## Segmentación de Patentes y Caracteres

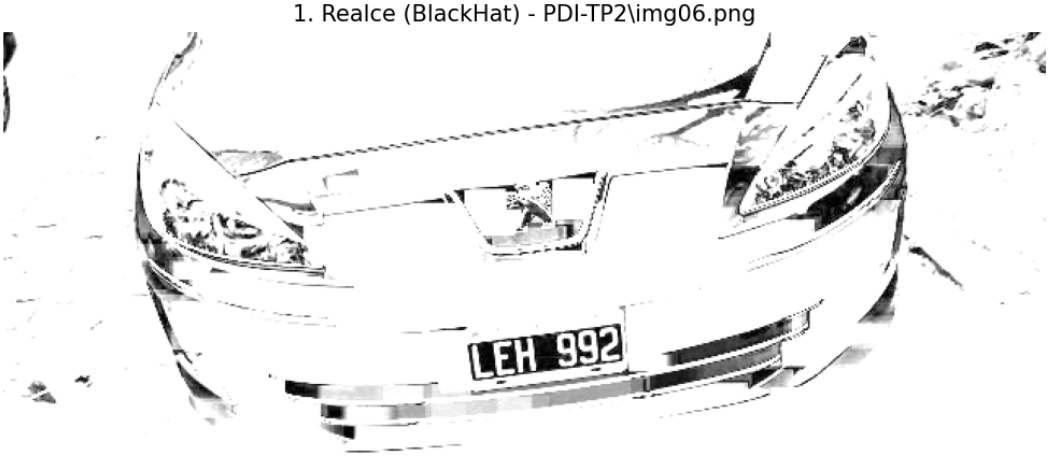
**Breve introducción**: A diferencia del código de monedas y dados, en este ejercicio tuvimos varios inconvenientes por la gran variedad de patentes, iluminaciones, tamaños y calidades de imagen.
Intentamos ser lo más abarcativos posible, a pesar de que entendemos que las primeras patentes no llegan a segmentarse de forma correcta, luego vemos que las últimas patentes tienen una gran mejoría pudiendo tomar los caracteres de buena forma.

## Qué hicimos

El objetivo del código fue detectar automáticamente la patente en cada imagen, localizarla, segmentar los caracteres y cortarlos de forma ordenada.
Para esto, armamos una función procesar_imagen que se aplica en forma automática a todas las imágenes img*.png de la carpeta, usando glob.

Carga y escala de grises
Se lee cada imagen y se la pasa a escala de grises para simplificar los cálculos.

Realce de la placa con BlackHat
Aplicamos un filtro morfológico BlackHat con un kernel rectangular grande, y luego un ajuste fuerte de contraste (convertScaleAbs con alpha y beta) para resaltar las zonas donde la placa se diferencia del fondo. La idea es que la zona de la patente quede casi “dibujada” en la imagen resultante.



Umbralización y bordes
Sobre esa imagen realzada usamos Otsu (con inversión) para binarizar, y después Canny para extraer los bordes principales.

Dilatación y cierre para unificar la placa
Con un kernel rectangular se dilatan los bordes y luego se aplica un closing para unir fragmentos y rellenar cortes. Esto genera contornos más completos de posibles patentes.

Selección de la placa
Se buscan contornos externos y, para cada uno, se calculan relación de aspecto, área y densidad de píxeles blancos dentro del bounding box.
Solo se aceptan como candidatos los contornos con forma y tamaño similares a una placa.
De esos candidatos, se elige el de mayor área como la placa final y se recorta esa ROI.

Una vez tenemos la placa candidata, la mostramos con su bounding-box en la imagen a color.


Preprocesamiento de caracteres dentro de la placa
Sobre la ROI en gris se aplica un suavizado Gaussiano y una umbralización Otsu NORMAL (caracteres blancos, fondo negro).
Luego usamos operaciones morfológicas (closing vertical y un thinning suave mediante erosión y resta) para limpiar la placa y resaltar el “esqueleto” de los caracteres.

Detección, filtrado y orden de caracteres
Se extraen los contornos en la imagen de caracteres y se los filtra con criterios relativos al tamaño de la ROI:

altura mínima y máxima,

relación de aspecto,

área mínima y máxima.

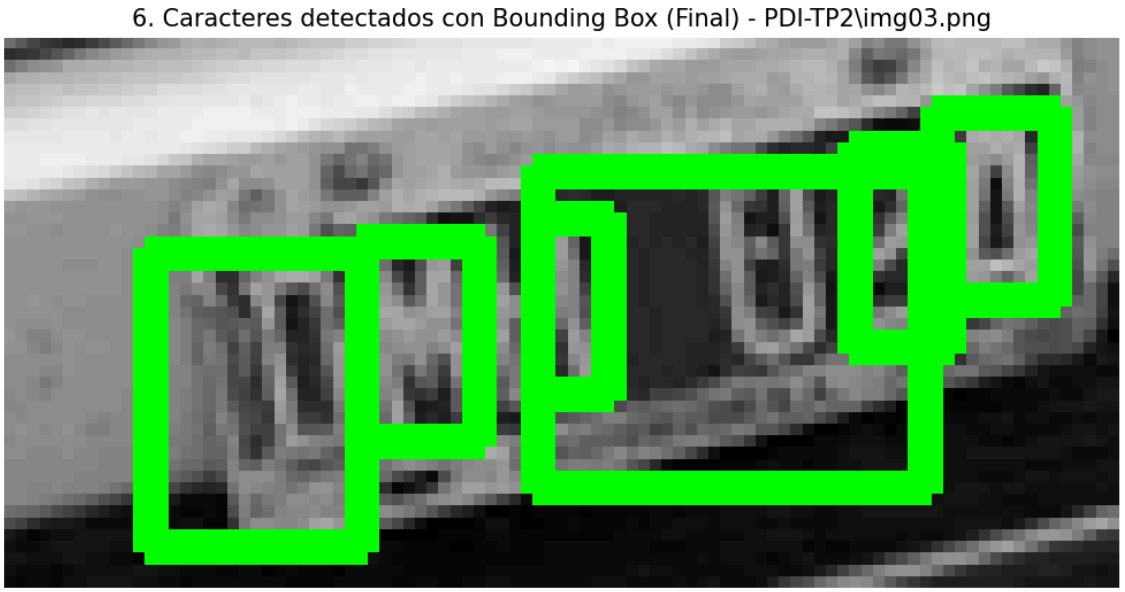
Los candidatos que cumplen todo se ordenan de izquierda a derecha y se dibujan sobre la ROI. Finalmente se recorta cada carácter individual y se muestra en subplots.

## Qué problemas tuvimos

Gran variabilidad entre imágenes: algunas patentes están muy contrastadas y otras muy lavadas; algunas están de frente y otras algo inclinadas o desplazadas; incluso cambia el formato (patentes viejas/nuevas).

Sensibilidad a la iluminación: en varias imágenes el fondo tiene reflejos o sombras que generaban bordes “falsos” similares a una patente, haciendo que el codigo eligiera primero contornos incorrectos.

Tamaño y calidad de los caracteres: en ciertas imágenes las letras/números salen muy finos o parcialmente pegados, lo que hacía que el contorno del carácter quedara roto o se uniera con otro.



Parámetros frágiles: los umbrales de área, relación de aspecto y densidad de bordes no funcionaban igual para todas las imágenes. Ajustar algo para una patente rompía otra, por lo que hubo que iterar bastante hasta encontrar valores razonablemente generales.

(Respecto a esto, creo que sobre el final encontramos cual era una buena práctica para ajustar estos parámetros con menor cantidad de iteraciones)

## Qué técnicas aplicamos para resolverlo

Para ir atacando estos problemas fuimos incorporando y ajustando varias técnicas:

Realce específico de la placa con Blackhat + contraste
Esto ayuda a resaltar la patente aun con fondos complejos o iluminación desigual, porque realza variaciones de brillo en estructuras rectangulares.

Umbralización Otsu (invertida y normal)
Usamos Otsu invertido para resaltar la placa en la etapa de detección global, y Otsu normal dentro de la ROI para obtener caracteres blancos bien definidos sobre fondo negro.

- Operaciones morfológicas (dilatación, cierre, thinning)
- Dilatación + cierre sobre bordes para unificar el contorno de la placa.
- Closing vertical y erosión con resta para afinar caracteres y separar mejor unas letras de otras.
- Filtros geométricos y de densidad para la placa

En la etapa de detección se filtran contornos por:

relación de aspecto (placa más larga que alta), revisando las medidas legales para estas (aunque mucho no sirvió, dado que tuvimos que ampliar esos rangos)

Área mínima y máxima,

densidad de píxeles blancos (descartando regiones con demasiados bordes dispersos).

Criterios relativos al tamaño de la ROI para los caracteres
Los umbrales de altura, área y relación de aspecto de los caracteres se definen en función de la altura y el área de la placa, y no como valores fijos. Eso hace que el filtro se adapte mejor a patentes más grandes o más chicas.

En resumen, aunque el problema resultó bastante más complejo de lo esperado por la diversidad de imágenes, el código terminó combinando varias técnicas de procesamiento morfológico, análisis de contornos y filtrado geométrico para lograr una detección y segmentación de caracteres razonablemente robusta en un conjunto variado de patentes.