

Procesamiento de imágenes 1

Trabajo Práctico n° 2

Integrantes:

- Lucía Jauregui J-0627/1
- Norberto Narvaez N-1283/1
- Nahir Sol Vera V-3139/9

Fecha de entrega:

01/12/2025

Introducción

En este trabajo se abordan dos problemas, el primer problema consiste en la detección de monedas y dados en una imagen, se aplicaron transformaciones, filtrado morfológico y algoritmos de reconocimiento de formas. El objetivo es identificar objetos circulares (monedas), separarlos del fondo y contabilizarlos, así como detectar la presencia y posición de dados. El segundo problema se centra en la detección automática de patentes vehiculares. Se trabajan técnicas para localizar la zona de la patente dentro de la imagen, y finalmente segmentar cada carácter.

En conjunto, ambos problemas permiten integrar los conceptos fundamentales de procesamiento de imágenes, desde el preprocesamiento básico hasta la extracción de información útil, aplicando algoritmos robustos para la detección de objetos en situaciones reales.

1º Problema: Detección y clasificación de monedas y dados.

El objetivo principal del primer problema fue detectar monedas y dados en una imagen, clasificarlos según sus características geométricas y calcular la suma total de los puntos de los dados. Para ello se aplicaron múltiples etapas de preprocesamiento, filtrado morfológico, transformaciones de color, y detección mediante HoughCircles, factores de forma y estadísticas de componentes conectadas.

En primer lugar como la imagen original viene en color (BGR), antes de hacer cualquier análisis tuvimos que pasarla a escala de grises, para ello hicimos la función preprocesar_imagen, la cual toma como parámetro una imagen en BGR y así simplifica la imagen a solo un canal en vez de tres. Esta función devuelve la imagen ya en escala de grises con un filtro Gaussiano para suavizar la imagen (*ver figura 1*).

Figura 1



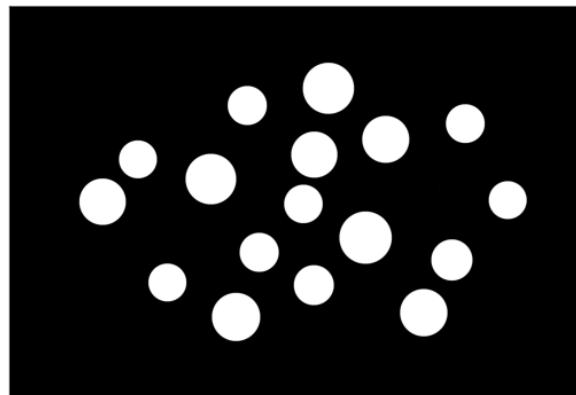
Luego creamos la función detectar_monedas, la cual toma una imagen en escala de grises. Utilizamos la Transformada de Hough para detectar monedas en la imagen. El método Hough Gradient analiza los gradientes de intensidad y guarda evidencia de bordes circulares. Los parámetros fueron ajustados para detectar únicamente círculos dentro del rango de tamaños correspondiente a las monedas de la imagen, evitando errores mediante la restricción del radio y una distancia mínima entre centros. El resultado es un conjunto de coordenadas (x, y, r) que representan el centro y el radio de cada moneda detectada.

Posteriormente implementamos la función dibujar_círculos que toma las coordenadas de los círculos detectados mediante la Transformada de Hough y dibuja su ubicación sobre dos imágenes distintas. En la primera, denominada `imagen_final`, se dibuja cada moneda con un contorno rojo con el objetivo de visualización (*ver figura 2*). En la segunda (`imagen_monedas`) se dibujan círculos llenos de color negro que sirven para segmentar las monedas y eliminarlas del fondo (*ver figura 3*), permitiendo un posterior análisis. La función devuelve ambas imágenes para ser utilizadas en las siguientes etapas del procesamiento.

Figura 2



Figura 3



La función filtrar_componentes toma la salida de `cv2.connectedComponentsWithStats` y elimina los componentes pequeños, para quedarte solamente con los objetos relevantes. Renumera las etiquetas para que queden ordenadas, y devuelve solo los objetos que nos interesan. El parámetro de `area_minima` sirve para filtrar ruido y quedarte solo con los objetos verdaderos.

La función clasificar_monedas recibe las estadísticas generadas por `connectedComponentsWithStats` y utiliza únicamente el área de cada componente para asignarla a un tipo de moneda. Dado que cada moneda tiene un tamaño característico en la imagen, se definen rangos de área para diferenciar monedas de \$0.10, \$1 y \$0.50. Cada componente cuyo tamaño cae dentro de uno de estos intervalos incrementa el contador correspondiente. Al finalizar, la función retorna la cantidad detectada de cada tipo de moneda.

La función procesar_imagen_para_deteccion realiza un preprocesamiento para facilitar la detección de datos. Primero transforma la imagen al espacio HSV y aplica un suavizado para la iluminación. Luego ajusta manualmente los canales H, S y V para resaltar las regiones de interés. Posteriormente, se convierte la imagen nuevamente a escala de grises y se aplica un umbral binario. Finalmente se aplican operaciones morfológicas de cierre y apertura que unifican los objetos grandes (dados) y eliminan restos pequeños. El resultado es una imagen limpia.

Para la detección de las caras de los dados utilizamos el Factor de Forma (Fp), una medida que permite diferenciar objetos sin depender del tamaño. Para ello creamos la función detectar_figuras_por_factor_forma, que recibe una imagen procesada y los límites del rango esperado para identificar correctamente las caras de los dados. En primer lugar obtiene los bordes de la imagen utilizando el detector de Canny y luego extrae los contornos externos mediante cv2.findContours. Para cada contorno se calcula su área (cv2.contourArea) y su perímetro (cv2.arcLength). Finalmente, los contornos cuyo Fp se encuentra dentro del rango definido son considerados candidatos a representar caras de dados.

Luego, para la detección del valor de cada dado, creamos la función contar_círculos, a la cual se le ingresa como parámetros los contornos de los dados detectados por la etapa anterior y la imagen sobre la que se trabaja.

Para cada dado identificado mediante el factor de forma, se recorta la región correspondiente dentro de la imagen original. Sobre cada recorte se aplica nuevamente un filtrado suavizado, a modo de reducir ruido y facilitar la detección de puntos. Se emplea la Transformada de Hough para círculos, la cual permite identificar los puntos característicos de la cara del dado. La cantidad de círculos detectados en cada recorte corresponde directamente al valor del dado. El puntaje individual de cada dado se guarda en un diccionario que asocia “Dado N” con la cantidad de puntos detectados, y además se acumula en una variable que representa el puntaje total obtenido en la imagen.

A lo largo de la resolución del enunciado nos enfrentamos a una serie de problemáticas, alguna de ellas fueron:

- **Componentes conectados muy pequeños o irrelevantes:** aparecían componentes conectadas muy pequeñas (*ver figura 4*), que no eran monedas ni dados (manchas). Lo que nos obligó hacer un filtrado por área.

Figura 4



- **Dificultad para diferenciar cada moneda:** la única diferencia real entre las monedas es su área por lo que usamos ese criterio para clasificarlas. Para poder poner correctamente los umbrales de cada moneda fuimos imprimiendo por pantalla el área de los componentes que se detectaron como válidos y así fuimos probando cuál era el umbral correcto para cada moneda (*figura 5*). Concluimos que las monedas con área menor a 7000 pixeles son de 10 centavos. Sus áreas medidas en la impresión entran en este rango. Monedas entre 7000 y 9000 pixeles son de 1 peso y por último las monedas de 50 centavos entran en el rango de 9000 y 12000 pixeles.

Figura 5

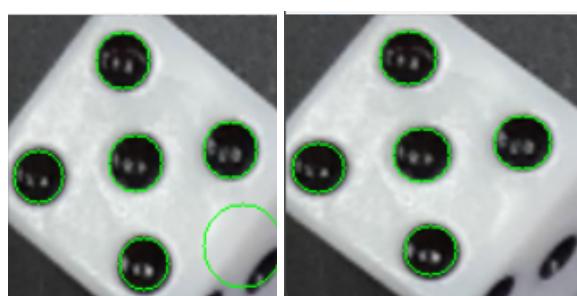
```

el area de la moneda es 5583
el area de la moneda es 8188
el area de la moneda es 7885
el area de la moneda es 5334
el area de la moneda es 9449
el area de la moneda es 8003
el area de la moneda es 5332
el area de la moneda es 5500
el area de la moneda es 10128
el area de la moneda es 5665
el area de la moneda es 6285
el area de la moneda es 5322
el area de la moneda es 5932
el area de la moneda es 8300
el area de la moneda es 8604

```

- **Selección del umbral por área:** para determinar qué componentes conectados pertenecían realmente a las caras de los dados, se imprimieron las áreas de todas las componentes detectadas. Esto permitió observar empíricamente la distribución de tamaños presente en las imágenes. A partir de ese análisis se identificó que el ruido presentaba áreas muy inferiores a 500 píxeles, mientras que las regiones correspondientes a los dados tenían áreas mayores. Este valor permite filtrar el ruido sin perder ninguna región válida. Como anteriormente nombramos, los umbrales se ajustaban manualmente y eran muy sensibles, en la figura 6 y 7 podemos ver que al ajustar los parámetros de HoughCircles, en particular el parámetro 2 que es el umbral del acumulador para los centros de los círculos en la etapa de detección, en la primer imagen (*ver figura 6*) estaba en 22 y en la segunda imagen (*figura 7*) lo cambiamos a 25 para que no detecte círculos falsos.

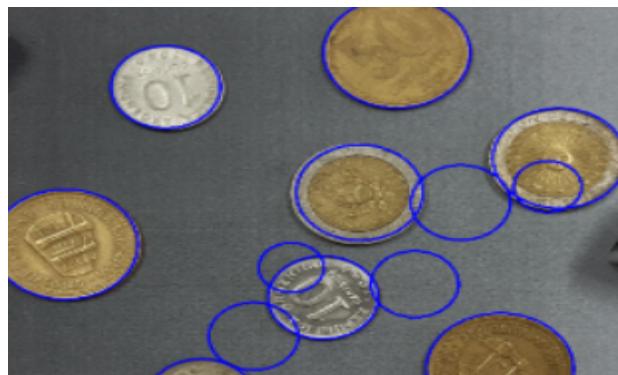
Figura 6 y 7



- **Parámetros muy sensibles en HoughCircles:** la Transformada de Hough requiere ajustar los parámetros, si uno solo de esos valores no es adecuado aparecen círculos demás. Por eso se tuvo que ajustar mediante prueba y error hasta llegar a los umbrales adecuados. En la figura 8 se detectaban círculos

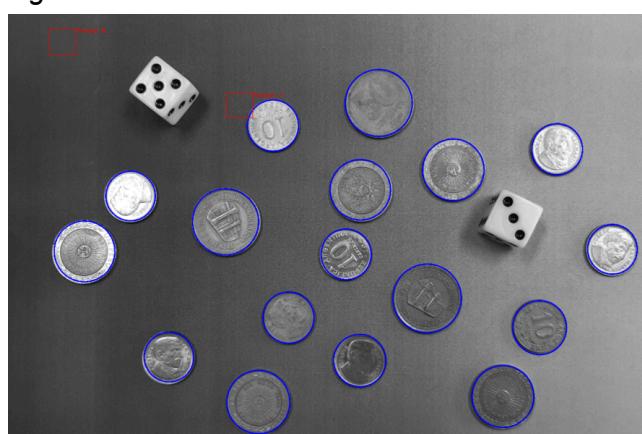
incorrectos por lo que tuvimos que ir cambiando los parámetros de la función para llegar a su forma correcta. En este caso detectaba círculos demás ya que la distancia mínima entre dos círculos era poca (20), luego ajustamos hasta que detecte las monedas correctamente (90).

Figura 8



- **Errores por redimensionamiento:** como la imagen fue reescalada en varias etapas produjo que las coordenadas de los contornos no coincidieran con la imagen original. La solución implementada fue aplicar el mismo factor de reescalado a las coordenadas del contorno obtenido en la imagen reescalada, o evitar el reescalado intermedio y trabajar con la imagen original o la imagen en escala de grises con suavizado, para asegurar que las coordenadas coincidan con la imagen final. En la figura 9 podemos observar que no volvimos a dimensionar de forma correcta, por eso los cuadrados que encierran los dados se encuentran corridos.

Figura 9



Conclusión del problema 1:

En primer lugar nos centramos en el preprocesamiento de la imagen, luego avanzamos con la detección y clasificación de las monedas, y finalmente abordamos la identificación de los dados y el recuento de sus valores. El preprocesamiento resultó ser una etapa fundamental. Allí aplicamos filtros, conversión a HSV, umbralizado, análisis de componentes conectados y operaciones morfológicas para obtener una imagen limpia. Esta etapa permitió reducir ruido, mejorar el contraste y aislar las regiones relevantes.

Luego nos enfocamos en las monedas. Gracias a la Transformada de Hough y al análisis del área de cada componente conectado, logramos detectar los círculos y clasificar las

monedas según su tamaño. Finalmente, pasamos a los dados. Utilizamos el factor de forma para distinguir formas cuadradas del resto de los objetos, y luego aplicamos nuevamente Hough sobre cada recorte para contar los puntos presentes. La combinación de contornos, el factor forma y detección de círculos nos permitió obtener el valor de cada dado y sumar el puntaje total. En la figura 10 podemos observar los resultados que obtuvimos luego de todos los pasos nombrados anteriormente.

Figura 10



Los resultados que obtuvimos fueron los siguientes:

Cantidad \$1: 5

Cantidad \$0.50: 3

Cantidad \$0.10: 9

Suma total de los dados: 8

2º Problema: Detección de patentes.

En el segundo problema nos encontramos con una serie de imágenes de autos, con sus respectivas patentes, el objetivo del problema fue crear un algoritmo capaz de segmentar cada patente de manera automática y que a su vez descomponga cada patente en sus componentes (6).

En primer lugar, al igual que en el primer problema nos centramos en trabajar la imagen en escala de grises para facilitar su procesamiento (ver figura 11).

Figura 11

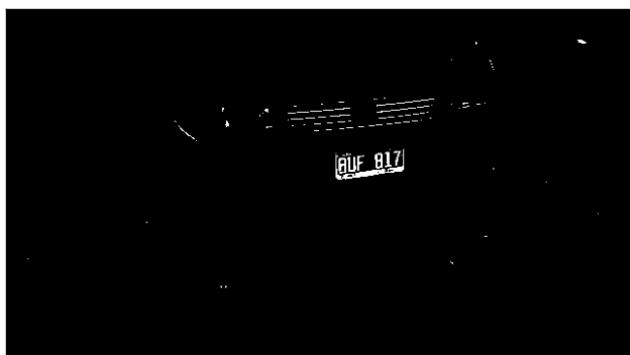


A partir de la imagen en escala de grises, armamos una función que realiza un preprocesamiento morfológico y binarización, prepara la imagen para que el texto sea visible y separable del fondo. Aplica la transformada Top-Hat usando un kernel rectangular, lo que elimina las variaciones de luz. La imagen realizada se normaliza y se convierte a blanco y negro mediante un umbral. El resultado es una imagen transformada (ver figura 12) y luego binarizada (ver figura 13).

Figura 12



Figura 13



Luego creamos la función filtrar_componentes que valida y agrupa los objetos detectados por su forma y posición relativa. Para ello se establecieron límites mínimos y máximos para la razón de aspecto (aspect_ratio), que es la relación entre la altura y el ancho del objeto, y para el área total del componente. En un primer momento, la función itera sobre todas las estadísticas de los componentes conectados recibidas. Por cada componente, calcula su razón de aspecto (h / w). Si la forma y el tamaño del componente caen dentro de los rangos predefinidos, se clasifica como un candidato a ser un carácter y se añade a una lista. Este primer filtro elimina de manera eficiente el ruido. Luego, aplica una heurística de alineación y proximidad para confirmar que los candidatos forman parte de una palabra. Utilizando un doble bucle, compara cada candidato con todos los demás. Esta comparación se basa en calcular la distancia horizontal y la distancia vertical entre los puntos de inicio de los dos componentes. Se considera que son parte de la misma palabra si la distancia horizontal (dist_x) es pequeña y, crucialmente, la distancia vertical (dist_y) es mínima, lo que implica que están alineados en la misma línea base de texto. Si cumplen esta condición, ambos componentes se añaden a la lista de caracteres. Finalmente, dado que el proceso de alineación puede generar muchos duplicados, la función realiza una limpieza final eliminando los duplicados. En la figura 14 podemos observar como quedaría la segmentación de la patente sin esta limpieza final.

Figura 14



A su vez creamos la función detector_objetos que utiliza la herramienta de OpenCV connectedComponentsWithStats aplicándola sobre la imagen binaria. En este paso estamos identificando y etiquetando todos los grupos de píxeles blancos contiguos. Esta función devuelve una matriz completa de estadísticas que contiene toda la información geométrica de cada objeto detectado: su posición, ancho, alto y área total. Una vez obtenidas estas estadísticas crudas, la función llama inmediatamente a filtrar_componentes. Le transfiere la matriz de estadísticas, delegándole la tarea de aplicar todas las reglas de validación necesarias (forma, tamaño y alineación horizontal). Finalmente, la función devuelve la lista filtrada y depurada de los componentes que han sido confirmados como caracteres válidos, quedando listos para el paso de recorte y segmentación final.

Por último la función `recortar_letras` representa el paso final y de visualización del proceso de segmentación de caracteres. Su objetivo principal es tomar las estadísticas validadas de los caracteres detectados y usarlas para aislar cada letra o número. El proceso se inicia tomando la imagen original en escala de grises y convirtiéndola a color. La función itera sobre la lista, que contiene las coordenadas y dimensiones de cada carácter que superó los filtros. Utilizando esta información, la función aplica rectángulos delimitadores verdes sobre la imagen de color. Posteriormente procede a la segmentación propiamente dicha. Antes de recortar, la función es crucial al ordenar los caracteres de izquierda a derecha utilizando su coordenada inicial, garantizando así que los recortes se muestren en el orden correcto de lectura de la patente. Luego, la función recorre esta lista ordenada y utiliza las coordenadas de cada componente para recortarlo individualmente. Luego los recortes se presentan individualmente en una secuencia de figuras (ver figura 15), logrando la segmentación completa y aislando cada carácter como una pequeña imagen independiente.

Figura 15



La función ejecutar es de control principal del programa. Esta función establece un bucle de procesamiento que carga secuencialmente las imágenes del conjunto. Por cada imagen cargada, realiza la conversión inicial a escala de grises. A continuación, la imagen en grises se procesa mediante la función img_procesada para realizar el realce de contraste y la binarización, aislando el texto blanco en la imagen binaria. El resultado binario alimenta

a la función `detector_objetos`, que inicia la fase de detección, utiliza `cv2.connectedComponentsWithStats` y luego aplica el filtrado heurístico (`filtrar_componentes`) para validar qué componentes son realmente caracteres por su forma y alineación. Finalmente, la lista de caracteres validados se pasa a `recortar_letras`, que completa la tarea: dibuja los recuadros de localización, ordena los caracteres de izquierda a derecha y recorta y muestra cada uno individualmente, logrando la segmentación final del texto.

A lo largo de la resolución del enunciado nos enfrentamos a una serie de problemáticas, alguna de ellas fueron:

- **Encontrar un punto estable para la correcta binarización:** el punto crítico en el preprocesamiento fue hallar un umbral que permitiera la correcta binarización, es decir, convertir la imagen a blanco y negro sin perder información. La dificultad principal reside en que la iluminación y el brillo de la patente varían en cada foto. La solución eficaz fue: aplicar la transformada Top-Hat, esta operación elimina el fondo no uniforme y hace que el texto resalte con un brillo consistente, independientemente de la iluminación ambiental. También la umbralización Post-Realce, una vez que el texto tiene un brillo uniforme gracias al Top-Hat, la aplicación de un umbral fijo o un umbral adaptativo se vuelve mucho más estable y robusta, asegurando que los caracteres no se fragmenten ni se fusionen al binarizarse.
- **Evolucion en la forma de encontrar la patente:** inicialmente, nos enfocamos en la forma rectangular de la placa lo cual presentaba problemas. En la figura 15 podemos observar que detectaba rectángulos de manera incorrecta lo cual resultó difícil resolver la detección de la patente mediante esta forma.

Figura 16



La solución más efectiva fue la detección basada en la textura interna del texto, combinando la forma con las características de las letras.

- **La segmentación de caracteres sin homografía:** al principio, como cada imagen estaba sacada desde una perspectiva distinta, estábamos con la idea de que sin homografía no podríamos detectar los caracteres de las patentes, pensábamos que era necesario corregir la perspectiva de la patente y luego segmentar, lo cual fue complejo. Gracias a las heurísticas geométricas pudimos detectar cada carácter sin homografía, en lugar de corregir la perspectiva, se utilizó la función `filtrar_componentes` para imponer reglas geométricas locales: un objeto debe tener la forma de una letra (razón de aspecto) y debe estar alineado verticalmente con otro objeto cercano.

- **Duplicados en los caracteres filtrados:** en la figura 14 podemos observar que se repetían los caracteres de las placas, por lo cual tuvimos que hacer un filtrado para sacar los duplicados de la lista y así garantizar que cada carácter válido fuera contabilizado una sola vez.

Conclusión del problema 2:

Se logró el objetivo de detección y segmentación de caracteres de cada patente de la mayoría de las imágenes analizadas. Se consiguió mediante una serie de pasos de procesamiento de imágenes: primero, se aplicó un preprocesamiento robusto que incluyó la transformada Top-Hat para aislar y realzar el texto sobre el fondo. Luego, se implementó un Filtrado Heurístico que combinó la validación de forma y tamaño (razón de aspecto y área) con una heurística de alineación. Finalmente, la Segmentación Final tomó los componentes validados, los ordenó y los aisló individualmente, transformando la placa completa en una secuencia de imágenes. Podemos observar los resultados de este proceso en la figura 17 una de las 12 imágenes que fueron analizadas, en este caso vemos a la imagen 6. En la figura 18 podemos observar la segmentación de los caracteres de la patente de la imagen.

Figura 17

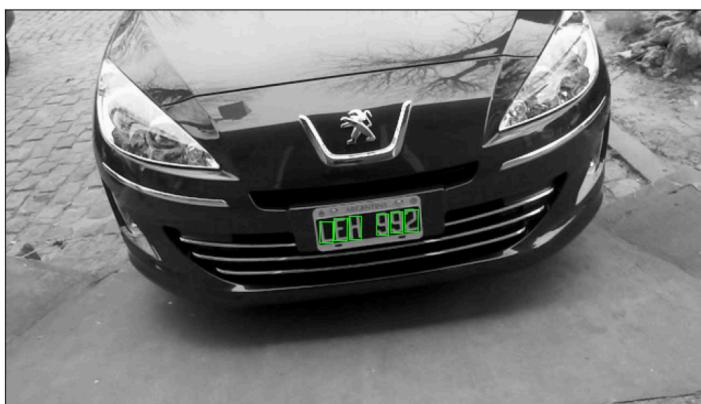
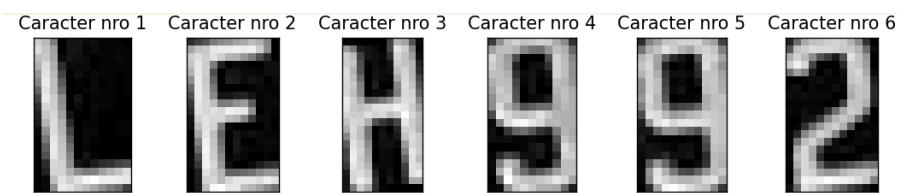


Figura 18



Conclusión final:

El desarrollo de este trabajo permitió aplicar de manera integrada diversos conceptos y técnicas del Procesamiento Digital de Imágenes, abordando dos problemas con características y desafíos distintos. En conjunto, los resultados obtenidos muestran que el correcto preprocesamiento de las imágenes es un proceso fundamental, al igual que la combinación adecuada de operaciones morfológicas y segmentación. El trabajo nos llevó a enfrentarnos a casos reales y comprender la importancia de seleccionar correctamente los parámetros y algoritmos para cada situación concreta. Llegamos a la conclusión de que el procesamiento previo de la imagen es un paso muy importante: la iluminación, el ruido y la

variación del fondo influyen directamente en la segmentación y, por lo tanto, en la clasificación posterior. Otro componente muy importante fue el manejo correcto de los umbrales y parámetros, ya que en el problema se manejaban umbrales muy sensibles, los cuales tuvimos que experimentar mediante prueba y error.