Facultad de Ciencias Exactas, Ingeniería y Agrimensura

Trabajo práctico N.º 01

Tecnicatura Universitaria en Inteligencia Artificial

Procesamiento de imágenes (IA4.4)

Fecha: 21/10/2025

Integrantes:

Nombre:	Legajo:	Mail:
Ezequiel Zahradnicek	Z-1214/9	ezeonico@gmail.com

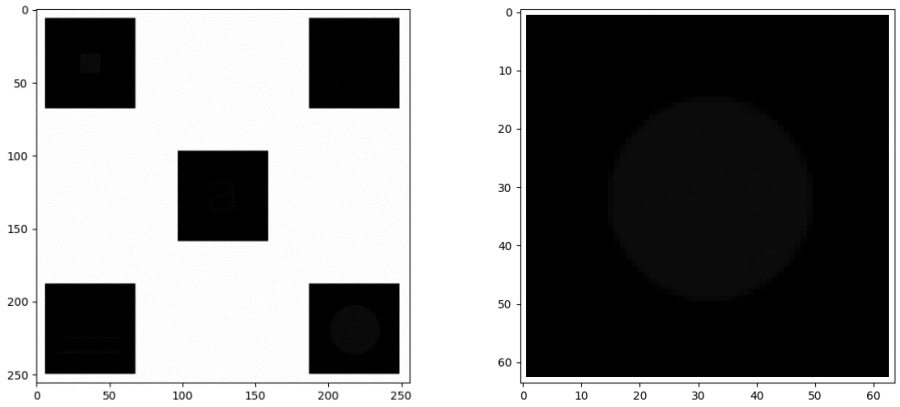
Introducción:

El presente trabajo práctico tiene como objetivo aplicar diferentes técnicas de **procesamiento digital de imágenes** para resolver dos problemas específicos, enfocados en aspectos esenciales del análisis, mejora y validación de información visual.

Este informe presenta el desarrollo de las actividades realizadas, la descripción de los métodos implementados y las dificultades encontradas durante el proceso, acompañadas de capturas y ejemplos que ilustran cada etapa del procedimiento.

Problema 1 - Ecualización local de histograma:

La imagen proporcionada (*Imagen_con_objetos_ocultos.tiff*) presenta cinco regiones principales de baja intensidad ubicadas en las cuatro esquinas y en el centro del plano. A simple vista, estas zonas se perciben como rectángulos de color negro uniforme, con valores de intensidad cercanos a cero.



Sin embargo, al observar la imagen con detenimiento, es posible distinguir ciertas formas en el interior de los rectángulos.

Si analizamos los valores únicos de la imagen, se observa que las estructuras internas presentan intensidades de píxeles que varían aproximadamente entre **1 y 11**, mientras que el fondo general mantiene valores más elevados, comprendidos entre **226 y 228**.

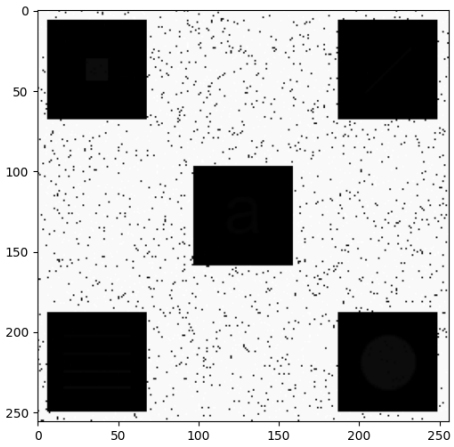
```
>>> np.unique(img)

array([  0,   1,   2,   3,   4,   5,   6,   7,   8,   9,  10,  11, 226, 227,
228], dtype=uint8)
```

Aunque a simple vista el fondo parece uniforme, esta ligera variación en los niveles de gris indica que no se trata de una región completamente homogénea.

Por lo tanto, un análisis básico a priori nos indicaría que los rectángulos negros contienen información visual oculta en forma de líneas, figuras o letras con intensidades bajas, mientras que el fondo restante presenta ligeras variaciones en sus valores de píxel.

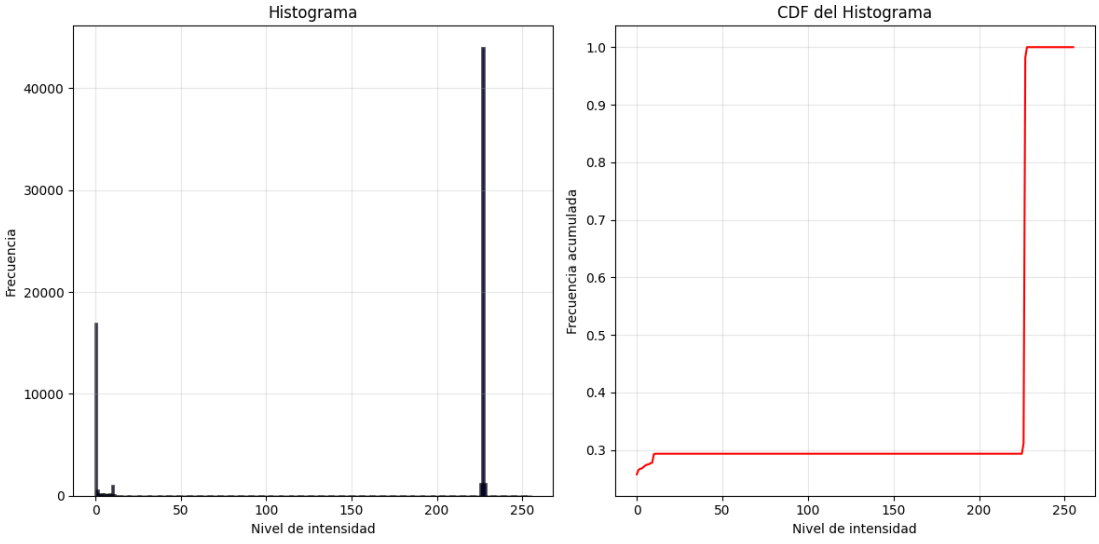
Pero podría ser útil preguntarnos primero: **¿por qué deberíamos usar una ecualización local del histograma en lugar de una global?** Para responder a esto, consideremos qué sucede si aplicamos la ecualización global sobre nuestra imagen:



Al observar la imagen resultante, notamos que los detalles internos siguen siendo poco perceptibles. Al mismo tiempo, se observa la aparición de puntos negros dispersos sobre el fondo, que generan un efecto de ruido visual.

Esto indica que la ecualización global **no logra resaltar adecuadamente los detalles de las regiones oscuras** y, al mismo tiempo, **introduce variaciones indeseadas en áreas que originalmente eran aparentemente uniformes.**

Para entender por qué ocurre esto, es útil analizar el histograma de la imagen y su correspondiente función de distribución acumulativa:



En esta imagen donde los píxeles de los cuadrados oscuros representan una proporción muy pequeña del total y el fondo domina el histograma, **la CDF global se ve prácticamente determinada por el fondo.**

Como consecuencia, los detalles en los cuadrados se comprimen en la parte baja del histograma y se mantienen poco visibles, mientras que el fondo, que debería ser uniforme, se ve afectado por pequeñas variaciones, generando el ruido observado.

Por ejemplo, si analizamos el histograma, podemos observar que el fondo está prácticamente compuesto por píxeles con valor 227, mientras que los valores 226 y 228 representan un porcentaje muy pequeño del mismo. Si quisiéramos determinar a qué valores se mapearán estas intensidades y **comprender así el origen del ruido**, podríamos calcularlo de la misma manera que OpenCV lo hace internamente, de la siguiente forma:

$$I_{nuevo} = round(\frac{CDF(I) - CDF_{min}}{CDF_{max} - CDF_{min}} \cdot 255)$$

Por ejemplo, si tomamos el valor acumulado correspondiente a 226 y consideramos el valor mínimo de la CDF, podemos calcular el valor que adoptará este píxel en la imagen ecualizada.

```
>>> cdf_normalized.min()
0.2574462890625
>>> cdf_normalized[226]
0.3112030029296875
```

Por lo tanto:

$$CDF(226) - CDF_{min} = 0.3112030 - 0.2574463 = 0.0537567$$

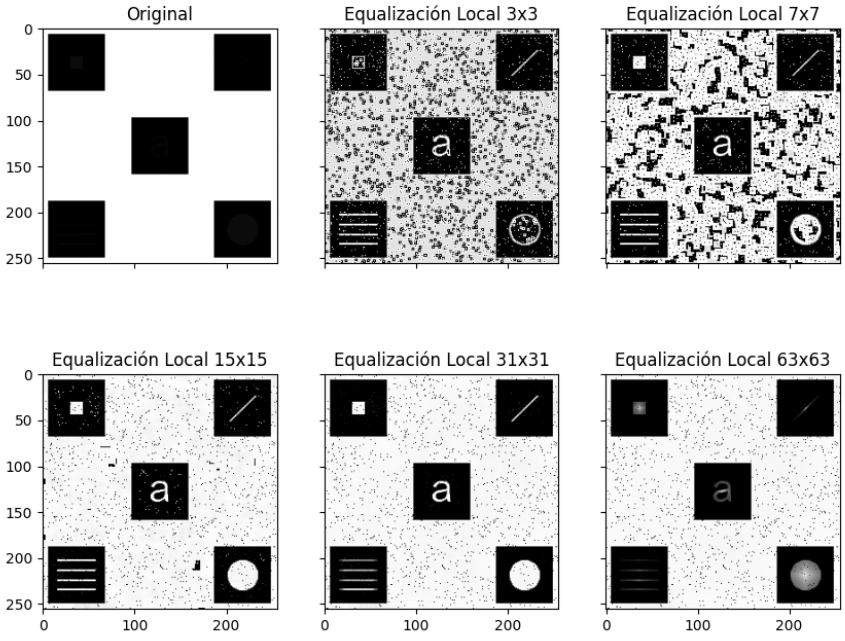
$$\frac{0.0537567}{1 - 0.2574463} = \frac{0.0537567}{0.7425537} \approx 0.0724$$

$$0.0724 \cdot 255 \approx 18.45$$

Estos son precisamente los valores que adopta el píxel 226 en la imagen ecualizada. Una vez comprendido esto, podemos apreciar la importancia de utilizar una ecualización local.

La función implementada para la resolución de este problema, denominada **ecualización local**, recibe como parámetros la imagen a procesar, un tamaño de ventana representado por una tupla de dos enteros y un parámetro opcional *background*, que por defecto se establece en False.

Para preparar la imagen para la ecualización local, se calcula un borde que se agrega a todos los lados de la imagen. Este borde se define tomando la parte entera del máximo entre el alto y el ancho de la ventana, dividido por dos. La razón de tomar el máximo entre el ancho y el alto de la ventana es para considerar el caso de ventanas no cuadradas. A continuación, se itera sobre cada fila y columna de la imagen, es decir, sobre cada píxel de la imagen con bordes. Para cada posición, se extrae una ventana centrada en el píxel actual, se aplica ecualización de histograma a dicha ventana y, finalmente, el valor del píxel central de la ventana ecualizada reemplaza al píxel correspondiente en la imagen de resultado. Este procedimiento se repite para todos los píxeles, logrando un ajuste localizado del contraste que preserva los detalles de cada región.



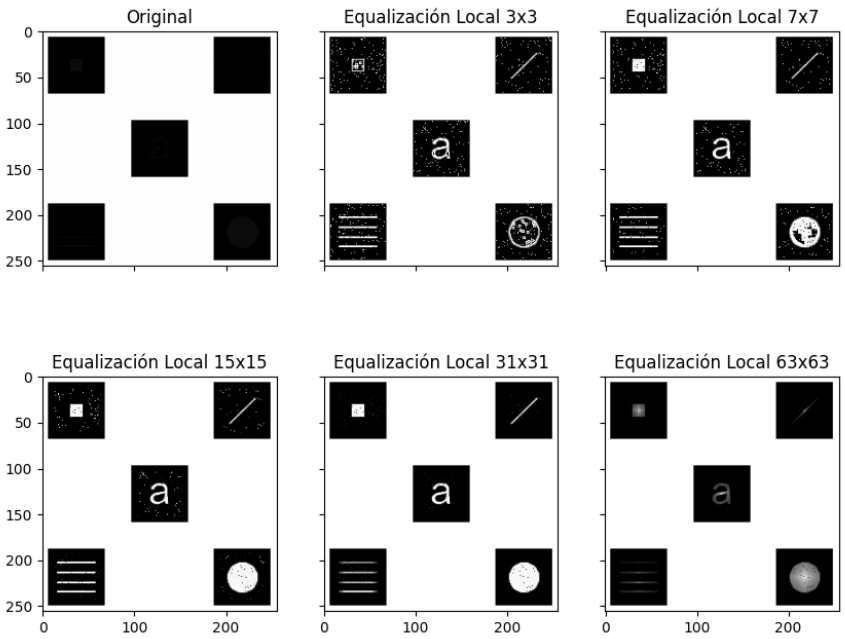
Como podemos concluir, los tamaños de ventana más pequeños, como 3x3 o 7x7, son muy sensibles a pequeñas variaciones en el fondo o en los detalles finos de la imagen. Esto se debe a que, al tener pocos valores en el histograma, su CDF presenta cambios abruptos, generando valores muy dispares.

A medida que se incrementa el tamaño de la ventana, el ruido de fondo disminuye, aunque nunca desaparece completamente.

Con ventanas de tamaño intermedio, como 15x15, aún pueden observarse remanentes de artefactos o pequeñas agrupaciones de píxeles con formas de cuadrados o rectángulos en el fondo. A partir de 31x31, estos artefactos exagerados desaparecen y el fondo presenta un ruido más uniforme, similar a un ruido de tipo pimienta, mientras que los detalles finos de la imagen siguen siendo perceptibles y bien definidos.

Por su parte, ventanas muy grandes, como 63x63, producen un efecto más global, donde el contraste de la imagen se ajusta sobre áreas mayores, reduciendo la exageración de los artefactos, pero los detalles finos comienzan a perder definición y se vuelven menos perceptibles.

Adicionalmente, si el parámetro *background* se establece en True, se aplica una máscara que uniforma el fondo de la imagen, evitando que pequeñas variaciones en los valores del fondo generen ruido en la ecualización; esta consideración es relevante para nuestro caso particular, dado que el fondo está compuesto por tres valores (226, 227 y 228).



Problema 2 - Validación de formulario:

En el **Problema 2**, se plantea la validación automática de los campos presentes en un conjunto de formularios provistos en formato de imagen. Cada formulario contiene distintos campos que deben ser verificados según ciertas **restricciones** que determinan si los datos fueron ingresados de manera **válida o inválida**.

El proceso consiste en analizar cada formulario y mostrar por consola qué campos cumplen con las condiciones establecidas y cuáles no. Además, se debe generar una **imagen de salida** que contenga un recorte (crop) correspondiente al nombre y apellido de la persona, acompañado de un indicador visual que refleje si el ingreso de esos datos es correcto o incorrecto.

Finalmente, el sistema debe crear un **archivo CSV** que registre, para todos los formularios procesados, el estado de validación de cada uno de sus campos, especificando cuáles fueron ingresados correctamente y cuáles no.

FORMULARIO A		
Nombre y apellido		
Edad		
Mail		
Legajo		
	Si	No
Pregunta 1		
Pregunta 2		
Pregunta 3		
Comentarios		

Para resolver este ejercicio, se propone seguir una serie de pasos que permiten abordar correctamente el proceso de validación de los formularios:

1. **Detectar las posiciones** (índices) en la imagen que correspondan a las filas **y columnas** donde se ubican los distintos campos del formulario.
2. **Recortar los campos de interés**, es decir, generar los recortes (crop) de las zonas específicas que se desean analizar.
3. **Analizar el contenido de cada campo**, contando la cantidad de caracteres presentes y detectando espacios.
4. **Verificar las respuestas** de cada pregunta, comprobando si fueron ingresadas de forma correcta o incorrecta según los criterios establecidos.
5. **Generar una imagen de salida** que contenga el recorte correspondiente al nombre y apellido de la persona que completó el formulario, junto con un indicador visual que muestre si los datos fueron ingresados correctamente o no.

1. Detectar las posiciones:

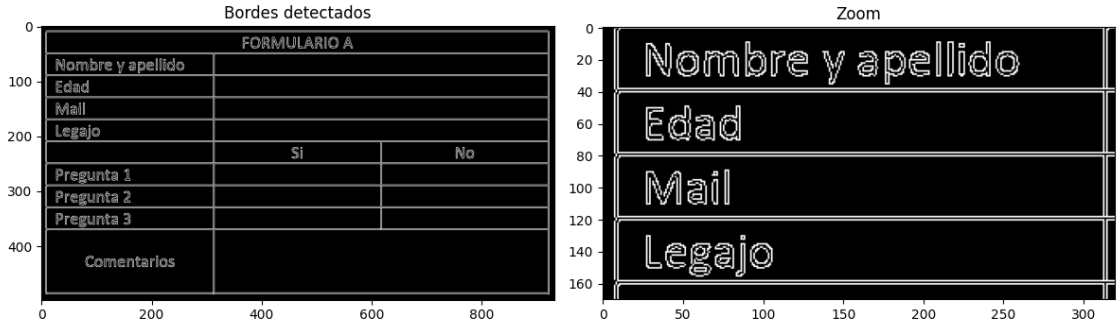
La idea general para resolver este problema consiste en aplicar un algoritmo de detección de bordes, en este caso el método de Canny, con el objetivo de identificar las zonas de la imagen donde se producen cambios fuertes de intensidad, que pueden interpretarse como bordes del formulario.

A partir de la imagen binaria obtenida, se realiza un análisis fila por fila o columna por columna, sumando los valores de los píxeles blancos en cada una de ellas. De esta forma, las filas o columnas que presentan mayores sumas corresponden a regiones con una alta densidad de bordes, lo que permite determinar la posición de las líneas que delimitan los distintos campos del formulario.

Para implementar este procedimiento se propone la función indice_bordes, que recibe como parámetros la imagen a analizar, el eje sobre el cual se trabajará (0 para filas y 1 para columnas), así como los parametros candidates y diff, que se explicarán posteriormente.

Esta función comienza aplicando el algoritmo de Canny, con el objetivo de resaltar los bordes presentes en la imagen y facilitar la identificación de las regiones de interés.

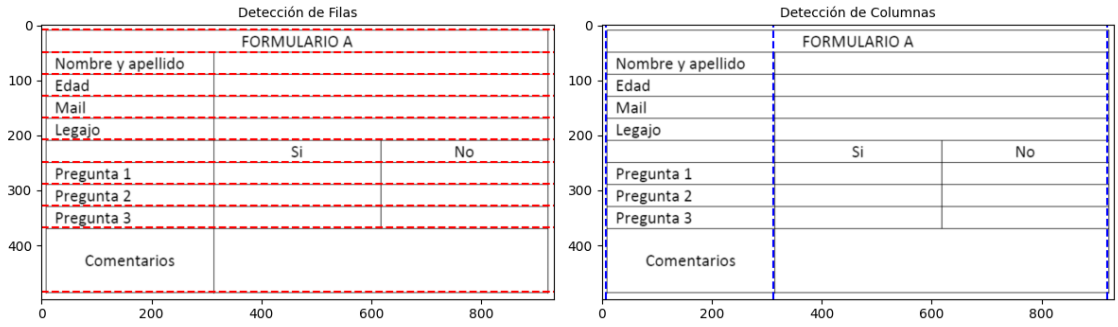
Podemos observar que Canny detecta correctamente tanto las filas como las columnas del formulario, así como otras áreas de gran variación en la imagen, como los caracteres de texto. A partir de esto, es posible realizar un análisis sumando los píxeles blancos fila por fila o columna por columna, de manera que se identifiquen las regiones con mayor variación y, por lo tanto, se detecten correctamente las filas y columnas. Sin embargo, es importante considerar que el algoritmo de Canny tiende a marcar tanto el inicio como el final de una fila o columna; es decir, genera dos índices para la misma línea, uno al comienzo y otro al final.



Para consolidar estos resultados y obtener un solo índice representativo por fila o columna, se propone evaluar las diferencias entre los índices detectados. Si dos índices están muy próximos entre sí, se asume que corresponden a la misma fila o columna. De esta forma, se puede controlar la cantidad de filas y columnas a detectar según lo requerido; por ejemplo, en nuestro caso, se buscan identificar diez filas y tres columnas, que son nuestras áreas de interés. El parámetro diff establece un umbral que permite eliminar índices duplicados dentro de esta distancia mínima, conservando siempre el primer índice detectado, que corresponde al inicio de la fila o columna (es decir, el índice más a la izquierda para las columnas o el más alto para las filas) como la posición representativa de dicha línea.

Por último, cabe destacar una consideración específica de nuestro problema relacionada con la detección de columnas. El objetivo era identificar tres columnas para segmentar correctamente el formulario; sin embargo, la presencia de separadores entre los campos de respuesta “sí” y “no” podía generar detecciones adicionales, interpretándose erróneamente como columnas. En estos casos, si el algoritmo detectaba más de tres columnas, se eliminaba siempre la columna ubicada en la tercera posición, considerando los índices de menor a mayor.

Esto se justifica porque, al analizar la imagen original, la primera columna corresponde al límite izquierdo del formulario, la segunda delimita los nombres de los campos, la tercera columna detectada corresponde al separador entre “sí” y “no”, y la cuarta al límite derecho del formulario; por lo tanto, la tercera posición siempre coincide con el delimitador interno.



En principio, se intentó extraer las filas y columnas de un formulario vacío para detectarlas una única vez y garantizar que la detección fuera lo más precisa posible, evitando que los datos ingresados afectaran la identificación de los límites de los campos. Sin embargo, esta estrategia no resultó viable, ya que los formularios, tanto completos como vacíos, presentan tamaños ligeramente diferentes entre sí, lo que impide trasladar los índices de uno a otro.

También, se intentó realizar un redimensionamiento de las imágenes, es decir, ajustar el tamaño del formulario vacío al del formulario que se estaba evaluando en cada iteración, con el objetivo de detectar de manera más precisa las filas y columnas. Sin embargo, a pesar de que ambos formularios tenían la misma cantidad de filas y columnas tras el ajuste, al momento de mostrar los resultados se observó que las filas se detectaban correctamente en el formulario vacío, pero no en el formulario correspondiente a la iteración. Esto evidencia que el simple redimensionamiento no era suficiente para trasladar los índices de manera confiable entre formularios.

2. Recortar campos de interés:

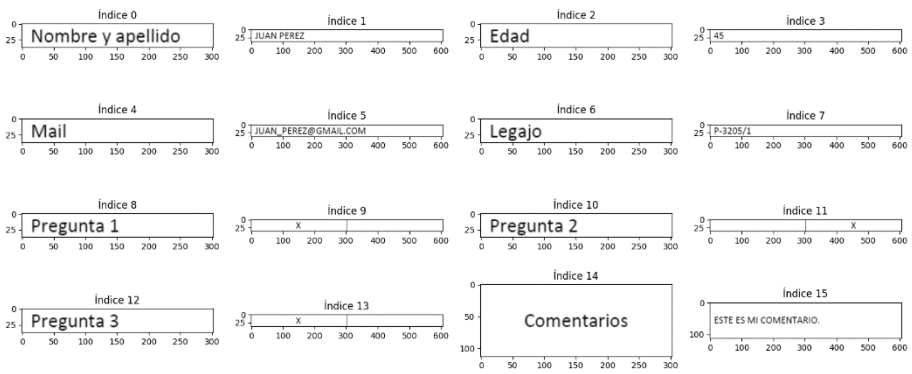
Una vez detectadas las filas y columnas que delimitan las distintas secciones del formulario, se procede al recorte de los campos de interés. El objetivo de este proceso es aislar cada una de las áreas relevantes del formulario para facilitar su análisis individual, reduciendo así la complejidad del procesamiento posterior.

Para ello, se implementó la función recortar_campos, la cual recibe como entrada la imagen original del formulario y los índices correspondientes a las filas y columnas previamente detectadas. El procedimiento consiste en recorrer la imagen iterando sobre cada par consecutivo de filas y columnas, realizando un recorte (*crop*) que abarca desde la posición de la fila actual hasta la siguiente y, de manera análoga, desde la columna actual hasta la siguiente. De esta forma, se obtienen subimágenes que corresponden a los distintos campos del formulario.

Durante este proceso se excluyen deliberadamente ciertas filas que no contienen información relevante para el análisis. En particular, se omiten:

- La primera fila (índice 0), correspondiente al encabezado o título del formulario.
- La sexta fila (índice 5), que contiene los campos de selección “Sí” y “No”.

Estas secciones no aportan datos útiles para el proceso de detección y validación de información, por lo que se descartan. Además, para mejorar la segmentación y evitar que los recortes incluyan bordes o líneas divisorias, se añade un margen de tres píxeles en la operación de corte.



A partir de los recortes obtenidos mediante la función recortar_campos, se observa que cada par consecutivo de subimágenes mantiene una estructura regular. En particular, los recortes con índices **pares** contienen el título o la etiqueta del campo (por ejemplo,

"Nombre y Apellido", "Edad", etc.), mientras que los recortes con índices **impares** contienen la información efectivamente ingresada por el usuario en dichos campos.

Por ejemplo, el recorte con índice 0 corresponde al encabezado "Nombre y Apellido", mientras que el recorte con índice 1 contiene el valor introducido por el usuario en ese campo. De igual forma, el índice 2 contiene el texto "Edad" y el índice 3 el valor correspondiente a dicha entrada. Este patrón se mantiene a lo largo de todo el formulario.

El resultado de esta operación es un conjunto de subimágenes que representan exclusivamente las áreas de interés del formulario, es decir, aquellas que contienen los valores efectivamente proporcionados por el usuario. Estas subimágenes se utilizarán en las etapas posteriores del proceso de validación o análisis automatizado.

3. Analizar el contenido de cada campo:

Una vez aisladas las áreas de interés del formulario, el siguiente paso consiste en analizar el contenido de cada una de ellas. Para ello, se plantean dos enfoques complementarios:

1. **Contar la cantidad de caracteres presentes en cada campo.**
2. **Detectar espacios dentro del texto ingresado.**

3.1 Contar la cantidad de caracteres:

El proceso implementado para contar los caracteres en un campo es el siguiente. En primer lugar, se aplica un umbral binario inverso que transforma el texto a color blanco sobre un fondo negro, lo que facilita la identificación de los distintos caracteres presentes en la imagen. Luego, se aplica un procedimiento de análisis de componentes conectados que permite detectar las regiones formadas por píxeles contiguos pertenecientes a un mismo carácter o símbolo. El resultado de este proceso devuelve la cantidad total de componentes encontrados, donde cada componente se asocia a un carácter presente en el campo. Dado que una de estas etiquetas corresponde al fondo de la imagen, se descarta esa unidad para obtener el número real de caracteres detectados.

Este método resulta efectivo en nuestro caso porque los recortes generados en las etapas anteriores no contienen bordes, líneas divisorias ni otros elementos estructurales del formulario. Es decir, cada recorte conserva únicamente la información relevante ingresada por el usuario, sin interferencias visuales que puedan afectar el conteo de caracteres.

La única excepción se presenta en los campos correspondientes a las preguntas que incluyen un separador. En esos casos, el separador se trata de manera particular mediante una función adicional desarrollada específicamente para el análisis de preguntas, que será descrita más adelante.

3.2 Detectar espacios dentro del texto ingresado:

El siguiente paso en el análisis del contenido consiste en determinar si el texto ingresado en cada campo contiene o no espacios. Para ello, se implementa un procedimiento basado en el análisis de componentes conectados y en la medición de distancias entre caracteres detectados.

En primer lugar, se recibe la imagen correspondiente al campo y se aplica un umbral binario inverso. A partir de esta imagen, se obtienen las componentes conectadas, que representan cada carácter o símbolo presente en el texto.

Una vez identificadas, las componentes conectadas se ordenan según la posición de su borde izquierdo, lo que permite recorrerlas en el mismo orden en que aparecen las letras en el texto, es decir, de izquierda a derecha. Este ordenamiento es fundamental para poder analizar correctamente las distancias entre caracteres y que las mismas no resulten negativas.

El cálculo de las distancias se realiza tomando el límite derecho de un carácter y el límite izquierdo del siguiente, y registrando la diferencia entre ambas posiciones. De esta manera se obtiene una lista de distancias horizontales entre letras. A partir de dicha lista, se calcula la **mediana de las distancias** y se la multiplica por un factor de **2.5**, valor que actúa como umbral para la detección de espacios.

Si alguna de las distancias supera este umbral, se considera que en el texto existe al menos un espacio. Este método ofrece buenos resultados en campos donde se detectan múltiples caracteres, ya que la mediana proporciona una estimación robusta frente a posibles variaciones entre letras.

En los casos en los que el campo contiene únicamente dos caracteres, el cálculo de la mediana pierde efectividad debido al reducido número de muestras. Para estos casos particulares, se define un umbral fijo de **10 píxeles**, determinado de forma empírica en función del ancho promedio de los caracteres observados (aproximadamente entre 6 y 8 píxeles).

4. Verificar las respuestas:

En esta etapa se lleva a cabo la verificación de las respuestas seleccionadas en el formulario. El propósito de este procedimiento es comprobar que cada pregunta tenga marcada una única opción y que dicha marca corresponda a un único carácter.

Cada recorte utilizado en esta parte del análisis corresponde a una región de la imagen que contiene las dos posibles respuestas para una pregunta, separadas por un divisor central. Para realizar la verificación, se aplica el método de conteo de caracteres sobre la imagen correspondiente a cada pregunta, y al resultado obtenido se le resta una unidad para descontar el componente asociado al separador entre las opciones.

Si la cantidad de caracteres detectados es exactamente **uno**, se considera que la pregunta fue respondida correctamente, es decir, que el usuario marcó una única opción con un único carácter. En cambio, si se detecta más de un carácter, se interpreta que la pregunta fue marcada de forma incorrecta, ya sea porque se seleccionaron ambas opciones o porque se utilizó más de un único carácter en la marca.

5. Generar una imagen de salida:

Para mostrar de manera visual si los campos del formulario fueron ingresados correctamente, se implementa un procedimiento de validación que genera una imagen de salida. Esta función recibe como entrada la imagen del formulario correspondiente al campo “Nombre y Apellido” y evalúa cada sección utilizando los criterios y funciones previamente descritos, determinando si los valores ingresados cumplen con las condiciones establecidas.

Una vez evaluados todos los campos, se genera un mensaje visual que refleja el resultado general. Si todos los campos se encuentran correctamente ingresados, se muestra un texto de confirmación en color verde; en caso contrario, cuando algún campo no cumple los criterios, se muestra un texto de error en color rojo.

Aplicación de las funciones implementadas:

En conclusión, mediante la implementación de las funciones y procedimientos previamente explicados, es posible resolver el problema planteado en el trabajo práctico aplicando las reglas específicas definidas para cada campo. De este modo, por ejemplo, en el caso del nombre y apellido se comprueba que el texto contenga al menos dos palabras y no exceda los 25 caracteres; en el caso de la edad, se verifica que el valor ingresado contenga entre dos y tres caracteres numéricos y que no existan espacios; y

así sucesivamente con los demás apartados. En síntesis, a partir del conteo de caracteres y la detección de espacios, junto con las condiciones particulares de cada campo, se logra validar la información ingresada de manera automatizada.

resultados_formularios.csv

	ID	Nombre y Apellido	Edad	Mail	Legajo	Pregunta 1	Pregunta 2	Pregunta 3	Comentarios
1	01	OK	OK	OK	OK	OK	OK	OK	OK
2	02	MAL	MAL	MAL	MAL	MAL	MAL	MAL	MAL
3	03	OK	OK	OK	OK	OK	OK	OK	OK
4	04	MAL	MAL	MAL	MAL	OK	MAL	MAL	MAL
5	05	MAL	MAL	OK	OK	MAL	MAL	MAL	OK

Finalmente, los resultados obtenidos se procesan en el script principal, donde se genera una imagen de salida y se almacenan todos los datos verificados en un archivo CSV, completando así la resolución del problema.