**PROCESAMIENTO DE IMÁGENES: TRABAJO PRÁCTICO 2**

**Francisco Domingo** – **Lara Valeri**

**TUIA**

**2024**

**Introducción**

En este trabajo práctico resolveremos dos consignas que fueron tratas de formas separadas. Se provee a la cátedra de los dos scripts *Ej1.py* y *Ej2.py* donde se encuentran las soluciones a los problemas propuestos.

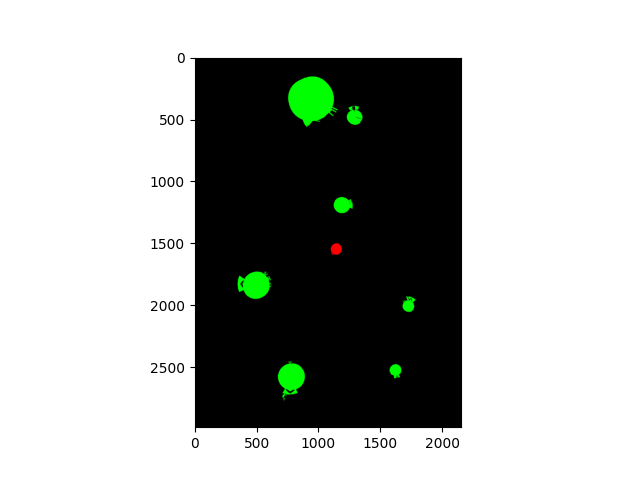
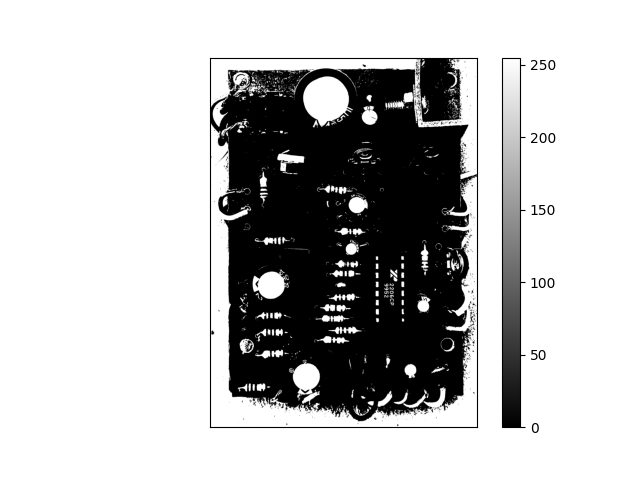
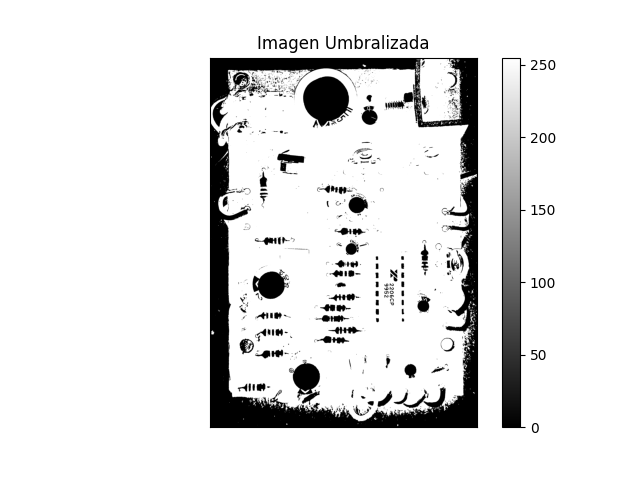
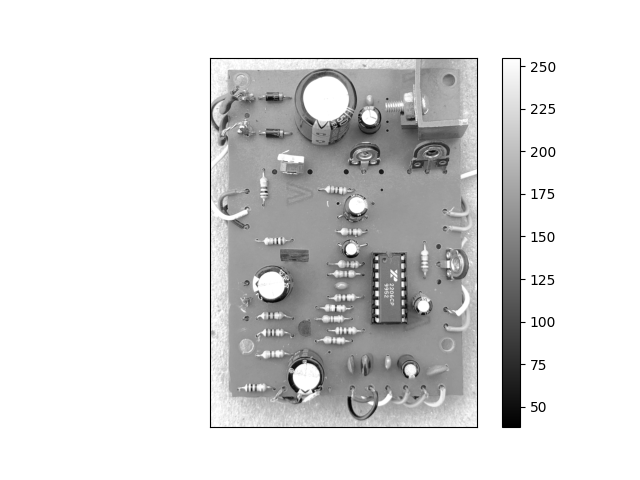
**Problema 1:** Detección y clasificación de componentes electrónicos

Este problema lo hemos dividido en tres sub-problemas diferentes, uno para cada componente que sea desea segmentar en la imagen (chip, capacitores y resistencias). Entendimos que cada una tiene sus particularidades, y el resultado es mejor en tanto se procese de diferentes formas la imagen de entrada.

En todos los casos creamos un pipeline de procesamiento de la forma:

1. Pre-procesamiento, mejorando brillo y contraste para diferenciar lo mejor posible el fondo de los componentes.
2. Umbralado, con diferentes parámetros para cada sub-problema, ajustando en función de lo que veíamos como resultado, intentando quedarnos que “se activen” solo los componentes que buscamos.
3. Aplicamos algoritmos de clausura, apertura, Canny, gradiente morfológico y/o rellenado de huecos, según lo vimos conveniente, para lograr segmentar los objetivos.
4. Análisis de componentes conexas, buscando patrones que sean propios de las etiquetas que se corresponden con los objetos deseados, y con ninguna más (área – relación de aspecto – morfología circular).

***Capacitores***



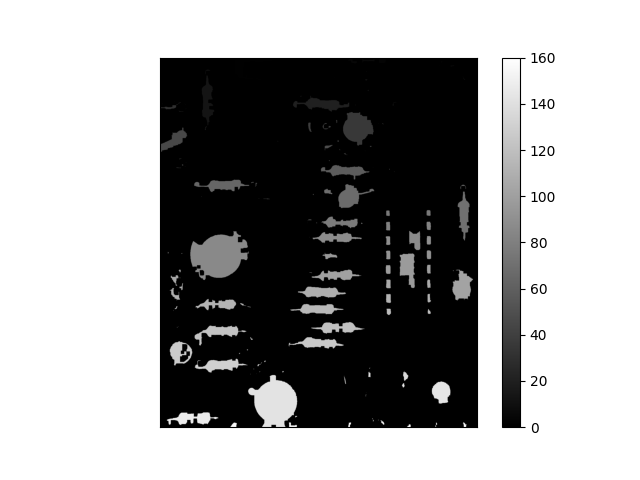
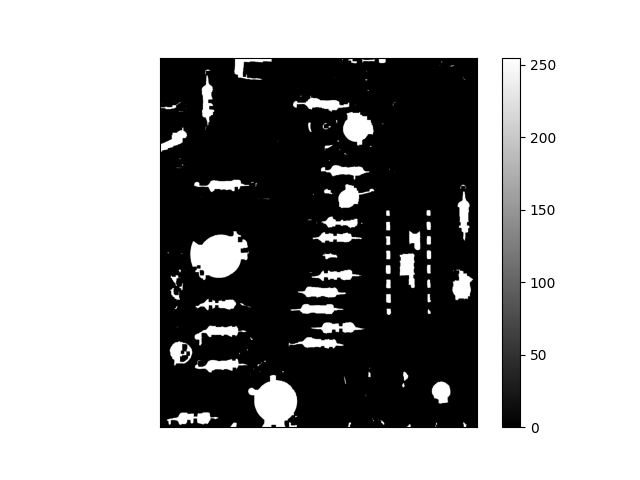
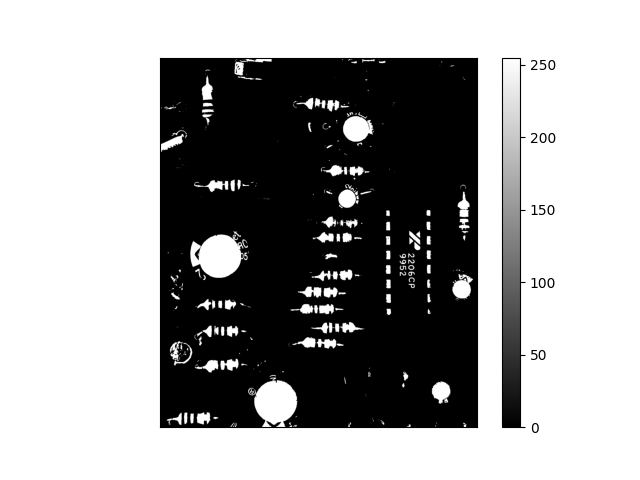
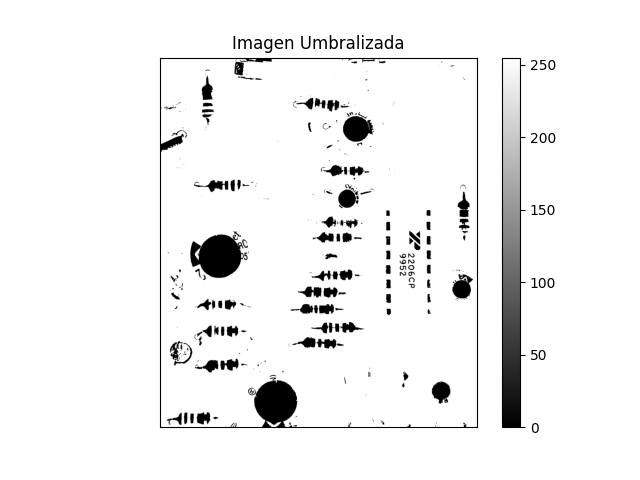
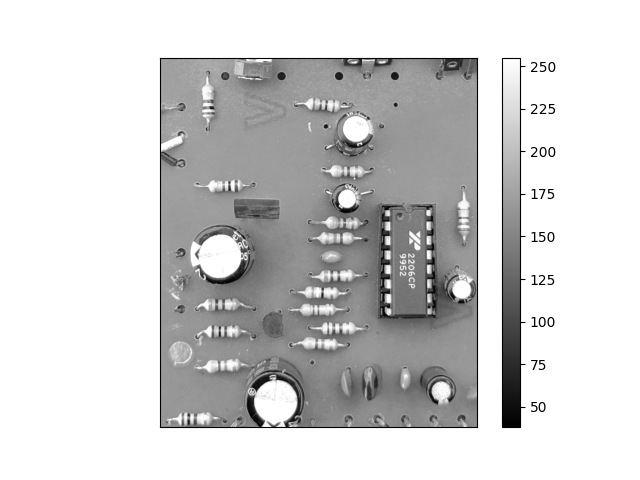
Así se ve nuestro procesamiento de la imagen para segmentar los capacitores. Los pasos son:

1. Conversión a escala de grises.
2. Umbralado.
3. Complemento del umbralado.
4. Hallado de componentes conexas bajo ciertas condiciones.

A partir de esa imagen realizamos una búsqueda de componentes “redondas”, alcanzando el objetivo deseado con la variable RHO en 0.23 y AREA en 6500. El resultado final se encuentra aquí debajo.



***Resistencias***:

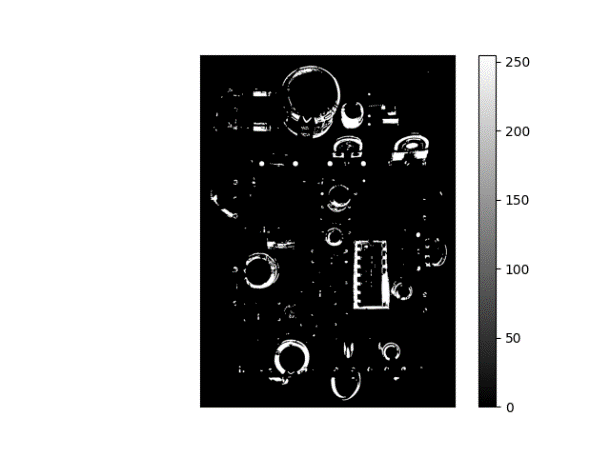
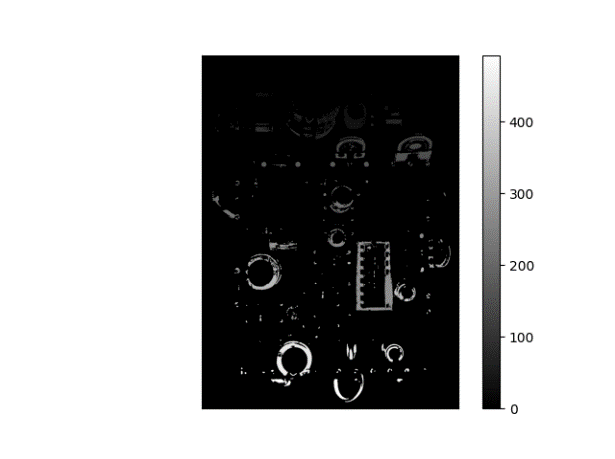
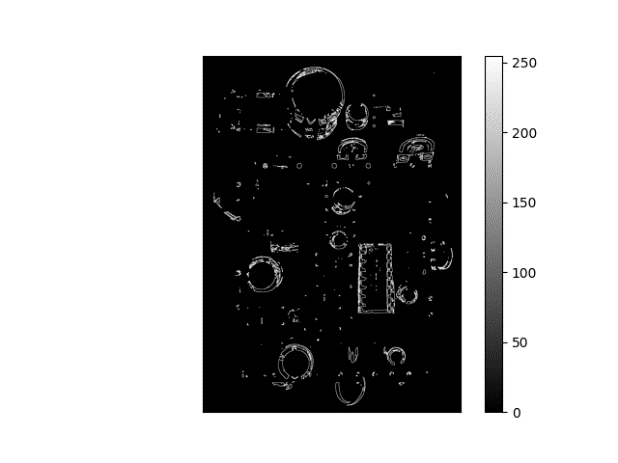
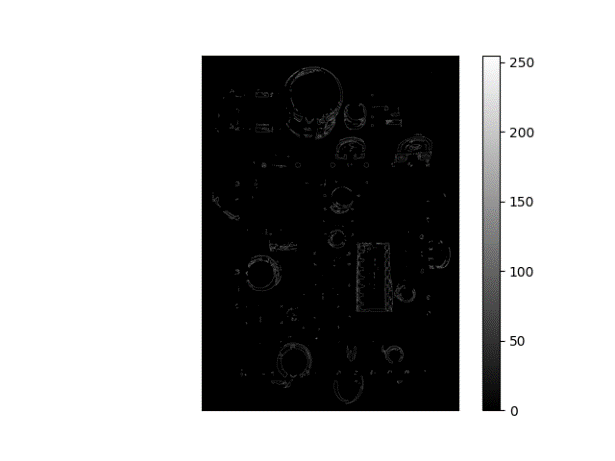
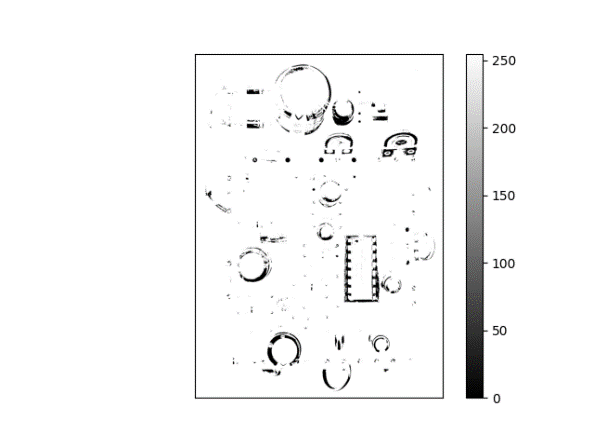


1. Recorte y escala de grises.
2. Umbralado.
3. Complemento.
4. Clausura (8 iteraciones).
5. Hallado de componentes conexas, con condición de relación de aspecto y área mínima.

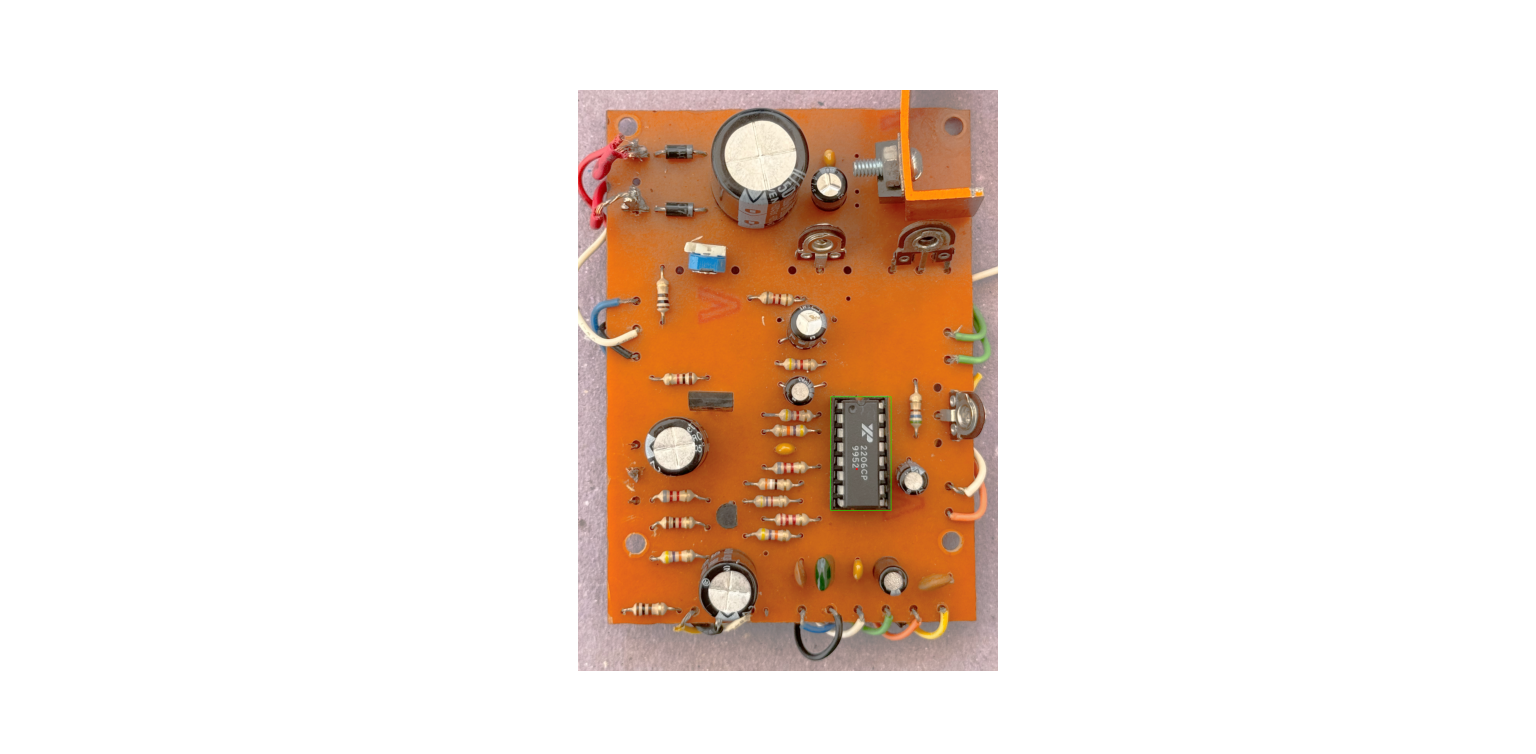
El mayor inconveniente en este problema era la subdivisión de etiquetas que generaban las marcas de colores sobre las resistencias. El pre-procesamiento (brillo-contraste) en ningún momento fue suficiente como para no perder el borde entre la resistencia y el fondo a la altura de dichas marcas. Además, la imagen se recortó para mejorar la performance del procesamiento y reducir ruido. En el resultado final encontramos todas las resistencias, pero no pudimos evadir tomar el componente celeste que se encuentra arriba a la izquierda. Consideramos que igualmente el algoritmo es robusto.



***Chip***



1. Escala de grises y pre procesamiento personalizado.
2. Umbralado.
3. Algoritmo de Canny.
4. Gradiente morfológico para mejorar las conexiones y separar el chip del capacitor a su derecha.
5. Rellenado de huecos.
6. Hallado de componentes conexas.

Con esta segmentación, pudimos encontrar el chip referenciando la etiqueta correspondiente.

**Problema 2:** Detección de patentes

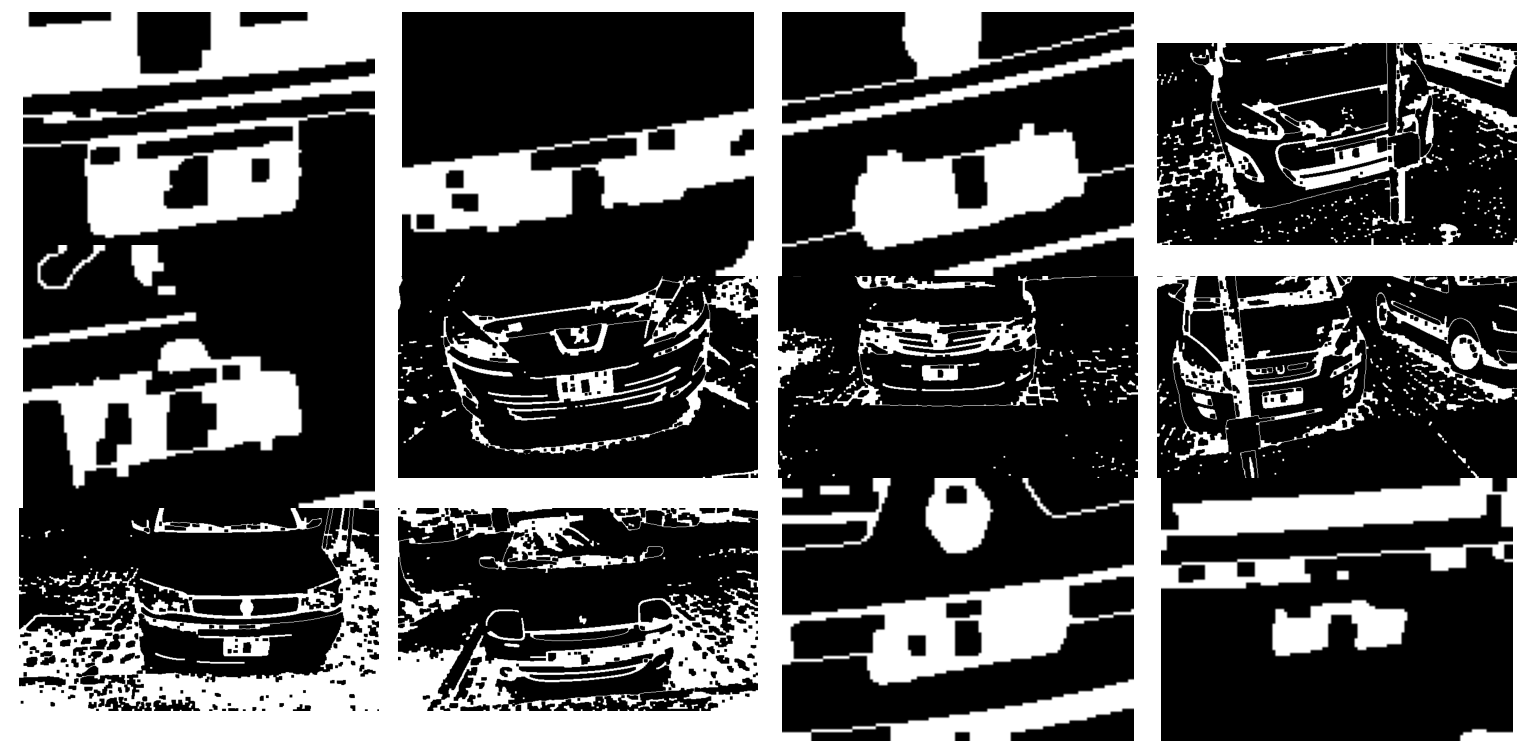
En este problema intentamos crear un algoritmo de detección de patentes, que logre segmentarlas a partir de imágenes conteniendo otros objetos (el mismo auto y variados objetos de la vía pública), tomadas desde diferentes ángulos y distancias.

La primera particularidad de la que queremos hacer uso es que, por lo general, en las imágenes provistas, no hay objetos celestes. Esto puede ser de utilidad, ya que todas las patentes (que son del formato AAA-999) tienen “Argentina” escrito arriba, en ese color. Implementamos código que pueda encontrar, con cierto margen de error, esos colores, sumando la condición de que 10 píxeles hacia abajo debería encontrar un cuadrado negro (que se corresponde con el espacio que se encuentra entre las letras y los números.

Solo teniendo eso en cuenta, logramos encontrar y recortar 6 patentes, reduciendo el área de búsqueda considerablemente. Para el resto de imágenes, se volvió a copiar la imagen en formato original.



Redujimos el rango dinámico, probamos varios métodos de generación de bordes, eligiendo Canny por sobre el resto, y aplicamos clausura para poder obtener el “rectángulo blanco” que se corresponde con las patentes, de la mejor forma posible.



Además, aplicamos un algoritmo de rellenado de huecos para mejorar el resultado de buscar componentes conexas.



Haciendo uso de las etiquetas provistas, segmentamos las patentes y las ploteamos sobre la imagen original, obteniendo el siguiente resultado final.



Incluso probando con apertura-clausura en diferentes órdenes y diferente cantidad de iteraciones, o con algoritmos de reconstrucción morfológica, cambiando los parámetros de umbralado y saturación, **no hemos podido segmentar correctamente todas las patentes**, algunas por particularidades del escenario (color de auto, protectores de patente, lejanía de la toma, etc.) y otras simplemente por la calidad de las imágenes y nuestra falta de expertise en el tema.

El problema de segmentar las letras lo atacamos de diferentes frentes, aplicando muchos algoritmos de procesamiento de imágenes (dilataciones, erosiones, aperturas, clausuras, gradientes morfológicos, canny, etc.), en diferentes órdenes y con diferentes parámetros, pero en ningún caso hemos podido armar un algoritmo lo suficientemente robusto como para poder segmentarlas del fondo de la imagen con un grado aceptable de error.

