

Resumen del Código de Procesamiento de Formularios

1. ¿Qué busca hacer este código?

Este código lo que intenta hacer es **corregir automáticamente un montón de formularios** que fueron llenados por distintas personas

El programa agarra cada imagen de un formulario, la revisa campo por campo (nombre, edad, unas preguntas de "Sí" o "No", etc.) y **decide si cada campo está bien llenado** según unas reglas que definimos.

Por ejemplo, que el nombre tenga al menos dos palabras o que en las preguntas de sí/no solo se haya marcado una de las dos casillas.

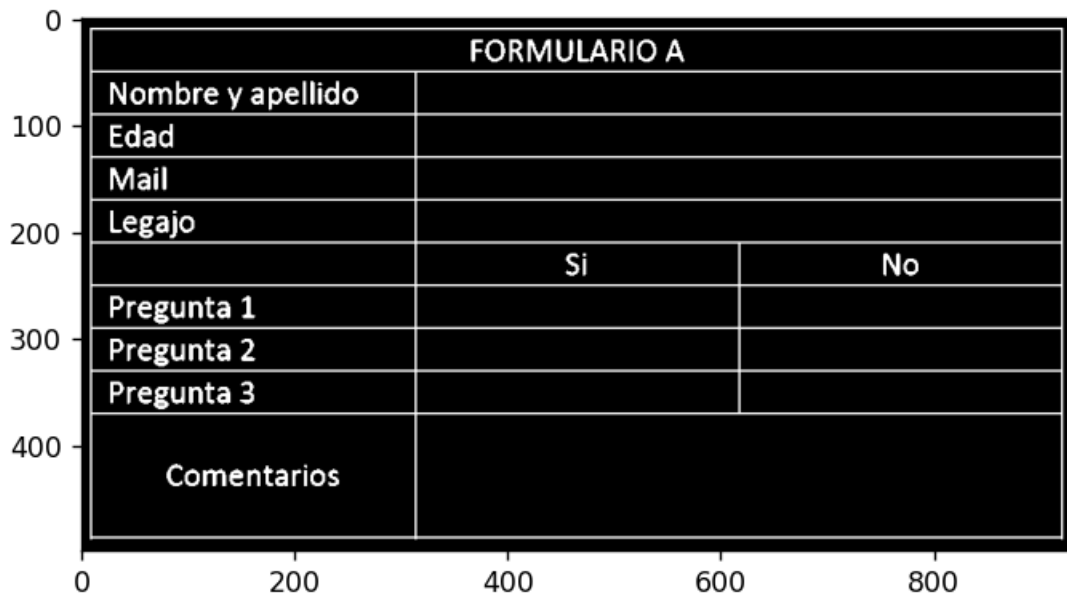
Al final, nos muestra una imagen resumen con los nombres de las personas y al lado un "OK" o "MAL" para saber si su formulario pasó la prueba, y también nos **genera un archivo CSV con todos los resultados** ordenados. De esta forma, evitamos realizar una primer validación de forma manual

¿Como empezamos?

Basandonos en el formulario base ("Formulario_vacio.png") buscamos las líneas divisorias de mayor dimensión, para realizar los primeros cortes por renglones.

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

Como primer medida, pasamos la imagen por un umbral de valor 200 (luego de varios ajustes). Esto nos dió una imagen binarizada



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

Luego, buscábamos realizar recortes de los renglones (Filas horizontales), pero también tener marcadas las filas verticales para delimitar estos renglones cuando sea necesario, pudiendo así separar entre Encabezado y Contenido de cada celda

Primer delimitación, intentamos con un Umbral estricto de 230 y un valor umbral de Columna del 95% del valor máximo.

Esto nos dio por resultado, un marcado claro de las filas/Columnas grandes, pero no de todas las que necesitábamos, ya que buscábamos obtener la división entre Encabezado y Contenido

```
th_col = img_cols.max()*0.95
```

```
th_row = 230
```



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

Luego de varios ajustes, los umbrales definidos fueron:

```
th_col = img_cols.max()*0.90
```

```
th_row = img_cols.max()*0.95
```

Dandonos un resultado mas cercano al esperado.

Celdas Delimitadas

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

2. ¿Con qué problemas nos enfrentamos?

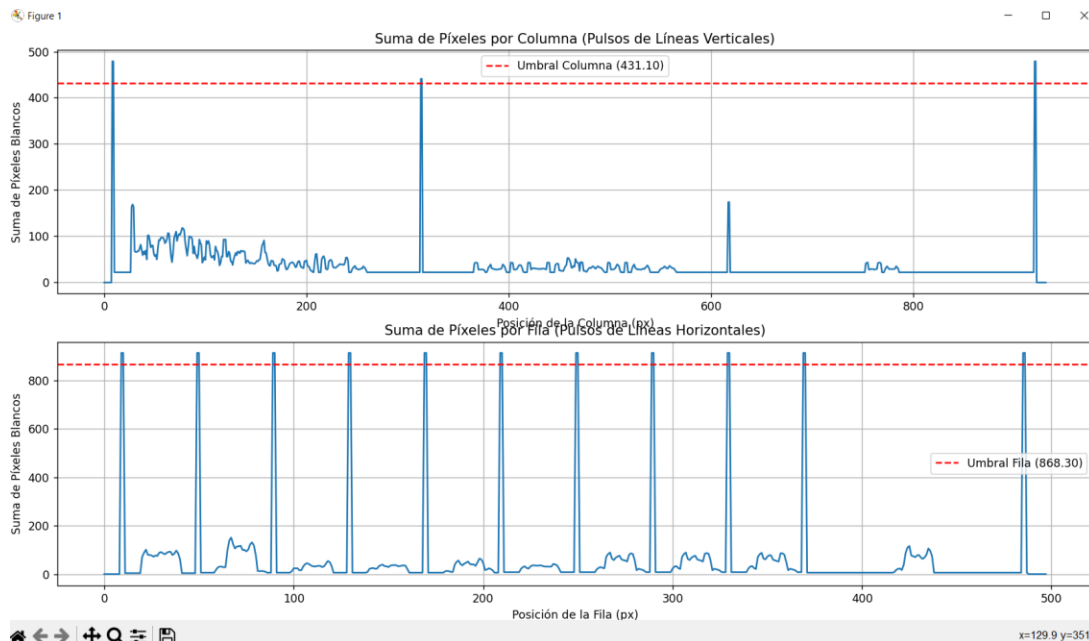
Cuando empezamos a hacerlo, nos topamos con varios problemas al procesar imágenes:

- **¿Dónde está cada cosa?:** El primer problema es que la debimos que idear una forma de **encontrar las líneas horizontales y verticales** de la tabla para poder "recortar" cada celda y analizarla por separado. Para esto, nos basamos en la ayuda descripta en el Trabajo Practica, de utilizar umbrales, y realizar sumatoria de valores de pixeles para encontrar las "líneas" o saltos.
- **"Leer"** Nuestro método es: contamos tamaños de pixeles. Esto nos trae problemas, como que una letra "i" puede contarla como dos caracteres (el punto y el palito) o que, si la letra está muy pegada, la cuenta como una sola.
- **Contar palabras:** Una vez que contamos los caracteres, ¿cómo sabemos cuántas palabras hay? Tuvimos que inventar una regla: si el espacio entre dos caracteres es más grande que un umbral, lo contamos como un espacio entre palabras.

3. ¿Qué técnicas de resolución usamos?

Para solucionar estos problemas, usamos algunas de las primeras técnicas que se ven en la materia:

- **Binarización (Umbralado):** Es lo primero que hacemos. Convertimos la imagen de escala de grises a una imagen en blanco y negro puro. Todo lo que es más oscuro que un cierto valor lo pasamos a negro (tinta) y el resto a blanco (papel). Usamos la función `cv2.threshold` para esto.}
- **Proyecciones de Píxeles:** Para encontrar las líneas de la tabla Sumamos todos los píxeles negros por cada fila y por cada columna. Donde hay una línea, hay un pico suma. Así detectamos las coordenadas exactas de todas las líneas del formulario para saber dónde cortar. En el código, esto se hace con `np.sum()`.



- **Análisis de Componentes Conectados:** Esta es la técnica clave. Una vez que tenemos una celda recortada y en blanco y negro, usamos `cv2.connectedComponentsWithStats`. Esta función nos encuentra todos los "grupos" de píxeles negros que están conectados entre sí. Nosotros asumimos que **cada grupo es un carácter**. Además, nos da el área de cada grupo, lo que nos sirve para filtrar el ruido

Utilizamos **8-conectados**, dado que es el método mas común, pero el resultado no variaba utilizando 4-conectados.

```
# 2. Obtener las componentes conectadas de esta única celda
num_labels, labels, stats, centroids = cv2.connectedComponentsWithStats(celda_binaria, 8, cv2.CV_32S)
```

- **Reglas Simples:** Para contar palabras o para validar las preguntas de "Sí/No", no usamos nada muy complejo. Aplicamos reglas basadas en la cantidad de caracteres. Por ejemplo: "Si en la celda 'Sí' hay un carácter (una 'X') y en la celda 'No' hay cero, entonces está OK".
- **Luego,** definimos las funciones para mostrar la imagen con los resultados, y guardar en un archivo CSV los mismo.