Facultad de Ciencias Exactas, Ingeniería y Agrimensura

Tecnicatura Universitaria en Inteligencia Artificial

Procesamiento de Imágenes I

Trabajo Práctico 1

Docentes: Gonzalo Sad, Juan Manuel Calle y Joaquín Allione

Integrantes: Sofía Grando, Jeremías Olivieri y Miranda Pilar Onega

23 de octubre de 2025

Índice

Problema 1 - Ecualización local del histograma

1. Descripción del problema

El primer problema presenta la siguiente imagen:

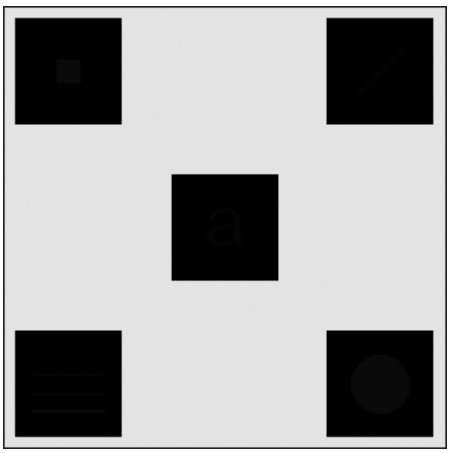


Figura 1

El objetivo principal es reconocer los detalles escondidos en la Figura 1 mediante la técnica de ecualización de histograma. Para ello, se deberá desarrollar una función que implemente la ecualización local del histograma, tomando como únicos parámetros de entrada la imagen del archivo Imagen_con_objetos_ocultos.tiff, y el tamaño de la ventana de procesamiento (MxN). Además, se deberá realizar un análisis de la imagen mostrada e informar cuáles son los detalles escondidos en las diferentes zonas de la misma. Por último, se requiere analizar la influencia del tamaño de la ventana en los resultados obtenidos.

2. Análisis de desafíos enfrentados

Algunos de los desafíos enfrentados al resolver este problema fueron:

- Aplicar y comprender la ecualización local a una imagen, haciendo un procesamiento previo y posterior para obtener los resultados deseados.
- Ajustar adecuadamente el tamaño de la ventana de procesamiento (MxN) para equilibrar la mejora del contraste local sin introducir ruido o artefactos visuales.
- Gestionar los bordes de la imagen al aplicar la técnica, seleccionando un método de extensión (reflect, replicate, constant, etc.) que preserve la coherencia visual y evite distorsiones.

3. Técnicas utilizadas

Para abordar este problema, se comenzó tomando la imagen original y a cada píxel de esta se le aplicó una máscara mxn. Para realizar esto, se tuvo que agregar un borde a la imagen,

a partir de las distintas funciones disponibles en cv2. Esto debido a que para píxeles cercanos a los bordes originales de la misma, esta máscara iba a sobresalir (y el código iba a fallar).

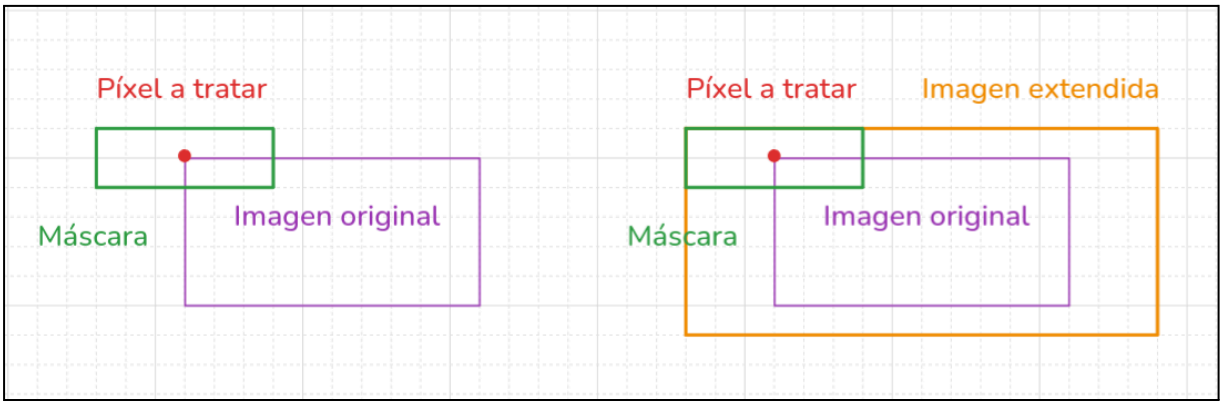


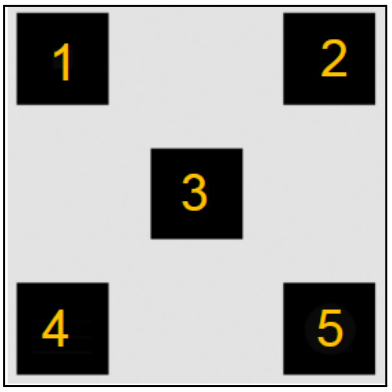
Figura 2

Una vez aplicada, se recortó la imagen extendida con un tamaño igual al de la máscara. A este recorte, se le aplicó la función *equalizeHist* de cv2, con el fin de poder filtrar los píxeles más oscuros y así obtener las figuras ocultas.

Para obtener la nueva imagen filtrada, se creó una imagen nueva (con el mismo tamaño de la original) y se fueron imprimiendo en ellas los píxeles centrales del recorte una vez aplicado el ecualizador. Cabe mencionar que para todo este proceso, se tuvo que tener cuidado con el tamaño de la máscara. Como se tenía que obtener “el píxel del centro”, que corresponde al píxel original de nuestra imagen. Al aplicar una máscara con alguna de sus dimensiones par, esto no iba a ser posible.

El proceso se realizó con varios tamaños de máscaras y varios bordes.

4. Conclusiones



A partir de la ecualización local de histograma se pudieron identificar los siguientes elementos que estaban escondidos:

1. Un cuadrado.
2. Una línea diagonal.
3. La letra ‘a’.
4. Cuatro líneas paralelas.
5. Un círculo.

Figura 3

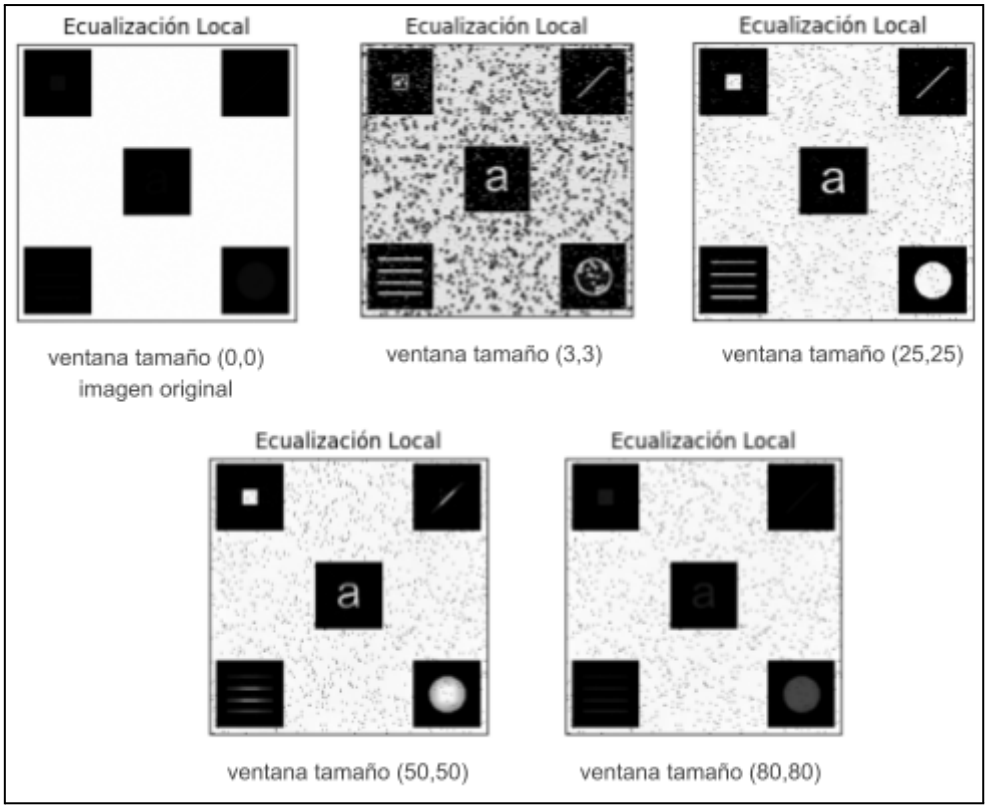


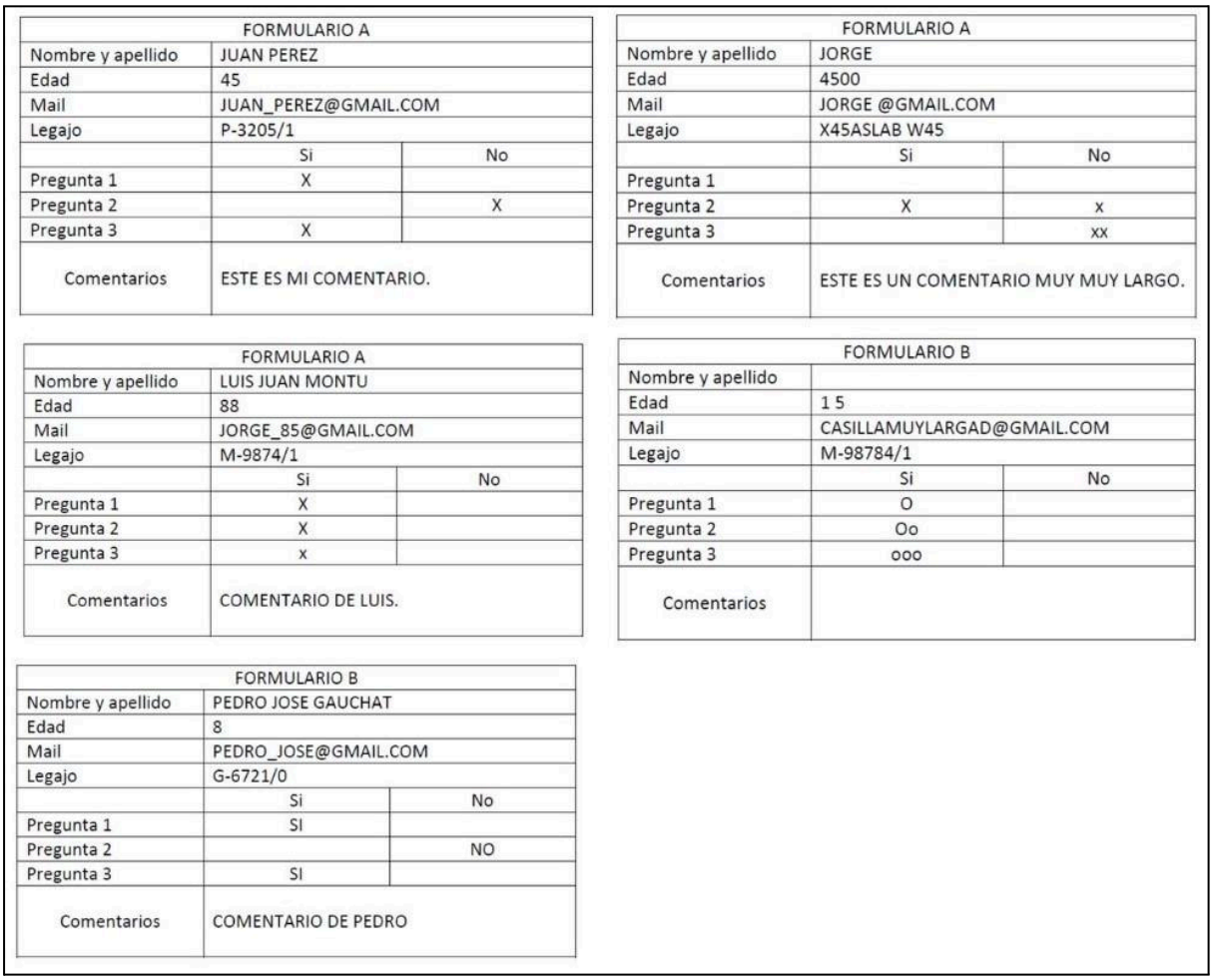
Figura 4

Se puede analizar en esta secuencia que: si la ventana es pequeña, se observa mucho ruido y los objetos ocultos no se ven claramente. A medida que se la va aumentando (alrededor de 25) se observan más nítidas, mejorando el contraste. Sin embargo, al aumentar demasiado el tamaño de la ventana, comienzan a desaparecer, ya que se va a estar tomando una mayor cantidad de píxeles negros alrededor del píxel al cual se quiere aplicar el ecualizador.

Problema 2 - Validación de formulario

1. Descripción del problema

El segundo problema, presenta las siguientes imágenes correspondientes a distintas versiones de un mismo formulario:



FORMULARIO A		
Nombre y apellido	JUAN PEREZ	
Edad	45	
Mail	JUAN_PEREZ@GMAIL.COM	
Legajo	P-3205/1	
	Si	No
Pregunta 1	X	
Pregunta 2		X
Pregunta 3	X	
Comentarios	ESTE ES MI COMENTARIO.	

FORMULARIO A		
Nombre y apellido	JORGE	
Edad	4500	
Mail	JORGE @GMAIL.COM	
Legajo	X45ASLAB W45	
	Si	No
Pregunta 1		
Pregunta 2	X	x
Pregunta 3		xx
Comentarios	ESTE ES UN COMENTARIO MUY MUY LARGO.	

FORMULARIO A		
Nombre y apellido	LUIS JUAN MONTU	
Edad	88	
Mail	JORGE_85@GMAIL.COM	
Legajo	M-9874/1	
	Si	No
Pregunta 1	X	
Pregunta 2	X	
Pregunta 3	x	
Comentarios	COMENTARIO DE LUIS.	

FORMULARIO B		
Nombre y apellido		
Edad	1 5	
Mail	CASILLAMUYLARGAD@GMAIL.COM	
Legajo	M-98784/1	
	Si	No
Pregunta 1	O	
Pregunta 2	Oo	
Pregunta 3	ooo	
Comentarios		

FORMULARIO B		
Nombre y apellido	PEDRO JOSE GAUCHAT	
Edad	8	
Mail	PEDRO_JOSE@GMAIL.COM	
Legajo	G-6721/0	
	Si	No
Pregunta 1	SI	
Pregunta 2		NO
Pregunta 3	SI	
Comentarios	COMENTARIO DE PEDRO	

Figura 5

A partir de dichos formularios mostrados en la Figura 3, se pide validar de forma automática los distintos campos (*Nombre y apellido, Edad, Mail, Legajo, Pregunta 1, Pregunta 2, Pregunta 3 y Comentarios*) con un script en Python, teniendo en cuenta las siguientes restricciones:

- ***Nombre y apellido***: Debe contener un mínimo de dos palabras y no más de 25 caracteres en total.
- ***Edad***: Debe contener 2 o 3 caracteres consecutivos (no deben existir espacios entre ellos).
- ***Mail***: Debe contener una palabra y no más de 25 caracteres.
- ***Legajo***: Debe contener sólo 8 caracteres en total, formando una única palabra.

- ***Preguntas 1, 2 y 3***: En cada pregunta debe haber una única celda marcada (referenciadas a las columnas de *Si* y *No*), con un único caracter. Es decir, en cada pregunta, ambas celdas no pueden estar marcadas ni ambas celdas pueden quedar vacías.
- ***Comentarios***: Debe contener al menos una palabra y no más de 25 caracteres.

Primero, el algoritmo debe tomar como parámetro de entrada la imagen de un formulario y mostrar por pantalla si cada campo es correcto (*OK*) o incorrecto (*MAL*). Dicho algoritmo debe ser aplicado de forma cíclica sobre el conjunto de las cinco imágenes e imprimir los resultados en función del tipo de formulario (*A*, *B* o *C*).

Luego, se debe generar una única imagen de salida que muestre cuáles personas completaron de manera correcta el formulario y cuáles no. Esta nueva imagen debe contener el contenido recortado del campo *Nombre y apellido* de cada formulario y diferenciar con algún indicador los que correspondan a formularios correctos de los incorrectos.

Por último, se debe generar un archivo CSV donde se almacene la validación de cada formulario. Cada fila debe corresponder al conjunto de resultados de un formulario procesado. A su vez, debe tener un identificador único en la primera columna que corresponda al ID del nombre del archivo. Debe contar también con una serie de columnas que correspondan a los campos nombrados anteriormente, respetando el mismo orden. Y cada celda debe registrar el resultado de la validación.

2. Análisis de desafíos enfrentados

Los desafíos enfrentados en este problema tuvieron que ver principalmente con la detección de los renglones, de las palabras y de los caracteres. Por ejemplo, al querer detectar palabras y hacerles un bounding box, al principio tomaba más área de la que se debía como se muestra en la Figura 6, o no consideraba los espacios entre medio de dos palabras, como se muestra en la Figura 7. En el caso de los caracteres, sucedió que en algunos casos, a dos caracteres contiguos, en lugar de detectarlos de esa manera, se los detectaba como un solo caracter como en la Figura 8. A cada uno de estos desafíos se los fue resolviendo probando y cambiando los umbrales hasta conseguir un buen resultado como el de la Figura 9.

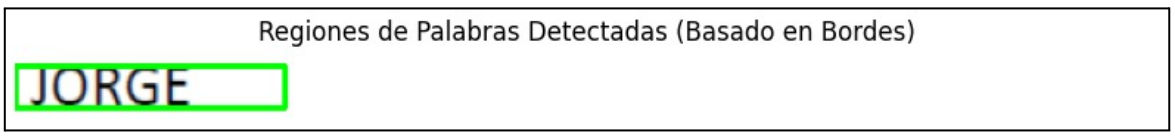


Figura 6

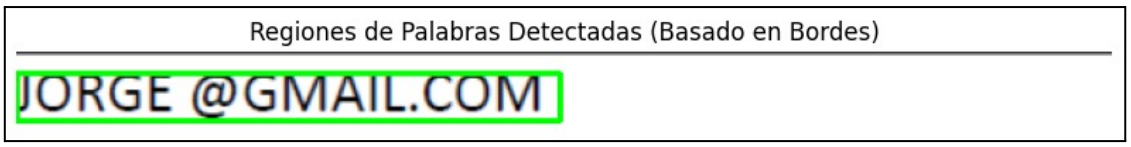


Figura 7

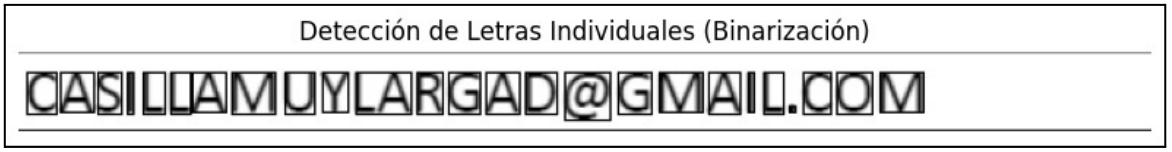


Figura 8

En el proceso de encontrar cómo contar la cantidad de palabras y caracteres que contenía cada campo, se utilizaron visualizaciones que fueron dejadas como comentarios dentro del código. Estas ayudaron a visualizar y ajustar los parámetros para que funcionen de manera correcta.

3. Técnicas utilizadas

Con respecto a las técnicas utilizadas, en principio se inicia con una detección de estructura del formulario. Esto se logra mediante la aplicación de filtros espaciales para resaltar los bordes y líneas en cada formulario. Después, la función *extraer_campos* segmenta la imagen de entrada sumando los valores de píxeles a lo largo de las filas y columnas de la imagen binarizada para identificar las coordenadas exactas de las líneas divisorias, definiendo así las Regiones de Interés (ROI) para cada campo (Figura 9).

Figura 9

Una vez que se tienen las regiones de interés, se pasó a la validación del contenido de los campos. Esto se divide en dos conteos:

1. **Conteo de palabras:** La función *detectar_palabras* usa el detector de bordes Canny seguido de una dilatación morfológica (con un kernel ancho). Esta dilatación conecta los contornos de los caracteres contiguos, permitiendo que *cv2.findContours* identifique los contornos como bloques de palabras.
2. **Conteo de caracteres:** La función *detectar_caracteres* utiliza la binarización y componentes conectadas *cv2.connectedComponentsWithStats* para rodear y contar cada carácter individualmente. Para separar las palabras, se analiza la distancia horizontal entre los rectángulos delimitadores de los caracteres contiguos.

Estos luego conteos se utilizan en las funciones de validación de campos, que aplican las reglas específicas del formulario para determinar si cada campo es *OK* o *MAL*. Además, se identifica el tipo de formulario midiendo el área del carácter detectado en el encabezado.

4. Conclusiones

Figura 10

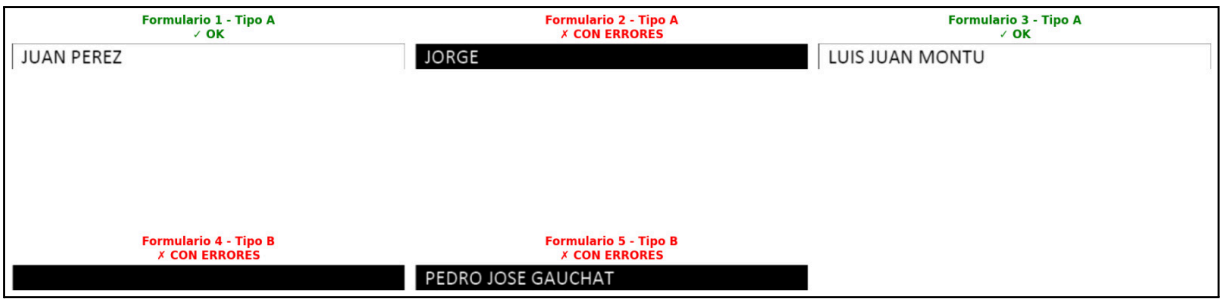


Figura 11

```
1  ID,Nombre y Apellido,Edad,Mail,Legajo,Pregunta 1,Pregunta 2,Pregunta 3,Comentarios
2  1,OK,OK,OK,OK,OK,OK,OK,OK
3  2,MAL,MAL,MAL,MAL,MAL,MAL,MAL,MAL
4  3,OK,OK,OK,OK,OK,OK,OK,OK
5  4,MAL,MAL,MAL,MAL,OK,MAL,MAL,MAL
6  5,OK,MAL,MAL,OK,MAL,MAL,MAL,OK
```

Figura 12

En resumen, en la Figura 10 se muestran cómo quedan las validaciones de los campos para cada formulario.

Luego en la Figura 11, que es la imagen de salida, se muestran cuáles formularios eran correctos o incorrectos. El primero, identificado con un recorte de la imagen original donde el fondo es blanco y las letras negras, un texto de color verde que indica el nombre del formulario, el tipo y que esta ‘OK’. Y el segundo, con fondo negro y letras en blanco (negativo) y un texto de color rojo que indica el nombre del formulario, el tipo y que tiene errores.

Por último, en la Figura 12 se muestra cómo queda el archivo csv con la primera línea siendo los nombres correspondientes a los campos de los formularios y el resto, sus respectivas validaciones.