustar dinámicamente el umbral hasta encontrar un equilibrio adecuado para resaltar las características deseadas en cada imagen, como las posibles patentes o áreas de interés.

Después de aplicar un umbral binario a la imagen, se utilizan los componentes conectados para identificar regiones contiguas. Para ello, se emplea la función *cv2.connectedComponentsWithStats*.

Componentes conectadas



Luego, se filtran las componentes según varios criterios. En primer lugar, se seleccionan aquellas con un área comprendida entre 17 y 200 píxeles. Esto permite excluir componentes demasiado pequeños o grandes que podrían no representar regiones relevantes. A continuación, se calcula la relación de aspecto de cada componente, es decir, la proporción entre su ancho y alto. Las componentes que tienen una relación de aspecto entre 1.5 y 3.0 se consideran adecuadas, ya que esto ayuda a filtrar aquellas que no tienen una forma deseada o que no están alineadas con las características esperadas para la detección de patentes.

Finalmente, las componentes que cumplen con estos criterios se almacenan en una lista llamada *l_aspecto*. Esta lista representa las regiones potencialmente válidas para ser analizadas y utilizadas en el procesamiento de imágenes para la detección de patentes.

Supongamos que se identifican al menos 6 componentes válidos en la imagen. En este punto, cada componente se procesa individualmente para calcular sus centros.

Para cada componente en la lista *l_aspecto*, se extraen las coordenadas de la caja delimitadora (*x*, *y*, *w*, *h*). A partir de estas coordenadas, se calcula el centroide de cada componente como un par de coordenadas (*cx2*, *cy2*), que representan las coordenadas del centro en los ejes horizontal y vertical.

Estos centros son almacenados en la lista *centro_caja*, que contiene la información esencial sobre la ubicación de cada componente significativo dentro de la imagen.

Después de calcular los centros de las componentes y almacenarlos en *centro_caja*, se procede a evaluar las distancias entre los centros para agrupar las cajas que están próximas unas a otras. Para cada combinación de centros, se calcula la diferencia en las coordenadas *x* e *y*.

caracteres de la patente



Si la diferencia en *x* es mayor a 0 pero menor a 80 y la diferencia en *y* es mayor a 0 pero menor a 20, se agrupan las componentes en *cotas_finales*. Este proceso permite

reducir el número de componentes individuales y consolidar aquellas que tienen proximidad espacial.

Una vez agrupadas las componentes según su proximidad, se verifica si el número de componentes agrupados está dentro del rango aceptable, generalmente de 6 a 8 componentes. Si se cumple esta condición, se detiene la iteración y se procede con la detección final.

Después de agrupar las componentes detectadas, se procede a dibujar las cajas alrededor de las áreas seleccionadas. Para cada componente, se calcula y actualiza las coordenadas de las esquinas superiores e inferiores (*esq_x_menor*, *esq_y_menor*, *esq_x_mayor*, *esq_y_mayor*) que delimitan la caja.

Estas coordenadas permiten definir un rectángulo alrededor de cada componente agrupado. Posteriormente, se dibuja este rectángulo en la copia de la imagen original (*copia_foto_original*) utilizando el color verde y un grosor de línea definido.

Imagen con los caracteres detectados



Finalmente, las coordenadas de las esquinas se actualizan para reflejar los valores mínimos y máximos encontrados en todas las cajas, permitiendo así definir un cuadro delimitador general para todas las áreas detectadas.

Después de dibujar las cajas delimitadoras para cada componente detectado, se dibuja un rectángulo adicional en la imagen procesada para representar la patente completa. Este rectángulo rojo se crea en base a las coordenadas mínimas (*esq_x_menor*, *esq_y_menor*) y máximas (*esq_x_mayor*, *esq_y_mayor*) obtenidas de las cajas individuales.

imagen con patente detectada



Luego, la imagen con los rectángulos correspondientes se agrega a una lista llamada imágenes, que contiene todas las imágenes procesadas.

Finalmente, todas las imágenes procesadas se visualizan en un conjunto de subplots usando matplotlib. Cada imagen se muestra junto con su respectiva caja delimitadora, y los ejes se ocultan para enfocar únicamente en las imágenes. Además, se ajusta el espaciado entre los subplots para una mejor visualización.

Resultados



Conclusión Final

El algoritmo desarrollado para la detección y segmentación de patentes vehiculares ha demostrado ser eficaz en la mayoría de las imágenes procesadas. A pesar de enfrentar ciertos desafíos técnicos, como la sensibilidad a características específicas de las imágenes (color del vehículo, iluminación y contraste) y la complejidad en la etapa final de agrupamiento, los resultados obtenidos validan el enfoque implementado.

En todos los casos, el sistema logra:

- Identificar componentes relevantes de las patentes en la mayoría de las imágenes, filtrando adecuadamente elementos no deseados.
- Detectar y reconstruir correctamente las patentes completas o, en su defecto, los caracteres que las componen en escenarios más complejos.
- Funcionamiento robusto bajo diversas configuraciones, ofreciendo un buen equilibrio entre precisión y eficiencia.

Una mejora significativa podría ser la implementación de un proceso más robusto para verificar la detección correcta de los caracteres. Un enfoque adaptativo ajustaría el sistema a las complejidades específicas de cada imagen, como diferencias en el color del vehículo, variaciones en brillo, contraste y otros factores visuales, asegurando una mayor precisión y adaptabilidad en diversas condiciones.