Facultad de
Ciencias Exactas,
Ingeniería y Agrimensura

Trabajo práctico PDI

Ecualización local de histograma

Validación de formulario

Año 2025 - 2° Semestre

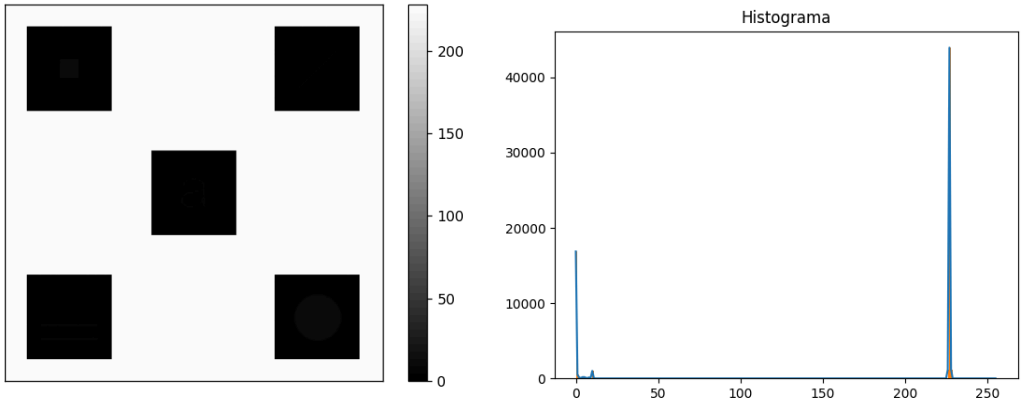
Integrantes: Juana Chies(C-7554/1), Mancini Nicolas (M-7429/2).

Profesores: Gonzalo Sad, Juan Manuel Calle, Joaquín Allione.

Problema 1

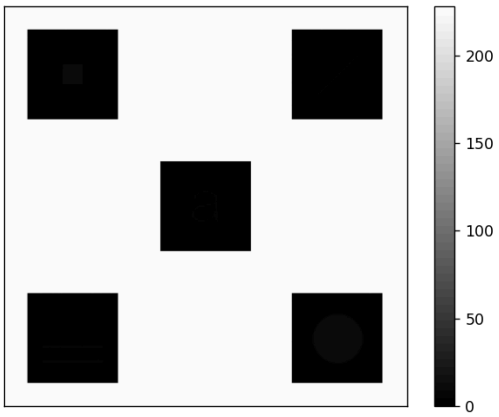
El primer problema consiste en definir una función que implemente la ecualización local del histograma, la cual reciba como parámetros una imagen y el tamaño de la ventana determinada por M y N. La función recorre la imagen píxel a píxel, para calcular el histograma de la ventana(MxN) en cada posición y ajustar la intensidad del píxel central para resaltar detalles.

Para empezar, observamos la imagen original, además de realizar un histograma de la imagen para ver la distribución de intensidad del color de los píxeles.

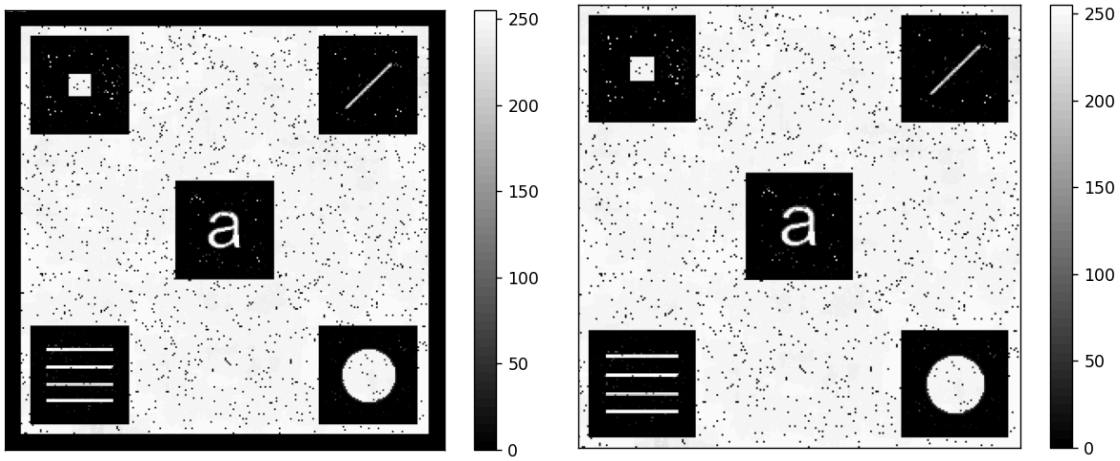


A partir de este gráfico, pudimos observar que no se podía realizar una ecualización global ya que había dos máximos locales, uno cerca de 0 y otro cerca de 227. Por esta razón, optamos por usar una ecualización local, la cual iba a almacenarse en una función.

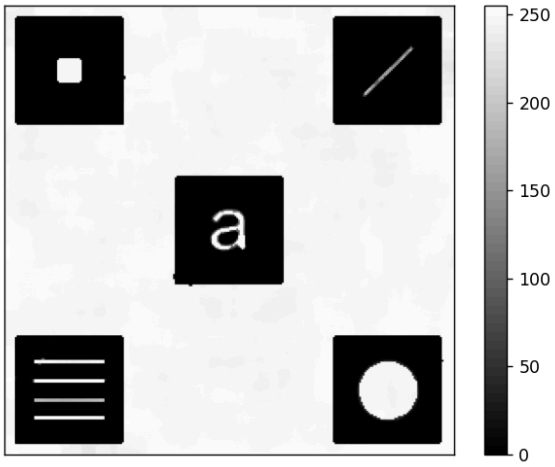
Para ello, lo primero que tuvimos que hacer fue extender la imagen basándonos en el tamaño de la ventana o kernel (lo cual iba a ser ingresado en nuestra función como parámetro). Ejemplo con kernel (20x20)



Teniendo esto, recorrimos la imagen completa, píxel por píxel, utilizando nuestro kernel para aplicar una ecualización de histograma local en cada vecindario, mejorando el contraste en zonas específicas. Luego, recortamos los bordes agregados durante el proceso para recuperar el tamaño original de la imagen.



Como en la imagen obtenida detectamos ruido del tipo pepper and salt, aplicamos un filtrado para eliminar los puntos extremos de ruido y suavizar la imagen sin perder detalles importantes, obteniendo así el resultado final.



Para concluir, la ecualización local del histograma permitió mejorar el contraste y resaltar los detalles que estaban ocultos en distintas zonas de la imagen. Además, el filtrado aplicado posteriormente eliminó el ruido, una imagen final más clara, homogénea y visualmente equilibrada sin perder información relevante.

Problema 2

El segundo problema consiste en definir una función que permita la validación automática de formularios a partir de imágenes, verificando que cada campo cumpla con restricciones específicas:

- Nombre y Apellido: entre dos palabras y máximo 25 caracteres;
- Edad: 2 o 3 dígitos consecutivos;
- Mail: una palabra de hasta 25 caracteres;
- Legajo: 8 caracteres formando una única palabra;
- Preguntas 1-3: una única celda marcada por pregunta, sin duplicados ni vacíos;
- Comentarios: al menos una palabra y máximo 25 caracteres.

Para realizar estas verificaciones necesitamos obtener las respuestas de cada campo, para ello, comenzamos cargando la imagen y aplicando un umbral manual para separar fondo de contenido. A partir de esta imagen binaria, detectamos las líneas horizontales que delimitan los renglones y, en base a ellas, recortamos cada campo del formulario.

Imagen original

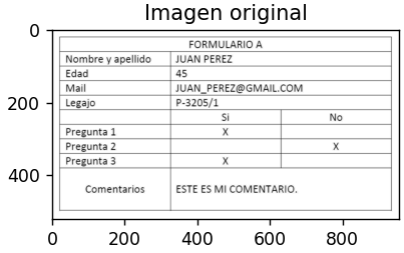


Imagen con umbralado manual

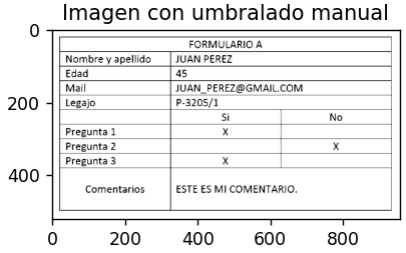
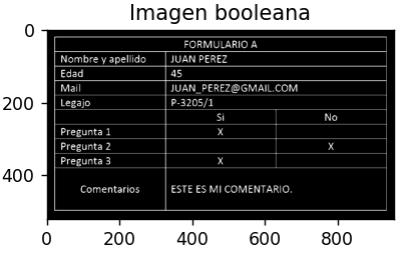


Imagen booleana



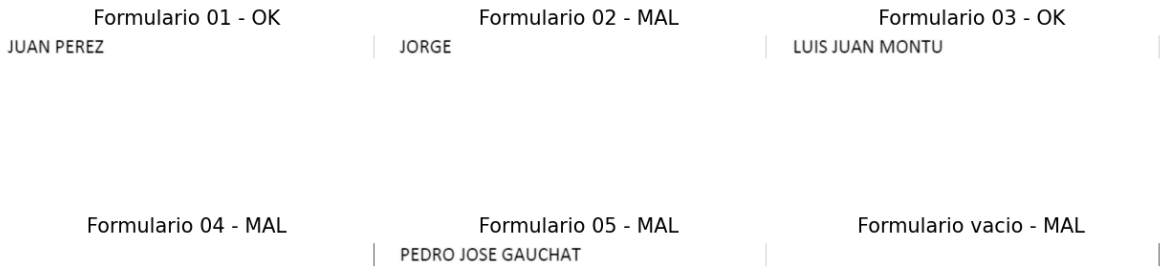
Dentro de cada renglón, analizamos las líneas verticales para segmentar las celdas correspondientes a cada campo. Sobre cada celda, aplicamos un umbral adaptativo con un kernel, con el fin de mejorar la detección de letras mediante componentes conectados. Posteriormente, filtramos el ruido descartando los elementos con áreas o proporciones no válidas.

Una vez que determinamos los componentes válidos de cada celda, contamos las letras, que son ordenadas según su posición horizontal, y las palabras, que son diferenciadas cuando la distancia entre dos letras consecutivas supera 8 píxeles.

Ahora, con los conteos obtenidos, realizamos los chequeos comparándolos con las restricciones de cada campo, por ejemplo, en las celdas de Nombre y apellido, Edad, Mail, Legajo o Comentarios verificamos la cantidad máxima/mínima de letras y palabras. Para las preguntas: se comprueba que sólo haya respuesta en una de las opciones y que las letras sean como máximo 1. Además, identificamos el tipo de formulario, basándonos en los contornos y cantidad de huecos, al conocer que las únicas letras posibles son A, B o C y que se diferencian entre ellas.

Así, la función devuelve:

- Un crop del nombre y apellido presente en el formulario y el estado del mismo, que puede ser correcto e incorrecto.



- Un diccionario, con los campos como clave y sus estados (OK o MAL) como valores.

Finalmente, realizamos la verificación sobre todos los formularios, obteniendo un gráfico que funciona como resumen de su estado, indicando la persona que lo realizó, junto con un archivo que contiene toda la información y validaciones de cada campo.

En conclusión, la función de detección y validación funcionó bien para reconocer las letras y palabras de los formularios, y comprobar si estaban correctas. Así, esta técnica puede servir como una forma práctica de automatizar la verificación de formularios.

Archivo csv obtenido:

indice	Tipo	Nombre y apellido	Edad	Mail	Legajo	Pregunta 1	Pregunta 2	Pregunta 3	Comentarios
01	A	OK	OK	OK	OK	OK	OK	OK	OK
02	A	MAL	MAL	MAL	MAL	MAL	MAL	MAL	MAL
03	A	OK	OK	OK	OK	OK	OK	OK	OK
04	B	MAL	MAL	MAL	MAL	OK	MAL	MAL	MAL
05	B	OK	MAL	OK	OK	MAL	MAL	MAL	OK
vacio	A	MAL	MAL	MAL	MAL	MAL	MAL	MAL	MAL