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

Tecnicatura Universitaria en Inteligencia Artificial

Asignatura: Procesamiento de Imágenes

TRABAJO PRÁCTICO Nº 2

Alumnos: Accurso, Agustín (A-4749/1)
Barbarroja, Federico (B-6714/8)
Cena, Lautaro (C-7565/9)

Equipo Docente: Gonzalo Sad
Juan Manuel Calle
Joaquín Allione

El presente Trabajo Práctico Nº 2 consta de dos problemas, que fueron resueltos en Python, utilizando Visual Studio Code para realizar los scripts y librerías vistas en clase, como OPENCV, NUMPY y MATPLOTLIB. Los problemas fueron analizados utilizando el pensamiento crítico, y resueltos mediante la aplicación de métodos vistos en clase, con ayuda de chatbots de IA para su implementación en ciertas situaciones, pero siempre manteniendo la comprensión total del código y priorizando la lógica y comprensión por sobre los resultados. Durante este informe se analizarán los resultados
obtenidos y se elaborarán conclusiones.

Problema 1 - Detección y Clasificación de Monedas y Dados

El algoritmo implementado en Python (OpenCV) resuelve la segmentación y clasificación superando la desigual luz del fondo mediante detección de bordes y análisis morfológico.

Segmentación Automática

En lugar de un umbralado global (que fallaba por la luz), se aplicó la siguiente técnica, cuyo resultado final podemos observar en la figura 1:

1. **Preprocesamiento**: Conversión a escala de grises y suavizado Gaussiano para reducir ruido.
2. **Bordes (Canny)**: Detección de bordes con umbrales 30 y 180.
3. **Morfología (Clausura)**: Se aplicó una dilatación (7 iteraciones) seguida de una erosión suave para fusionar los bordes y crear máscaras sólidas ("manchas") detectables.

4. **Extracción**: Uso de **findContours** para aislar los 19 objetos presentes.

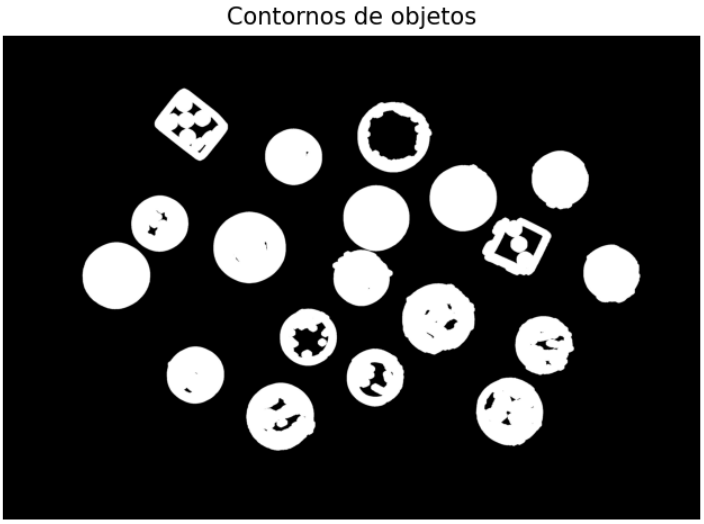


Figura 1.

Clasificación y Conteo de Monedas
Se diferenciaron los objetos midiendo su Circularidad:

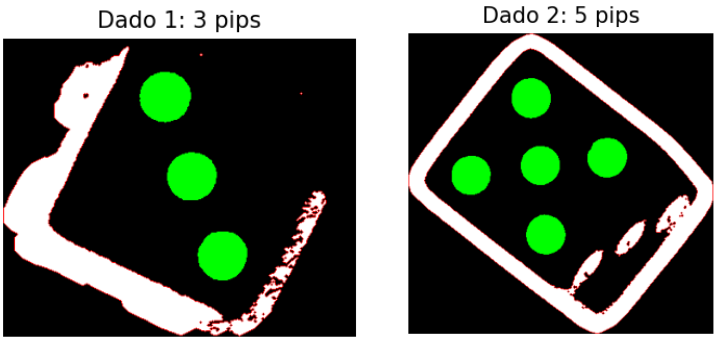
- Tipo: Objetos con circularidad > 0.80 se clasifican como monedas; el resto, como dados.
- Valor: Se utilizó el Área en píxeles para distinguir los valores, estableciendo cortes claros:
    - Área < 81k: 10 centavos.
    - Área entre 81k y 111k: 1 peso.
    - Área > 111k: 50 centavos.

- Podemos observar los resultados en la figura 2. (Azul=dados - verde=monedas)

Figura 2.

Lectura de Dados (observar resultados en figuras 6 y 7).
Para leer el valor de los dados, se extrajo la Región de Interés (ROI) de cada dado detectado:

1. Se aplicó un umbralado local inverso para detectar los puntos negros (pips).
2. Se filtraron los falsos positivos (bordes del dado) contando solo los contornos internos con una circularidad > 0.6.



Figuras 6 y 7.

Conclusión
El script final segmenta correctamente todos los objetos, ignora las variaciones de iluminación del fondo y etiqueta automáticamente el valor de las monedas y la suma de los dados, mostrando los resultados en la imagen de salida (figura 8).

Figura 8.

Problema 2 - Detección de patentes

Planteamiento del Problema Se debe implementar un algoritmo para el reconocimiento automático de patentes en imágenes de vehículos. La problemática se desglosa en dos tareas secuenciales:

1. **Detección de Placa:** Identificar y recortar la patente del resto de la carrocería y el entorno.
2. **Segmentación de Caracteres:** Procesar la patente recortada para separar los números y letras que la componen. Como requisito funcional, el sistema debe mostrar la eficacia de cada paso del proceso, mostrando las imágenes resultantes tras cada operación.

Desarrollo de metodologías implementadas:

1. Detección y Rectificación de la Placa: En esta primera instancia se busca aislar la patente del resto del vehículo, utilizando métodos de umbralado, filtrado, morfología y homografía.

Figura 9 - 'Pasos intermedios' :

- **Preprocesamiento (Sobel + Otsu):** Se observa cómo el filtro de Sobel vertical resalta las transiciones de contraste (caracteres y bordes laterales), mientras que la binarización de Otsu separa estas características del fondo.

- **Morfología (Candidatos):** Mediante una operación de clausura con un kernel rectangular horizontal, se fusionan los detalles verticales individuales en un único "bloque" sólido (mancha blanca). Esto permite que el algoritmo de búsqueda de contornos detecte la patente como un único objeto rectangular, ignorando el ruido circundante.

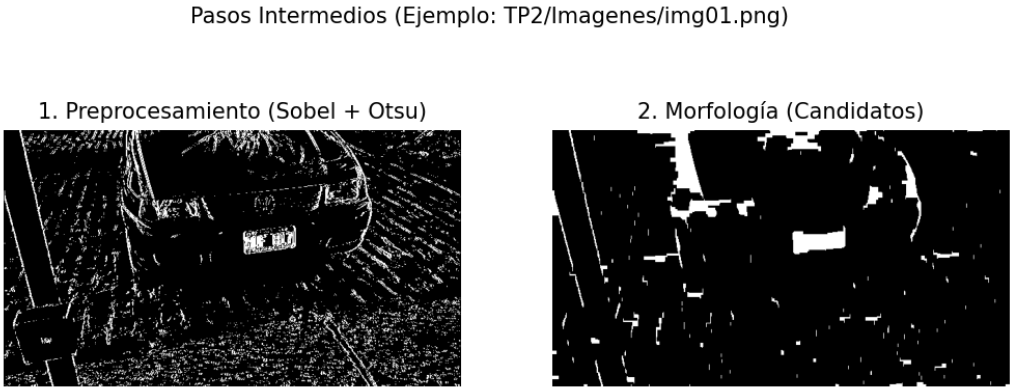


Figura 9.

Figura 10 – 'Patentes Rectificadas (Homografía)':
Una vez detectado el contorno candidato, se calculan sus 4 vértices y se aplica una **transformación de perspectiva (Homografía)**. Como se observa en la grilla de resultados, esto permite "enderezar" patentes que estaban inclinadas o rotadas, generando una vista frontal estandarizada necesaria para la segmentación de caracteres. Se incluye una etapa de filtrado por relación de aspecto para descartar falsos positivos (como ópticas o rejillas). Aún así, quedan varias detecciones que son incorrectas

Figura 10.

2. Filtrado de Falsos Positivos
Tras la rectificación por homografía, se obtuvieron diversos recortes candidatos. Algunos de ellos correspondían a objetos rectangulares ajenos a las patentes (ópticas, logotipos, texturas del entorno). Para depurar estos resultados, se aplicó un filtrado en dos etapas:

Figura 11 'Patentes finales y Descartes': En una primera instancia se aplicaron filtros basados en las propiedades físicas esperadas de una patente argentina:

- **Geometría:** Se descartaron objetos que no cumplieran con una relación de aspecto de entre 2.0 y 3.15, característica de la forma alargada de la placa. También se filtraron recortes por área mínima y máxima para eliminar ruido diminuto o detecciones excesivamente grandes.
- **Contenido (Densidad de Bordes):** Para diferenciar una patente (rica en textura por los caracteres) de un faro o un logo (generalmente lisos), se calculó la densidad de bordes mediante el operador Canny. Aquellos recortes con una densidad de bordes inferior al 5% fueron descartados por falta de información textual.
    - *Observación:* Como se ve en la figura 12, este paso eliminó exitosamente logotipos (Peugeot, Renault) y ópticas, pero algunos falsos positivos con textura compleja (ej. rejillas o fondos rojizos) lograron pasar a la lista de "Patentes finales".

Figura 11.

Figura 12.

Figura 13: Para eliminar los falsos positivos que superaron el filtro geométrico, se implementó un análisis de **Proyección Vertical**:

- Se binarizó cada candidato y se sumaron los píxeles blancos por columna (eje X), generando un histograma de "tinta".
- Como se observa en los gráficos, las patentes reales presentan un perfil oscilante con picos y valles definidos, correspondientes a la alternancia entre caracteres y fondo. En contraste, los falsos positivos (como la textura roja o el recorte de carrocería) muestran perfiles planos o irregulares sin periodicidad clara.

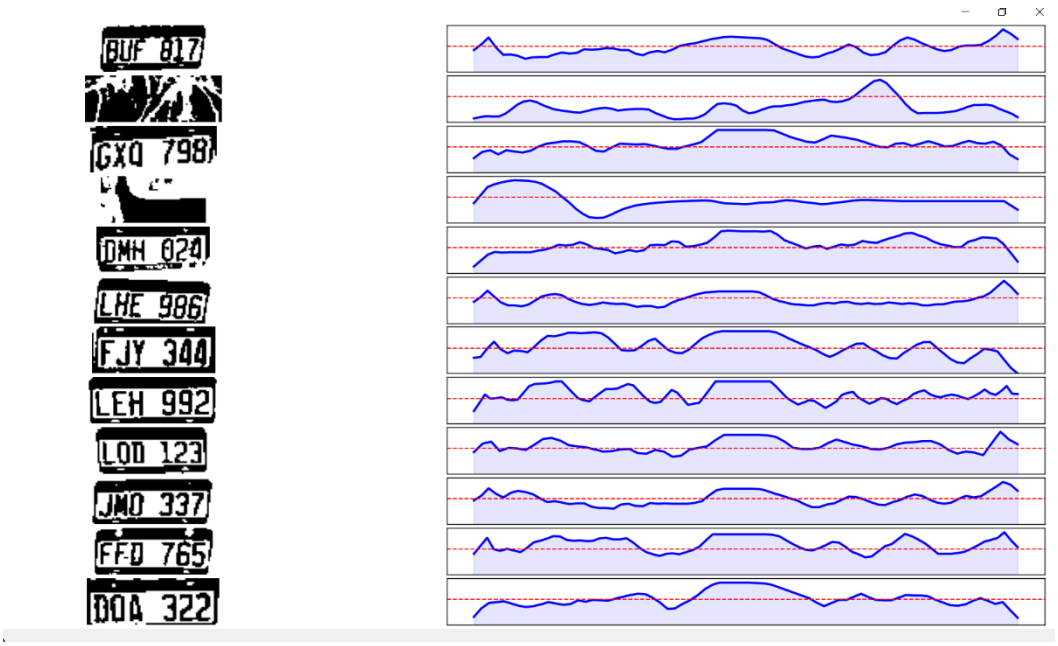


Figura 13.

Figuras 14 y 15 - Patentes Confirmadas y Objetos Descartados: Utilizando el perfil suavizado, se aplicó un criterio lógico: si el recorte no presenta al menos 2 picos significativos (que superen el 60% del máximo local), se considera que no contiene caracteres legibles.

- *Resultado:* Este último paso logró segregar los últimos casos de error, dejando en "Patentes confirmadas" únicamente las imágenes aptas para la segmentación de caracteres, y moviendo el ruido restante a "Objetos descartados".

Figura 14

Figura 15.

3. Segmentación de Caracteres

Para trabajar con el contenido de la patente, se inicia un nuevo pipeline de procesamiento de alta definición:

Figuras 16 y 17:

- Dado que los recortes originales tienen baja resolución, se aplicó un Upscaling x2 con interpolación bicúbica. Esto suaviza los bordes de las letras, facilitando su separación.
- Se normalizó el histograma para maximizar el contraste y se aplicó un Umbralado Adaptativo (Gaussian C). En la imagen resultante se aprecia cómo esta técnica logra binarizar correctamente las letras incluso en patentes con sombras fuertes o iluminación desigual, donde un umbral global fallaría.

Figura 16.

Figura 17.

Figuras 18, 19 y 20

- En primer lugar, con la intención de eliminar los marcos de la patente, se implementó un recorte simple: se eliminó una cierta cantidad de pixeles de la parte superior y la parte inferior del recorte, pasados como parámetros y utilizando como referencia un porcentaje de la altura total. (Figura 18)

- Luego, se implementó un recorte inteligente basado en Proyección de Perfil Vertical. El algoritmo analiza la densidad de píxeles blancos fila por fila desde el centro hacia afuera, eliminando los márgenes superior e inferior sin cortar los caracteres.
- Finalmente, se aplicó una operación morfológica de **Clausura** suave para reconectar trazos rotos dentro de las letras, asegurando que cada carácter sea un componente conexo único.
- *Comentario:* como conclusión a esta etapa, consideramos que tal vez las dos fases de recorte no fueron muy fructíferas en relación con la complejidad del código y la dificultad que tuvo su desarrollo

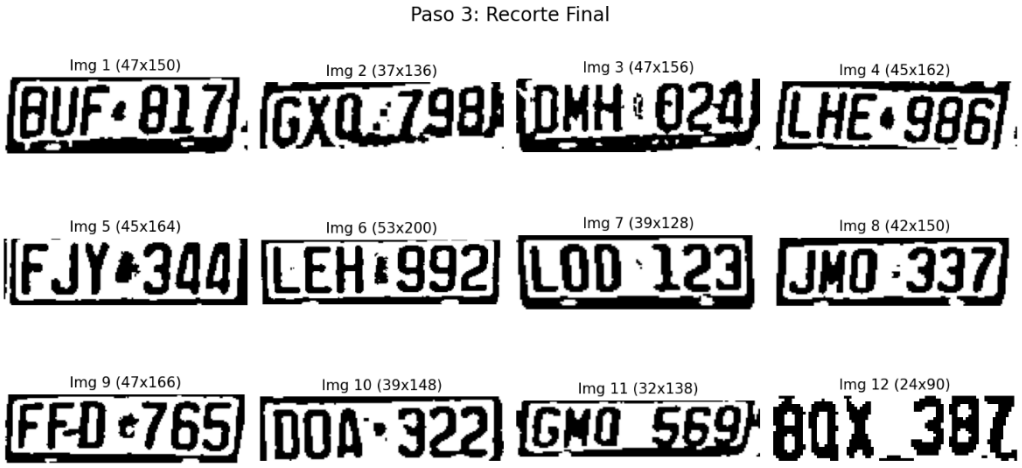


Figura 18.

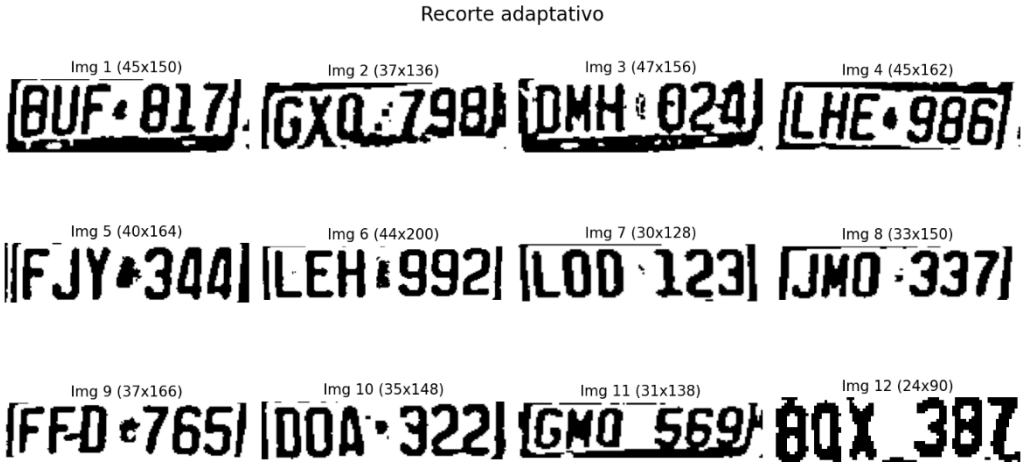


Figura 19.

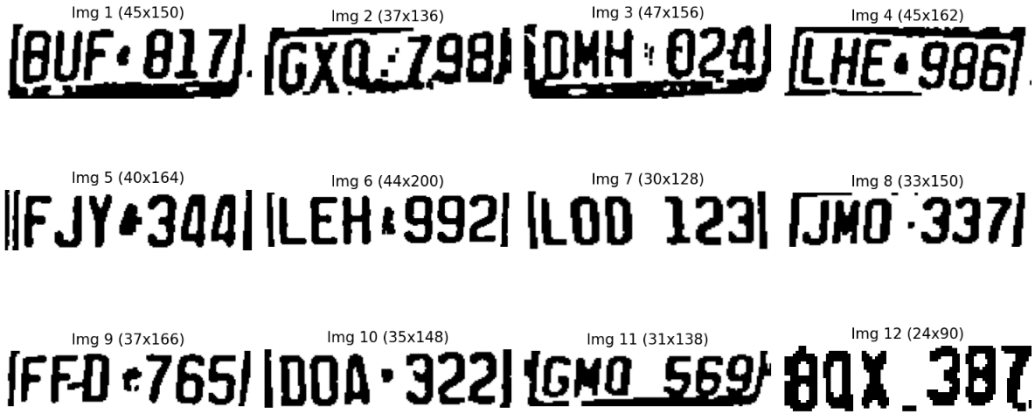


Figura 20.

Figuras 21 y 22:

- Primero se invierte la imagen para detectar componentes conectados (letras blancas sobre fondo negro) y evalúa cada objeto según su altura, proporción y área para descartar el ruido que no tenga forma de carácter. Si tras este filtrado detecta exactamente seis caracteres, clasifica la patente como "OK"; de lo contrario, la marca como "REVISAR" para que sea procesada posteriormente por la etapa 2. (Figura 21)

- Luego, en la figura 22, observamos los resultados de la etapa 2, que realiza lo sig:
  - **Separación de Bloques:** Se observa cómo el algoritmo detecta bloques inusualmente anchos (letras pegadas) y los divide geométricamente.
  - **Limpieza de Bordes**: En patentes donde el texto toca el marco (ej. ID 10), se aplicó una limpieza de márgenes que desconecta las letras del borde, permitiendo su detección individual.
  - **Inferencia Estructural**: En casos de pérdida de información, el sistema utiliza la mediana del ancho de los caracteres detectados para "inferir" la posición de los caracteres faltantes, rellenando los huecos lógicos en la grilla de la patente.

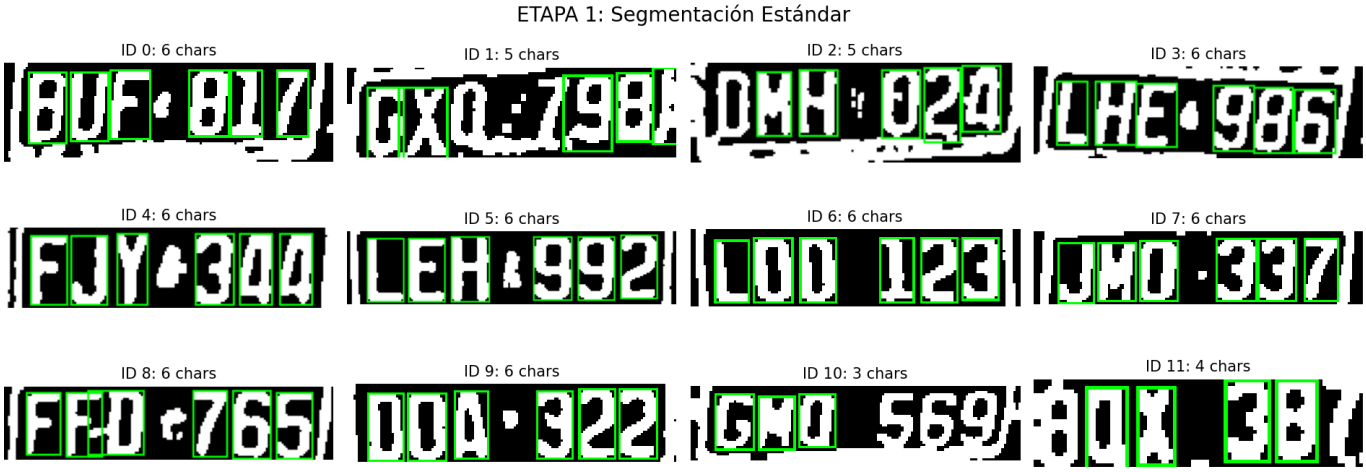


Figura 21

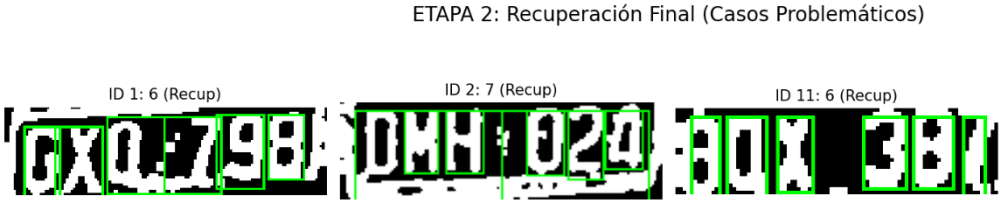


Figura 22.

4. Resultado Final

Figura 23: El resultado final del sistema es la extracción de los 6 caracteres individuales de cada patente válida. La grilla muestra los recortes limpios, binarizados y centrados, listos para ser utilizados como entrada en un sistema de reconocimiento óptico de caracteres (OCR) o para otras utilidades.

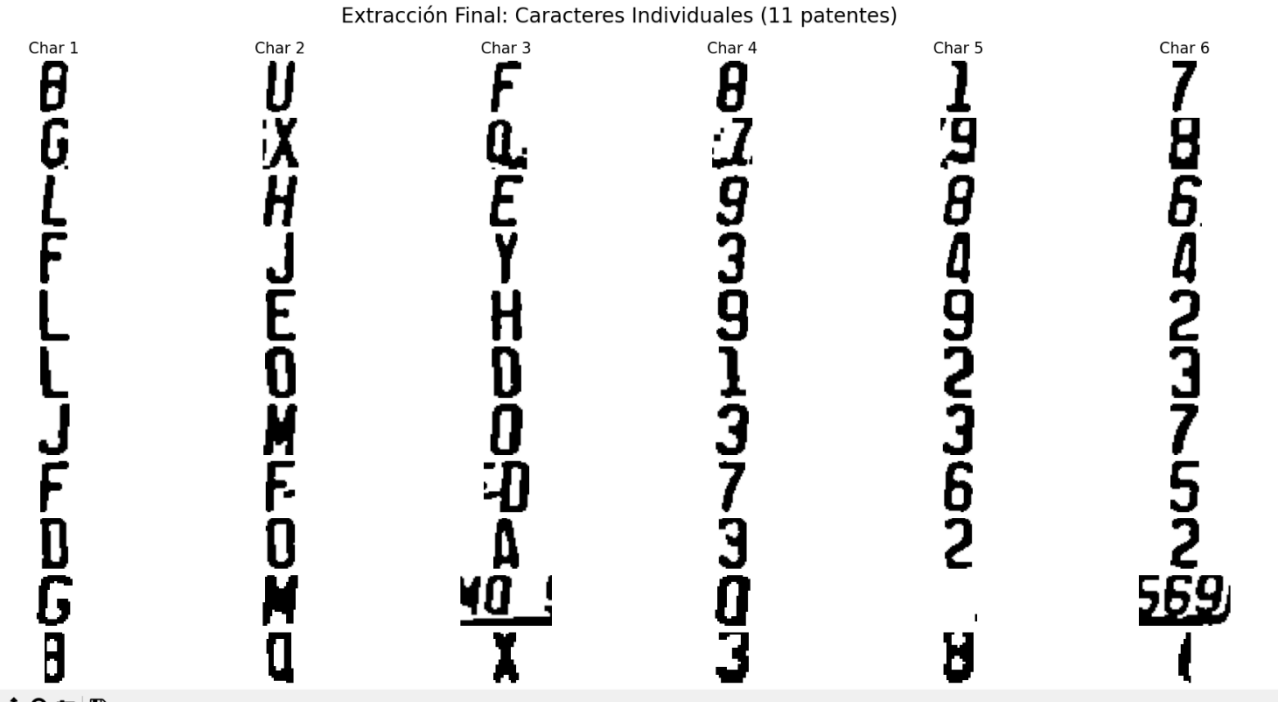


Figura 23

Análisis de Resultados:
El desarrollo de este sistema de reconocimiento de patentes permitió evaluar la complejidad de la segmentación de caracteres. Si bien el pipeline general demostró ser efectivo para la mayoría de los casos, el análisis detallado de los resultados finales expone ciertas limitaciones técnicas y desafíos pendientes.
Durante la **Etapa 2 (Recuperación Final)**, diseñada para corregir segmentaciones fallidas mediante inferencia estructural, se produjo un error de ejecución en el código que no logramos hallar ni corregir, resultando en la omisión del procesamiento de la **Patente 12**. Esta falla impidió que dicha imagen, clasificada correctamente para "revisión", pasara por la lógica de recuperación. Adicionalmente, la inspección visual de los caracteres extraídos revela imperfecciones en la lógica de separación y limpieza:

- **Patente 2:** Si bien la lógica de división logró separar los caracteres "Q" y "7" que estaban unidos, el recorte resultante conserva ruido y fragmentos de los caracteres adyacentes.
- **Patente 10:** La estrategia de "limpieza de bordes" no fue suficiente para desconectar los números del marco o entre sí, resultando en bloques de caracteres fusionados que no pudieron ser individualizados.
- **Patente 11:** Se observa un recorte excesivo en el último dígito, evidenciando que los parámetros de inferencia o bounding box fueron demasiado agresivos, perdiendo información estructural del carácter.

Conclusión: El Trabajo Práctico evidenció que, en el procesamiento de imágenes, la **segmentación** suele ser una tarea más crítica y sensible al ruido que la detección del objeto en sí. Pudimos aprovechar el uso de técnicas como la **Homografía** y la **Binarización Adaptativa,** que resultaron fundamentales para normalizar la entrada, pero se concluye que depender exclusivamente de filtros geométricos rígidos (área, aspecto) es insuficiente ante la variabilidad del mundo real (sombras, suciedad, marcos).
La implementación de una arquitectura en dos etapas (una estricta para casos ideales y una permisiva con inferencia para casos complejos) parece haber sido un enfoque correcto, aunque su implementación requiere un ajuste fino de parámetros mucho más exhaustivo para garantizar robustez en todos los escenarios.