



FACULTAD DE
CIENCIAS EXACTAS,
INGENIERIA Y AGRIMENSURA



Tecnicatura Universitaria en Inteligencia Artificial

Procesamiento de Imágenes

“TRABAJO PRÁCTICO N° 2”

Balverdi, Valentina - (B-6588/9)

Caballero, Franco - (C-7328/8)

Grimaldi, Damián - (G-5977/3)

1º Semestre - Año 2025

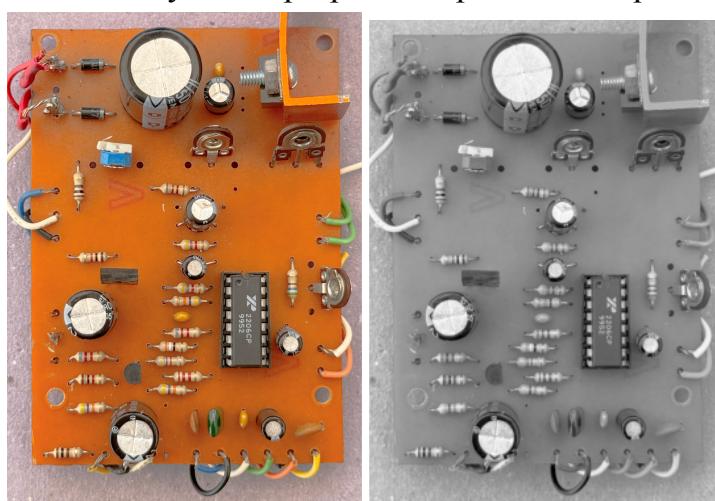
Problema 1 - Detección y clasificación de componentes electrónicos

Introducción al problema

En este trabajo se aborda el procesamiento de una imagen de una placa de circuito impreso (PCB) con el objetivo de identificar y clasificar automáticamente sus componentes electrónicos principales: resistencias, capacitores y chips. El desafío consiste en distinguir cada elemento a partir de su forma y tamaño, pese a las variaciones de iluminación y la similitud de tonos en el fondo. Para ello, se aplica una cadena de operaciones de preprocesamiento y segmentación que incluye la conversión a escala de grises, un filtro de mediana para atenuar el ruido, la detección de bordes mediante el operador Canny y el uso de operadores morfológicos para reforzar los contornos. Finalmente, se recurre al análisis de componentes conectados para aislar cada pieza y clasificarla según criterios dimensionales predefinidos.

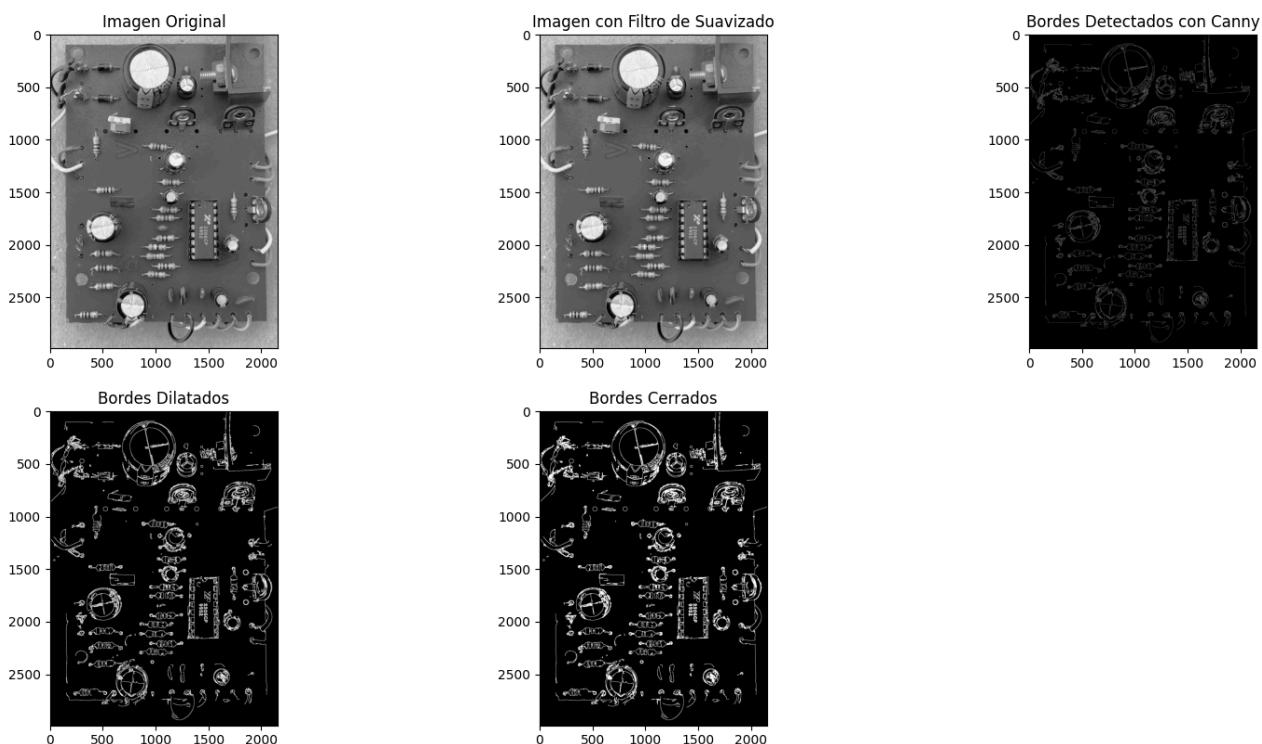
Desarrollo y resolución

Iniciamos con la conversión de la imagen original a escala de grises para simplificar el análisis de intensidad. Nos encontramos con similitud de los tonos grises en el fondo y en los propios componentes lo que dificulta la segmentación.



Comparación imagen original e imagen en escala de grises

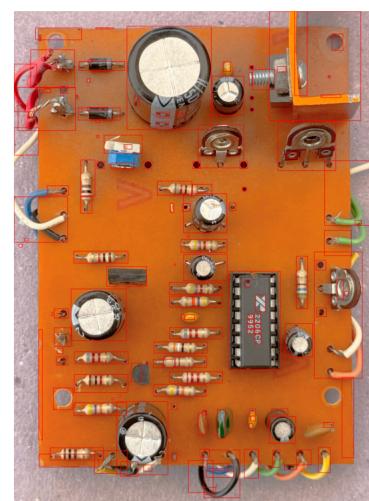
A continuación, utilizamos la función `detectar_bordes()`, que recibe la imagen en escala de grises y aplica en primer lugar un filtro de mediana. Este filtro no lineal de paso bajo sustituye cada píxel por la mediana de sus vecinos inmediatos, atenuando el ruido sin sacrificar la precisión de los contornos. A continuación, la función ejecuta el operador Canny para identificar bordes basados en gradientes de intensidad, generando una máscara binaria. Para mejorar la continuidad de los contornos, la máscara se somete primero a dilatación, que engrosa y refuerza las líneas detectadas; y luego a clausura (cierre morfológico), que conecta discontinuidades y afina las formas, obteniendo bordes continuos y bien definidos.



Procedimiento de detección de bordes.

Una vez realizado esto, pasamos a visualizar cómo detecta todos los componentes, para poder guiarnos de cómo segmentarlo. A partir de ello, procedimos con 3 funciones (`detectar_resistencia()`, `detectar_chips()` y `detectar_capacitadores()`), que se encargan de detectar cada componente. Las 3 funciones utilizan el mismo principio: utilizar de cv2, la función de `connectedComponentsWithStats`.

Con la función de `detectar_chips()`, a partir de los stats obtenidos, filtramos por un ancho y un alto para

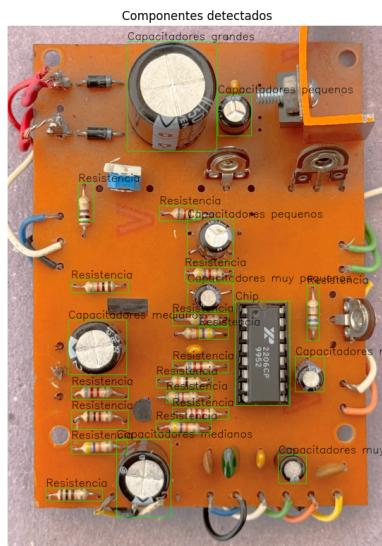
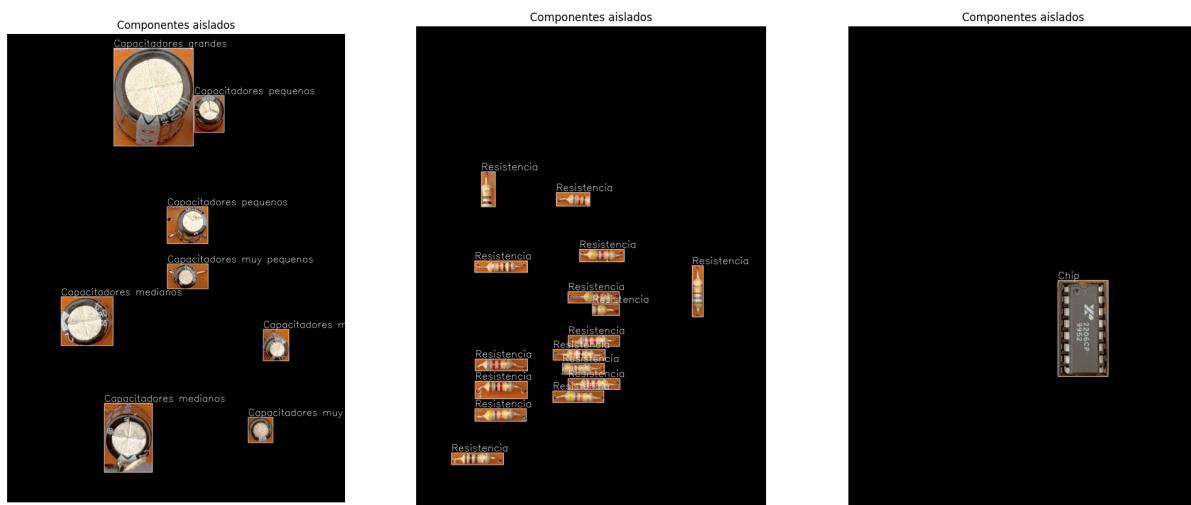


almacenar en una lista las coordenadas del mismo, y posteriormente aislarlo.

Con la función **detectar_capacitadores()**, se realizó el mismo procedimiento que con el chip, pero además, fuimos clasificando los capacitores por su tamaño (grandes, medianos, pequeños, y muy pequeños) almacenandolos en una lista de cada tipo.

Con la función de **visualizacion_componentes_aislados()**, visualizamos cada uno de los componentes detectados de forma individual.

Por último generamos una imagen, con todas los componentes detectados junto a su respectiva etiqueta.



Conclusión

El algoritmo implementado demostró ser eficaz para la detección y clasificación de los distintos componentes electrónicos sobre la PCB. Al convertir la imagen a escala de grises y aplicar un filtro de mediana seguido de Canny, logramos resaltar los bordes; luego, con operaciones de dilatación y clausura, reforzamos las formas para facilitar la segmentación.

Gracias a la función de componentes conectados, pudimos aislar cada elemento y, mediante criterios de tamaño y forma, distinguir resistencias, chips y capacitores (estos últimos clasificándolos además según su tamaño). Finalmente, la visualización individual y la imagen compuesta con todas las detecciones confirmaron que el procedimiento cumple con los requisitos planteados.

Problema 2 - Identificación de resistencias eléctricas

Introducción al problema

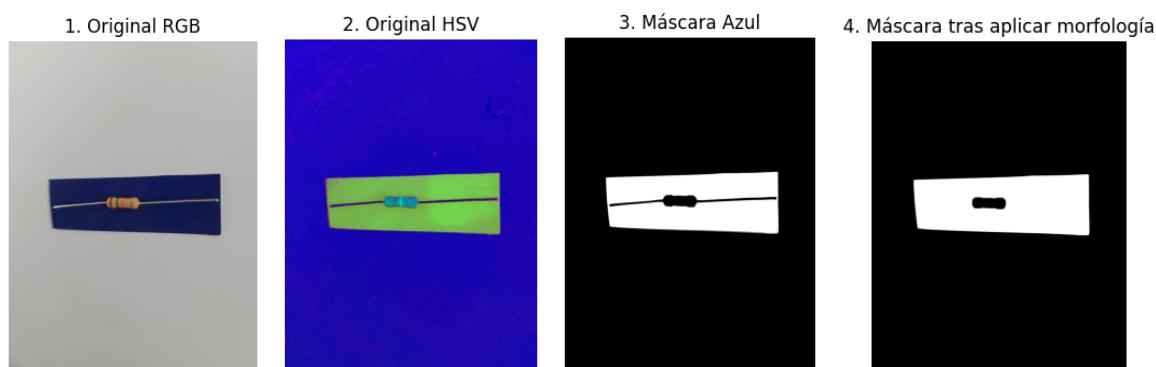
En este problema se aborda la elaboración de algoritmos en Python para la segmentación y clasificación de resistencias eléctricas a partir de fotos tomadas desde diferentes perspectivas. El conjunto de datos está compuesto por 40 fotografías de 10 resistencias distintas (cada una fotografiada en cuatro perspectivas diferentes). Cada imagen muestra una resistencia apoyada sobre un rectángulo azul con fondo blanco.

El objetivo es, primero, transformar cada imagen para obtener una vista superior del rectángulo azul donde se encuentra la resistencia; luego, a partir de esas vistas superiores, localizar las tres bandas de color que definen el valor de la resistencia (ignorando la banda dorada que siempre se encuentra ligeramente separada); y finalmente, traducir esos colores a dígitos y calcular el valor numérico en ohmios.

Desarrollo y resolución

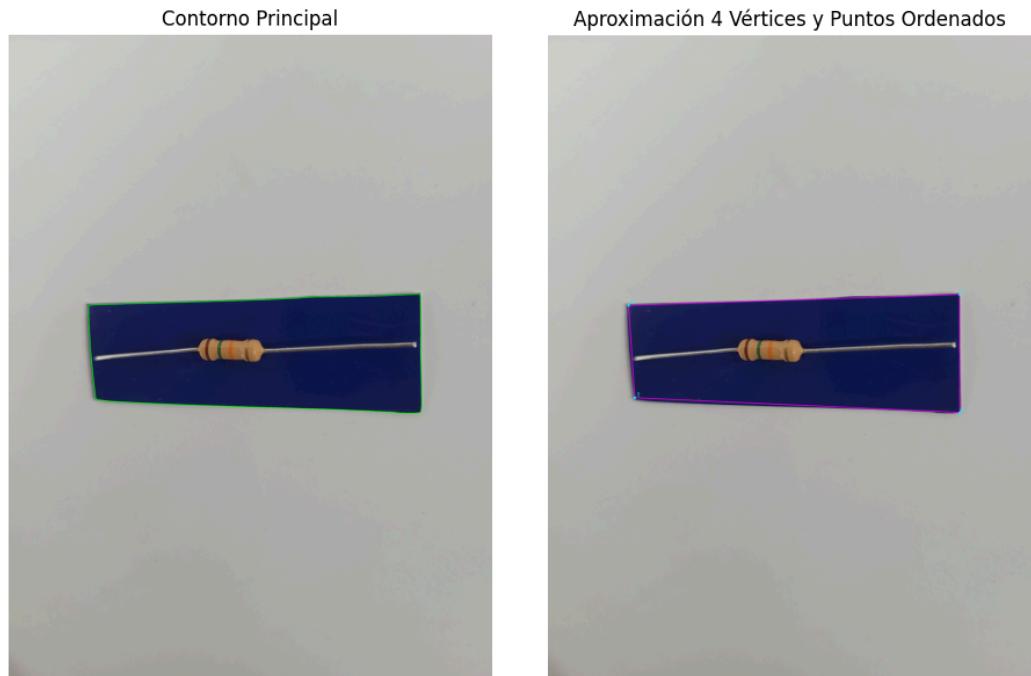
En primera instancia, nos enfocamos en obtener la vista superior del rectángulo azul sobre el que se encuentran las resistencias.

Con la función `crear_mascara_azul()` cada imagen original se convirtió a HSV para filtrar los píxeles azules del rectángulo, este formato de imagen se caracteriza por ser ideal para la aplicación en estos casos. Además, se aplicó cierre y apertura morfológica para evitar que el rectángulo se dividiera en componentes más pequeños luego.



Posteriormente, con `extraer_rectangulo_azul()` se detectó el contorno de mayor área (el rectángulo) y se aproximaron sus vértices a un polígono de cuatro esquinas. Los cuatro puntos obtenidos se reordenaron (top-left, top-right, bottom-right, bottom-left) para luego calcular la homografía que endereza la vista y

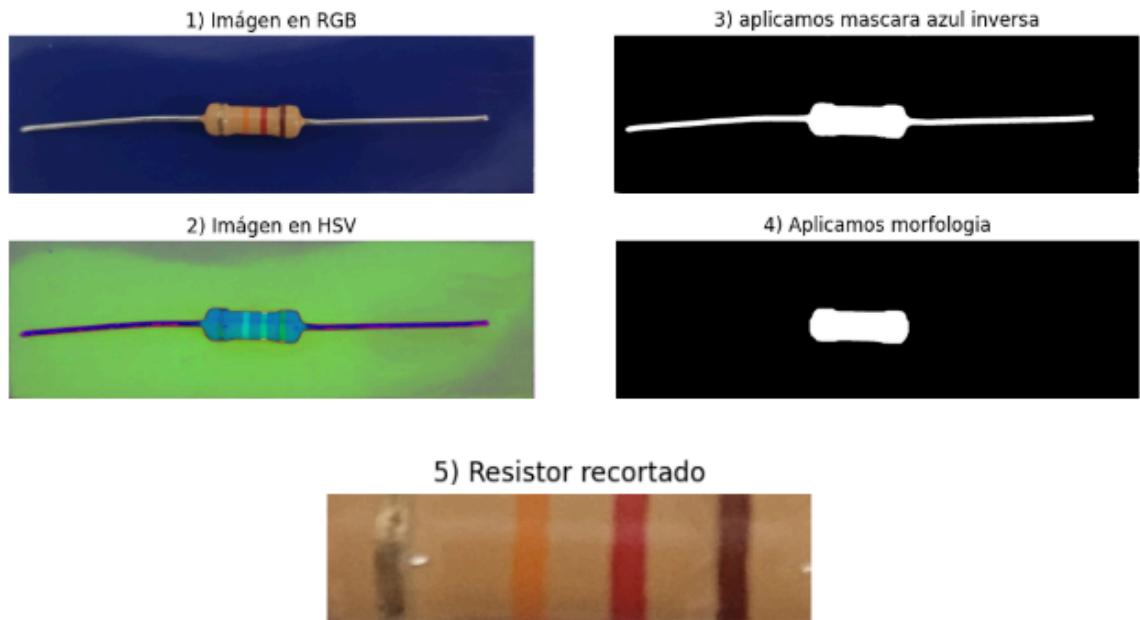
recorta la imagen al área del rectángulo con la función `convertir_a_vista_superior()`. El resultado son 40 archivos “R{i}_j_out.jpg” con la resistencia en vista superior.



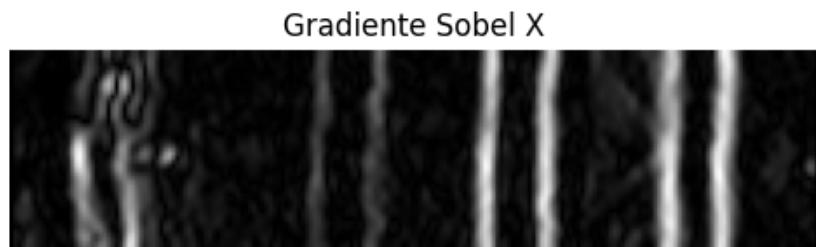
Área azul corregida con homografía



La segunda parte consistía en detectar el valor de la resistencia mediante sus colores. Para ello, empezamos desde el uso de las imágenes llamadas “R{i}_a_out.jpg” y las sometimos a un procedimiento inicial muy similar al que empleamos con el rectángulo azul, con la diferencia de que en este caso nos interesaba lo que no era azul, por lo que aplicamos una máscara para identificar esa zona con la función `mascara_azul_inversa()` donde además, utilizamos operaciones morfológicas con diferentes tamaños de kernels. Luego, con la función `extraer_resistencia()` detectamos contornos y conseguimos un rectángulo que encuadre la parte importante de la resistencia para hacer un recorte.



A continuación, describimos la segunda fase del algoritmo, centrada en la detección y reconocimiento de las bandas de colores que definen el valor de cada resistencia.

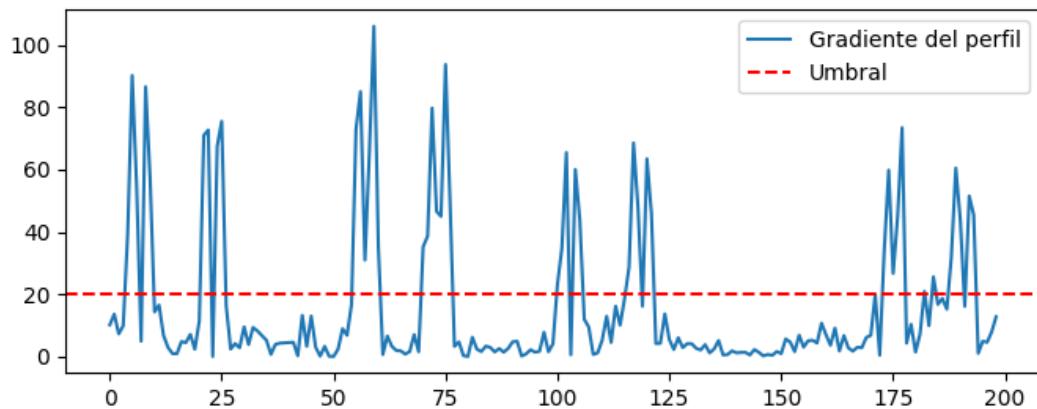


Para localizar las transiciones de color en la resistencia redimensionada, aplicamos la función **bordes_con_sobel()**, que convierte la imagen a escala de grises, ecualiza localmente el histograma (CLAHE) y suaviza el resultado con un filtro Gaussiano antes de calcular el gradiente horizontal (Sobel). Sobre esta imagen de bordes, **detectar_lineas_verticales()** calcula el perfil de intensidad promedio por columna y deriva su gradiente para encontrar los picos que corresponden a los bordes de cada banda. Mediante un bucle adaptativo se ajusta un umbral relativo para obtener exactamente ocho posiciones de borde (dos por banda), descartando falsos positivos cercanos según una distancia mínima configurable.

Franjas detectadas: 8 (umbral: 0.190)



Perfil derivado con umbral

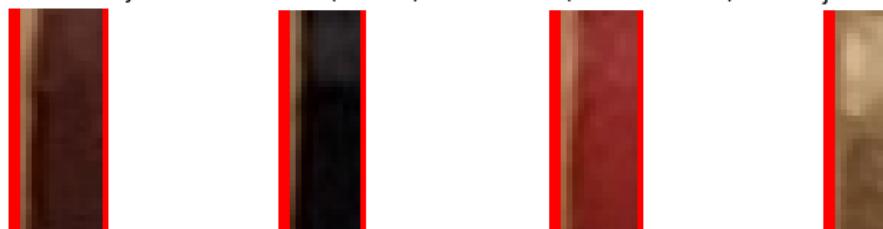


Una vez localizadas las líneas límite, [detectar_bandas\(\)](#) las agrupa de dos en dos formando tuplas (inicio, fin) de cada franja de color. Calcula las distancias entre las franjas internas para decidir de qué lado está la banda dorada (que siempre aparece más separada) y, según ello, ordena las tres bandas útiles en la secuencia “Banda 1 (más alejada de la dorada)”, “Banda 2 (central)” y “Banda 3 (más cercana a la dorada)”. Cada par de coordenadas se recorta con [recortar_banda\(\)](#), que convierte la banda a RGB y extrae la región entre las columnas detectadas.

Franja dorada a la derecha



Banda 1 (más lejana) Banda 2 (media) Banda 3 (más cercana) Franja Dorada



Para asignar a cada banda un nombre de color, `puntos_promedio_color_rgb()` toma en la banda recortada cinco muestras equidistantes en vertical alrededor del centro y calcula la media en cada canal R, G y B. Con esos valores, la función `detectar_color_rgb()` compara el vector promedio con rangos predefinidos para cada color base (p. ej. Marrón, Rojo, Naranja, etc.). Si cae dentro de uno de los rangos, devuelve el nombre correspondiente; en caso contrario, marca la banda como “desconocido”.

Los nombres de colores detectados se pasan luego a `calcular_resistencia()`, que utiliza un diccionario de mapeo `color_a_valor` para traducir cada color en su dígito significativo o exponente de multiplicación. Con ello construye el valor base de dos dígitos y aplica la potencia de diez adecuada, obteniendo finalmente el valor de la resistencia en ohmios.

Finalmente, la función `procesar_resistencia()` integra todo el flujo: crea la máscara inversa del fondo azul, extrae y redimensiona la resistencia, detecta y recorta las bandas, identifica los colores y calcula el valor numérico. En el bucle principal, recorremos las 10 imágenes “R{i}_a_out.jpg” generadas anteriormente y para cada una imprimimos:

```
Resistencia R2_a_out.jpg:  
Banda 1: Naranja  
Banda 2: Blanco  
Banda 3: Verde  
Valor: 3900000 Ω
```

Conclusión

Se cumplió exitosamente la primera etapa: convertir cada imagen a espacio HSV, filtrar el rectángulo azul y aplicar operaciones morfológicas para extraer sus cuatro vértices; luego, mediante reordenamiento de puntos y cálculo de homografía, se obtuvo una vista superior uniforme de las 40 fotos. En la segunda etapa, se emplearon técnicas similares de enmascaramiento y morfología para aislar la resistencia dentro de ese rectángulo.

Para la segunda etapa, la detección de las bandas y la clasificación de colores no fue plenamente satisfactoria en todos los casos. En varias imágenes, la segmentación generó bordes faltantes o adicionales lo que provocó errores en el recorte de bandas y, por ende, en la determinación del color.

Conclusión final

En conjunto, en este trabajo se muestra cómo un mismo enfoque de procesamiento de imágenes (basado en filtrado, morfología y detección de contornos) puede adaptarse a dos casos prácticos distintos: la identificación de componentes en una PCB y la extracción del código de color de resistencias.

En el primer caso, se logró aislar y clasificar cada elemento (resistencias, chips y capacitores) validando que las etapas de preprocessamiento generan mapas de contorno suficientemente nítidos para la agrupación mediante componentes conectados.

En el segundo caso, se consiguió recortar y corregir la perspectiva de las resistencias para uniformizar las tomas desde cuatro ángulos distintos. Luego, en algunas imágenes se pudo segmentar las bandas de las resistencias, detectar sus colores y valor.