

**Facultad de  
Ciencias Exactas,  
Ingeniería y Agrimensura**



**FCEIA**

Procesamiento de imágenes

---

## **Trabajo práctico 1 – 2025**

### **Integrantes**

|                            |                 |
|----------------------------|-----------------|
| Armas Alejandro            | <i>A-4685/1</i> |
| Ferreira da Camara Facundo | <i>F-3766/4</i> |
| Soda Gabriel               | <i>S-5679/1</i> |

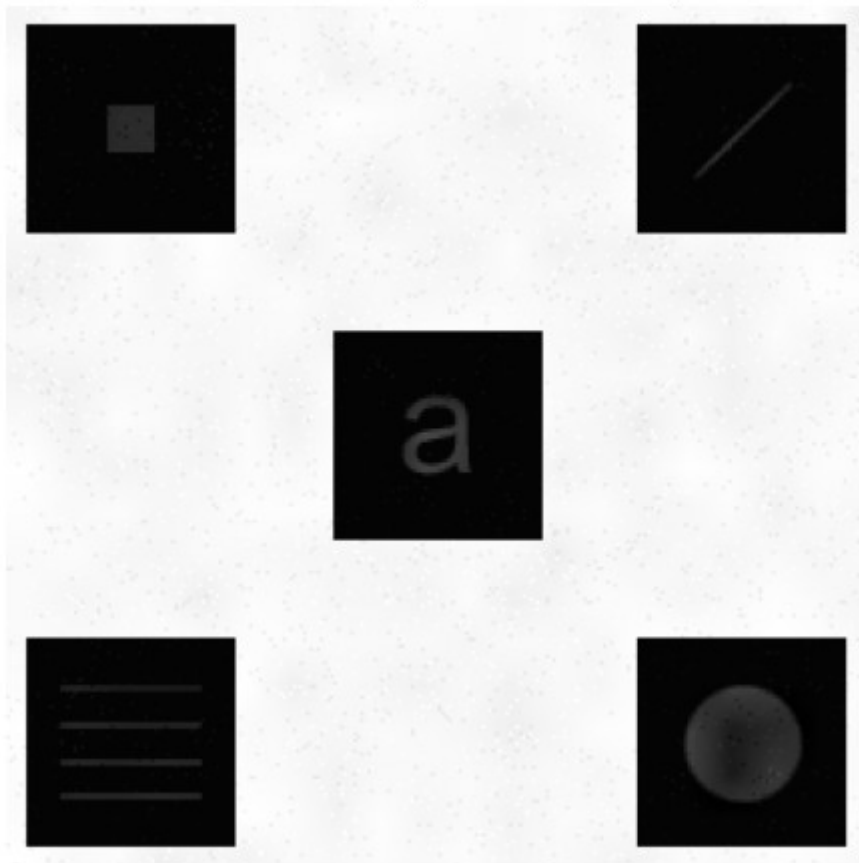
# Problema 1

## Ecualización local de histograma

Las imagenes que podemos detectar en la imagen .tiff son:

- un cuadrado en la esquina superior izq
- una linea diagonal ( / ) en la esquina superior der
- en el centro una a minuscula
- en la esquina inferior izq 4 lineas horizontales apiladas similares a un menu de celular
- por ultimo en la esquina inferior der, un circulo relleno

IMAGEN con (ventana 21x21)



originalmente probamos con diferentes dimesiones de ventanas para la ecualizacion local, pero terminamos teniendo mas exito utilizando valores entre 10 y 21. Apartir de 10 \* 10 vemos que se pueden distinguir las formas ocultas, al ir aumentando la ventana va aumentando los efectos del contraste, ya que los marcos de la imagen son de un tamaño grande y la imagen contiene pixeles ruidosos que hacian menos eficiente utilizar valores menores a 10.

luego de realizar la ecualizacion local procedemos a normalizar utilizando cv2.NORM\_MINMAX.

# Problema 2

## Validación de formulario

El problema consiste en analizar imágenes de formularios y validarlos en función de una serie de reglas para cada campo.

En primer lugar, diseñamos la función *procesar\_imagen\_formulario()*, que toma la imagen de un formulario, la binariza (umbral automático OTSU), la invierte para que los píxeles del contenido tengan valor 255 (blanco sobre negro) y la recorta ajustandose a la tabla del formulario. Con esto podemos procesar la imagen de cualquier formulario y obtenerla 'normalizada' y lista para trabajar sobre ella.

| 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. |    |

Figure 1: Ejemplo del formulario\_01.png luego de aplicarle la función

Programamos la función *obtener\_celdas()* que toma la imagen procesada de un formulario y devuelve un array de celdas (subimagenes del formulario) en el formato `celda=[filas[columnas]]`, es decir, para acceder a la celda con el número que indica la edad debemos escribir `celdas[2][1]` (Tercer fila, segunda columna)

El algoritmo para identificar las celdas consiste en iterar sobre las sumas de todas las columnas a lo largo de las filas y quedarnos con las que sumen  $255 * A$ , siendo  $A$  el ancho de la imagen. De esta forma identificamos filas de píxeles completamente pintadas \*, es decir, filas de la tabla. Luego se aplica análogamente para encontrar las separaciones de columnas (columnas de píxeles que sumen  $255 * H$ , siendo  $H$  la altura de la celda dada por la diferencia de altura de 2 filas de píxeles completamente pintadas no consecutivas.

\*Surge el problema de que algunas columnas o filas de la tabla pueden tener mas de 1px de grosor. Para eso, al iterar sobre la suma del eje contrario, se guardan unicamente las filas o columnas no consecutivas

Para validar correctamente los campos necesitabamos poder contar la cantidad de letras y palabras de cada celda, para esto desarrollamos 2 funciones:

*contar\_letras()* : Recibe la subimagen de una celda y devuelve la cantidad de contornos detectados. Se consideró y se probó aplicar apertura morfológica para mejorar la detección de las letras. Pero finalmente nos quedamos con la opción mas sencilla por ser la más efectiva.



Figure 2: Ejemplo de detección de letras aplicada a una celda

*contar\_palabras()* : Recibe la subimagen de una celda, le aplica dilatación morfológica y devuelve la cantidad de contornos detectados, que debido a la dilatación coincidirá con la cantidad de palabras.

La dilatación se aplica 2 veces con un kernel de 3x3, de esta forma obtuvimos los mejores resultados.



Figure 3: Ejemplo de detección de palabras aplicada a una celda. Originalmente dice "ESTE ES MI COMENTARIO."

Una de las consignas requería identificar el tipo de formulario ('A', 'B' o 'C'), para esto desarrollamos la función *identificar\_formulario()* que toma el crop de la celda inicial, identifica la componente conectada más a la derecha (La letra que identifica al formulario) y compara su valor de Área con los valores precalculados de Área de las letras. (Por ejemplo 'A' tiene área 134 y 'B' 152).

Para validar o no los formularios, utilizando las funciones previamente descritas, generamos la función *obtener\_resultados()*, la cual recibe la lista de celdas de un formulario y devuelve una tupla conformada por un diccionario -cuyas claves son cada campo y los valores si están correctos o no- y un booleano indicando si el formulario es válido.

Finalmente generamos un csv concatenando todos los diccionarios de resultados de cada formulario. También producimos una imagen concatenando las subimágenes de las celdas de 'Nombre y apellido' de cada formulario, separando los correctos de los incorrectos, mediante la función *generar\_imagen\_resultados()*