Universidad Nacional de Rosario
Facultad de Ciencias Exactas, Ingeniería y Agrimensura

IA4.4 Procesamiento de Imágenes

Trabajo Práctico N°2

Tecnicatura Universitaria en Inteligencia Artificial

Integrantes (Grupo 16):

Alsop, Agustín (A-4651/7)

Asad, Gonzalo (A-4595/1)

Castells, Sergio (C-7334/2)

Hachen, Rocío (H-1184/3)

Docentes:

Gonzalo Sad

Julián Alvarez

Juan Manuel Calle

Fecha: 25/11/2024

1. INTRODUCCIÓN

En el presente trabajo se aplican diversas técnicas de procesamiento de imágenes estudiadas en clase, como detección de bordes, segmentación, filtrado y operaciones morfológicas. El objetivo es desarrollar un sistema capaz de detectar automáticamente ciertos elementos en las imágenes.

En el primer problema, se busca identificar y clasificar objetos, distinguiendo entre dados y monedas. Además, se debe determinar el número visible en los dados y el tipo de moneda correspondiente.

El segundo problema tiene como propósito detectar las placas patentes contenidas en imágenes de distintos vehículos y segmentar los caracteres presentes en las placas identificadas.

2. PROBLEMA 1 – DETECCIÓN Y CLASIFICACIÓN DE MONEDAS Y DADOS

2.1 DESCRIPCIÓN

El problema consiste en desarrollar un algoritmo en Python para procesar la imagen contenida en el archivo *monedas.jpg*, la cual presenta un fondo de intensidad no uniforme y objetos (dados y monedas) de diversos valores y tamaños (Figura 1).

Figura 1: Imagen original con monedas y dados.

El objetivo es implementar un algoritmo capaz de resolver los siguientes puntos:

A. Procesar la imagen para segmentar automáticamente las monedas y los dados.
B. Clasificar los diferentes tamaños de monedas y realizar un conteo automático.
C. Identificar el número visible en la cara superior de cada dado y contabilizarlos de forma automática.

2.2 RESOLUCIÓN

Para la resolución del problema, se busca generar contornos sobre los objetos, para luego clasificarlos entre monedas o dados a partir de su factor de forma. Para eso, se aplican los siguientes procesos:

2.2.1 Preprocesamiento

I. Se carga al entorno de trabajo la imagen monedas.jpg
II. Se convierte la imagen a escala de grises.

III. Se aplica un filtro *Gaussiano* con un tamaño de ventana de 11x11 para reducir el ruido presente en la imagen y mejorar la detección de bordes.

2.2.2 Detección de bordes

Se utiliza el algoritmo Canny para resaltar los bordes de los objetos en la imagen. De esta manera, se consiguen identificar los contornos iniciales que luego serán refinados en los pasos posteriores.

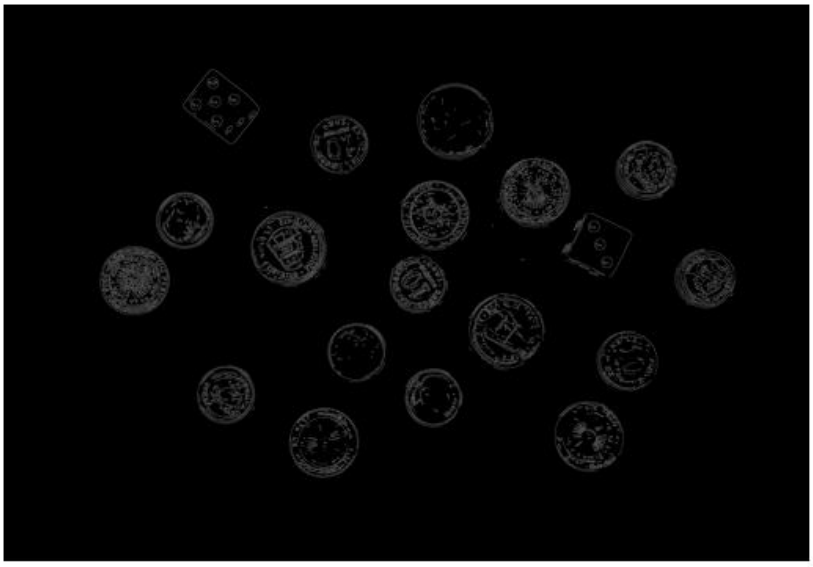

Figura 2: Bordes detectados mediante Canny.

2.2.3 Operaciones morfológicas

La operación anterior da como resultado algunos bordes discontinuos (que no se cierran sobre los objetos), lo que puede generar conflictos más adelante en el proceso de detección de contornos. Para solucionar el problema de discontinuidad y unir regiones cercanas, se aplican entonces dos operaciones morfológicas:

- *Dilatación:* Para expandir las regiones detectadas.
- *Clausura:* Para rellenar los espacios entre los bordes formando contornos completos y continuos.

En ambas operaciones, se utiliza un kernel de forma elíptica.

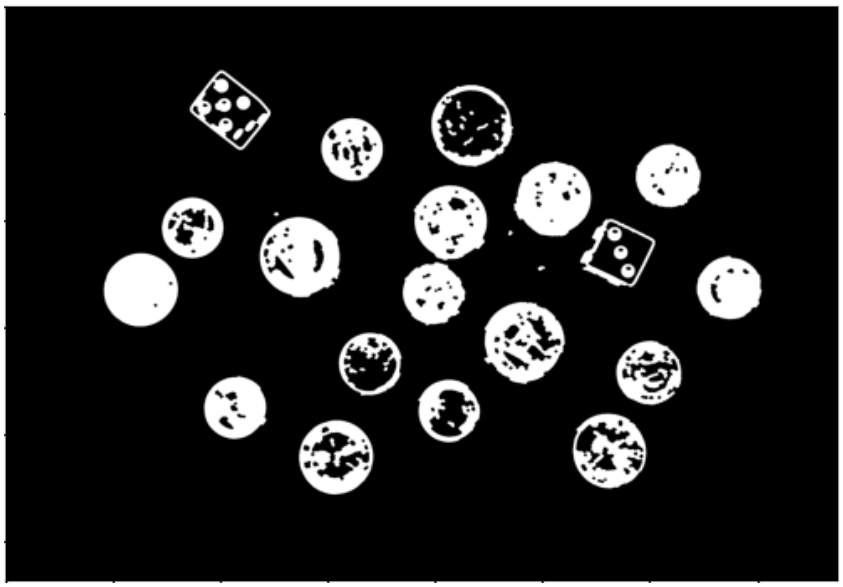

Figura 3: Bordes modificados con técnicas morfológicas.

2.2.4 Detección de contornos y relleno

Se obtienen nuevos contornos que, gracias a la transformación morfológica anterior, logran encapsular a los objetos en su totalidad y sin aperturas.

Figura 4: Bordes de objetos detectados correctamente.

Luego, estos son rellenados, obteniendo así una figura completa.

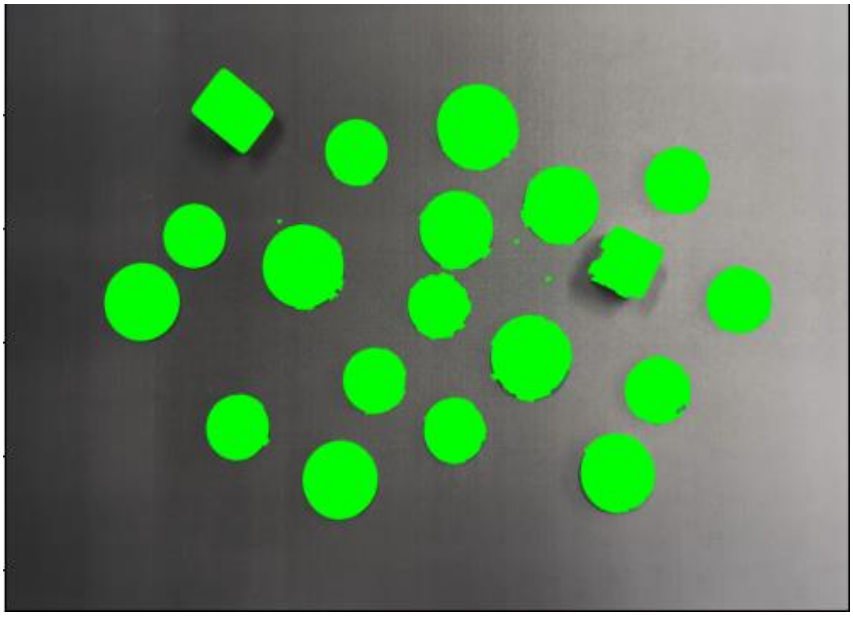

Figura 5: Objetos rellenados.

2.2.5 Detección y clasificación de objetos

A partir de los contornos detectados, se calculan métricas clave como el área, el perímetro, el factor de forma para evaluar la circularidad de los objetos y una aproximación poligonal para cada uno. Estos datos permiten clasificar los objetos según los criterios que se detallan a continuación:

- *Descartes*: Si el área o el perímetro son muy pequeños, se descarta el objeto.
- *Monedas*: Si la aproximación poligonal no presenta un polígono de 4 lados y su factor de forma indica una alta circularidad, los objetos se clasifican en tres categorías basados en su área:
  - Menores a 80000 px^2: Monedas de 10 centavos.
  - Entre 80000 px^2 y 100000 px^2: Monedas de 1 peso.
  - Mayores a 100000 px^2: Monedas de 50 centavos.
- *Dados*: Si la aproximación poligonal presenta un polígono de 4 lados, lo que indica una forma cuadrada o rectangular, el objeto es clasificado como dado. Para identificar el número mostrado en la cara superior, se aplican dos operaciones morfológicas ambas con kernel elíptico:
  - *Clausura*: Para eliminar los reflejos de luz sobre los círculos negros del número.
  - *Apertura*: Para separar los círculos negros del número y eliminar otros objetos indeseados.

Por último, se obtienen los contornos y se los contabiliza, determinando el número mostrado en la cara superior del dado.

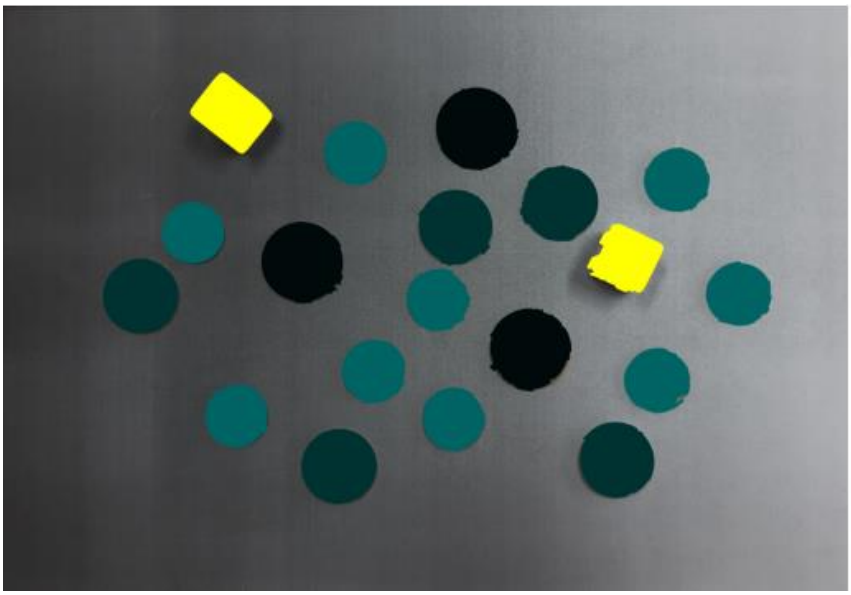

Figura 6: Objetos clasificados según su tipo y denominación.

Finalmente, sobre una copia de la imagen original, se dibujan cuadros delimitadores alrededor de cada objeto clasificado. Además, se superpone un texto descriptivo que indica el tipo de objeto identificado *(M: moneda, D: dado).*

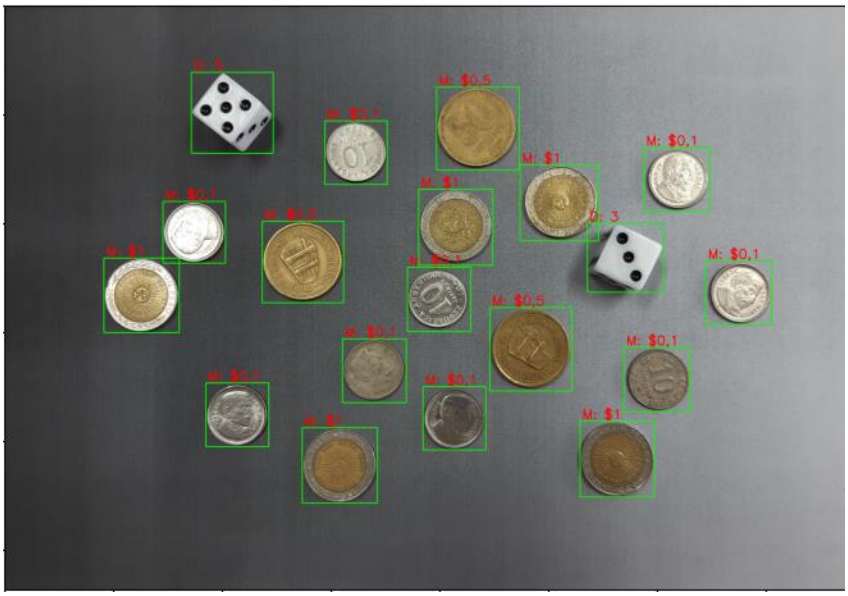

Figura 7: Resultado final.

2.2.6 Análisis del desarrollo

Inicialmente, el enfoque se centró en identificar los contornos de los objetos utilizando distintos canales de color de diversas codificaciones. Este método buscaba encontrar un canal lo suficientemente robusto como para manejar fondos con intensidades no uniformes y, a la vez, permitir la detección de los objetos sin incluir detalles internos. Sin embargo, dado que este enfoque no lograba generalizar de manera efectiva para monedas y dados simultáneamente, se decidió implementar la estrategia descrita anteriormente.

Durante el desarrollo del algoritmo actual, fue necesario realizar numerosas ejecuciones para ajustar cuidadosamente los parámetros de las funciones. La selección adecuada del tamaño de las matrices utilizadas en el filtrado Gaussiano y de los elementos estructurales resultó ser un aspecto clave para lograr un resultado exitoso en la etapa final del procesamiento.

2.3 CONCLUSIONES

El algoritmo consigue identificar exitosamente todas las monedas y dados en la imagen, además de clasificar los objetos de acuerdo a su denominación o valor respectivamente. A pesar de esto, no es lo suficientemente robusto como para reconocer monedas o dados en otras imágenes, ya que sus funciones fueron ajustadas específicamente para la imagen de este trabajo.

Como posibles mejoras, se podría investigar un nuevo método de clasificación de monedas que no sea dependiente de parametrizaciones de área. Una opción sería implementar el algoritmo *k-means*, permitiendo clasificar las monedas de manera más inteligente y autónoma, identificando patrones naturales en los datos sin necesidad de establecer umbrales específicos.

3. PROBLEMA 2 – DETECCIÓN DE PATENTES Y CARACTERES

3.1 DESCRIPCIÓN

El problema consiste en desarrollar un algoritmo en Python capaz de detectar automáticamente las placas patentes contenidas en imágenes de distintos vehículos, y segmentar los caracteres presentes en las placas identificadas.

Si bien el enunciado ofrece un orden en el cual realizar las detecciones (primero detectar la patente y luego segmentar sus caracteres) nuestra resolución opta por resolver la problemática al revés. Primero, se detectan y segmentan los caracteres y luego, utilizando sus coordenadas y conociendo las dimensiones de una patente Argentina, se encuadra la placa.

Las patentes, al presentar un diseño en blanco y negro, a menudo no contrastan lo suficiente con el vehículo, cuyo color puede variar ampliamente, lo que complica su identificación. No obstante, los caracteres de las patentes siempre son blancos sobre un fondo negro, lo que facilita su detección y diferenciación visual.

Nota: Este algoritmo está diseñado específicamente para trabajar con las patentes argentinas emitidas entre 1994 y 2016.

Figura 8: Ejemplo patente argentina.

3.2 RESOLUCIÓN

Para la resolución de este problema, se diseñó una función llamada `matDetection`, destinada a identificar y delimitar regiones en una imagen que correspondan a los caracteres de una patente. Esta detección se realiza aplicando criterios basados en el tamaño y la relación de aspecto de los elementos. Luego, con las coordenadas obtenidas, se delimita el área correspondiente a la placa de una patente.

La función toma los siguientes parámetros:

- `img(np.ndarray)` **:** Imagen a procesar (en escala de grises).
- `th_ini(int)`: Valor de umbral inicial para aplicar umbralado.
- `max_area(float)` **:** Área máxima aceptable para un componente conectado.
- `min_area(float)` **:** Área mínima aceptable para un componente conectado.

- `max_aspect_ratio(float)`: Relación de aspecto máxima permitida para los componentes conectados.
- `min_aspect_ratio(float)`: Relación de aspecto mínima permitida para los componentes conectados.
- `jump(int)`: Incremento en el valor de umbral tras cada iteración.

Retorna un `np.ndarray` correspondiente a la imagen procesada, donde se marcan las áreas detectadas que cumplen con los criterios establecidos y una variable de `status` que indica si el procesamiento fue realizado de forma exitosa.

3.2.1 Detección de caracteres de patente

El algoritmo comienza realizando una conversión de la imagen a escala de grises, para luego aplicarle un umbralado binario.

Figura 9: Imagen umbralada.

A continuación, se extraen las componentes conectadas de la imagen binarizada, lo que permite identificar las coordenadas, tamaños y áreas de cada componente presente en la imagen. Este análisis es fundamental para localizar las regiones que podrían corresponder a los caracteres de la patente.

Figura 10: Componentes conectadas.

Utilizando los parámetros definidos en la función, se filtran las componentes cuyas áreas no se encuentren dentro de los límites especificados. Estos límites han sido ajustados específicamente para identificar componentes que correspondan a los caracteres de las patentes, asegurando una detección precisa y consistente.

Figura 11: Filtrado por área.

Luego, se realiza un segundo filtrado para descartar elementos con áreas similares que no corresponden a caracteres de patentes. Este proceso utiliza la relación de aspecto de cada

componente y los parámetros establecidos en la función, eliminando aquellos cuya métrica no se encuentre dentro de los límites especificados.

A continuación, se realiza un conteo de los elementos resultantes del proceso de filtrado. Si esta cantidad es menor a 6, el algoritmo reajusta el umbral y repite el procesamiento. De lo contrario, se analiza la proximidad de las coordenadas *x* de los elementos mediante la función personalizada `slice_when`, que agrupa un iterable en sublistas basándose en una condición específica (en este caso, si la distancia entre dos coordenadas *x1* y *x2* supera los 30 píxeles). Este paso busca identificar un grupo de 6 elementos cercanos entre sí.

Si no se cumple esta condición, se reajusta el umbral y se repite el proceso. Si se obtiene un conjunto válido de 6 elementos, se evalúa la relación entre sus coordenadas *y*. Si la diferencia máxima entre las mismas es inferior a un valor predefinido, se concluye que los elementos corresponden a los caracteres de la patente. En caso contrario, se ajusta nuevamente el umbral y se repite el proceso hasta cumplir con las condiciones establecidas.

Si el valor del umbral excede 250 durante este proceso, el algoritmo considera que el procesamiento no fue exitoso y actualiza la variable `status` para reflejar esta situación.

Figura 12: Filtrado por factor de forma.

Finalmente, sobre una copia de la imagen original, se dibujan los cuadros delimitadores alrededor de cada carácter de la patente.

3.2.2 Detección de placa de patente

Habiendo detectado los caracteres de la patente, se procede a dibujar un cuadro delimitador adicional alrededor del conjunto de caracteres, utilizando un margen proporcional a su tamaño. Dado que las patentes tienen un tamaño estándar, este proceso permite encuadrar la placa en su totalidad.

De esta manera, se obtiene el resultado final en una copia de la imagen original, donde todos los elementos de interés están claramente resaltados.

Figura 13: Resultado final.

3.2.3 Análisis del desarrollo

El algoritmo debe ser capaz de identificar los caracteres de las patentes en imágenes con condiciones de iluminación y distancias variables respecto al vehículo. Fijar un único valor de umbral no garantiza resultados óptimos en todos los casos. Por ello, se optó por iterar sobre una gama de valores de umbral, aplicando validaciones en cada iteración para descartar aquellos que no producían los resultados esperados.

I. *Cantidad de caracteres encontrados*: si luego del filtrado por área y relación de aspecto no se obtienen al menos 6 componentes, el valor de umbral es descartado.
II. *Distancia entre componentes*: los caracteres se encuentran muy cercanos el uno del otro, tanto en el eje x como en el eje y. Se observa la distancia máxima entre componentes en ambos ejes. Si la distancia máxima supera el máximo permitido, se descarta el valor de umbral.

Si se superan las dos verificaciones, se considera que se han encontrado exitosamente los 6 caracteres de la patente.

Es importante aclarar que, en un principio, la iteración se realizaba sobre el rango completo de valores de umbral, avanzando con un incremento de 1 en cada paso. Una vez que se lograron resultados satisfactorios en todas las imágenes, se analizó el rango de umbrales que producían resultados exitosos. Con base en este análisis, se acotaron los valores de iteración, optimizando así la ejecución del algoritmo sin comprometer su efectividad.

3.3 CONCLUSIONES

El algoritmo logra identificar correctamente los caracteres y las placas de patentes de manera satisfactoria, funcionando eficazmente en todas las imágenes utilizadas. La iteración sobre diferentes valores de umbral y la evaluación de la cercanía de componentes son elementos clave para asegurar la identificación precisa en todas las imágenes, permitiendo filtrar los caracteres de forma correcta en cada caso. Gracias a estas estrategias, el algoritmo muestra un buen nivel de generalización al aplicarse sobre las 12 imágenes utilizadas en este trabajo.

Sin embargo, es importante señalar que la parametrización de factores como los límites de área o el factor de forma fue específicamente diseñada para este conjunto de imágenes. Por lo tanto, no se puede garantizar que el algoritmo funcione de igual manera con fotos tomadas desde otras distancias, ángulos o condiciones de iluminación diferentes.

Como posibles mejoras, se podría investigar una manera diferente de localizar la placa de patente sin depender de los caracteres. También, estudiar cómo permitir que la detección de caracteres se haga sobre imágenes de todo tipo, donde el tamaño y ángulo de las letras presenta una variación mayor al de las imágenes utilizadas en este trabajo.

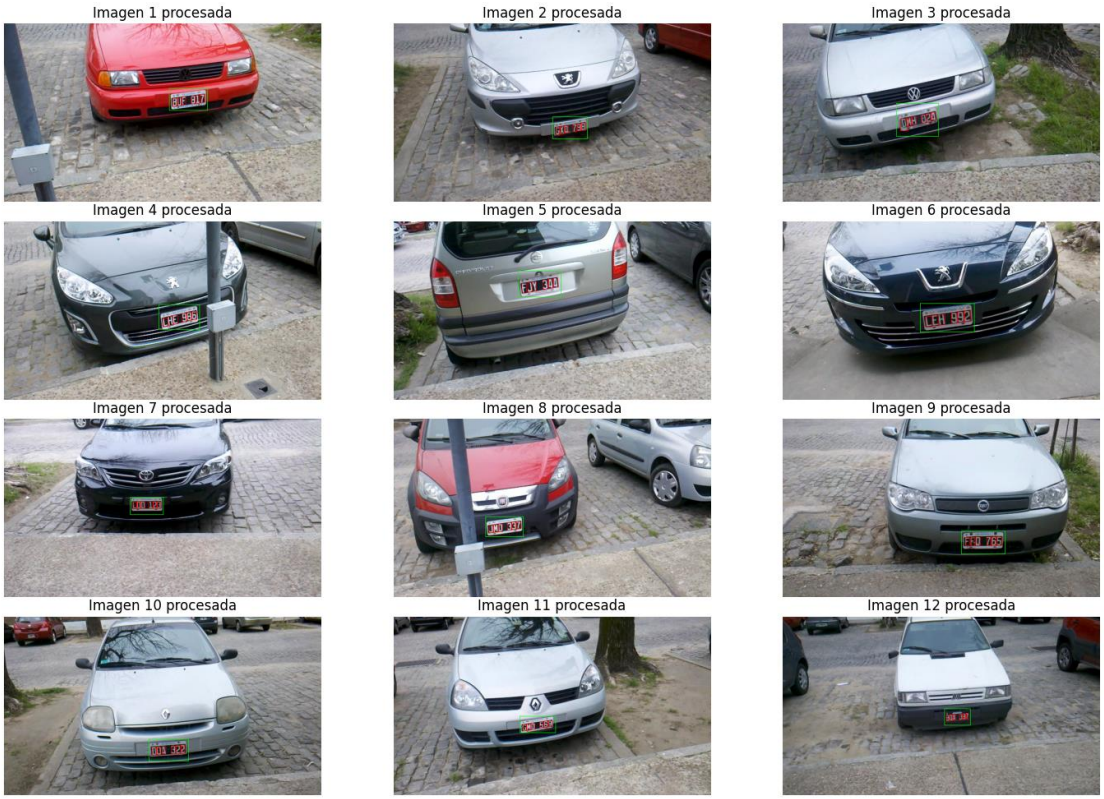

Figura 14: Resultados generales.