Universidad Nacional de Rosario

Facultad de Ciencias Exactas, Ingeniería y Agrimensura

Tecnicatura Universitaria en Inteligencia Artificial

PROCESAMIENTO DE IMÁGENES 1



Informe:

“Trabajo Práctico 1”

**Fecha:** 21/10/2025

**Autores:**

* CORRADINI, JULIO
* LEÓN, ESTEBAN
* SAAD, JUAN MANUEL
* ZIMMER, FEDERICO

**Docentes:**

* SAD, GONZALO
* CALLE, JUAN MANUEL
* ALLIONE, JOAQUIN

2025

**PROBLEMA 2 – VALIDACION DE FORMULARIO**

**Problemática a resolver**

Se tiene como objetivo desarrollar un script en Python para validar automáticamente una serie de formularios en formato de imagen, basándose en un conjunto de restricciones específicas para cada campo.

**Puntos a Desarrollar**

El algoritmo debe abordar los siguientes puntos principales:

* A. Tomar la imagen del formulario como entrada y mostrar por pantalla el resultado de la validación para cada campo: OK (correcto) o MAL (incorrecto).
* B. Aplicar la validación de forma cíclica sobre un conjunto de cinco imágenes de formularios e informar los resultados.
* C**.** Crear una única imagen que informe visualmente qué personas completaron el formulario correctamente y cuáles incorrectamente. Debe incluir un "crop" del contenido del campo Nombre y Apellido de cada formulario, con un indicador de validación.
* D**.** Crear un archivo CSV para almacenar los resultados de cada validación. Cada fila debe corresponder a un formulario procesado, con columnas para un ID único y los ocho campos validados, registrando el resultado como OK o MAL en cada celda.

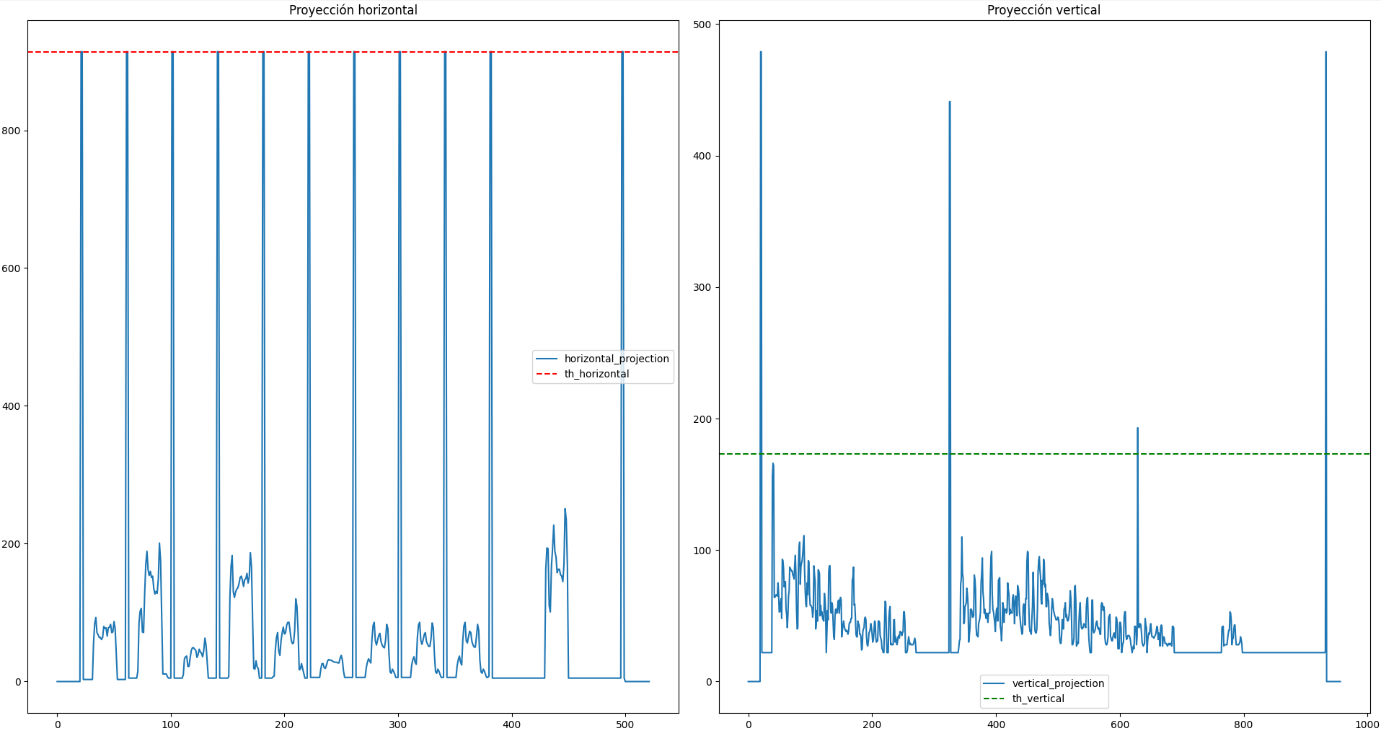
**Resolución**

Comenzamos analizando el formulario en blanco, cargando la imagen y aplicando un proceso de binarización. Luego, pasamos a la detección de líneas. El primer inconveniente que encontramos fue que todas las imágenes presentaban diferentes tonalidades de gris, lo que provocaba irregularidades en la binarización y generaba líneas que no eran completamente rectas.

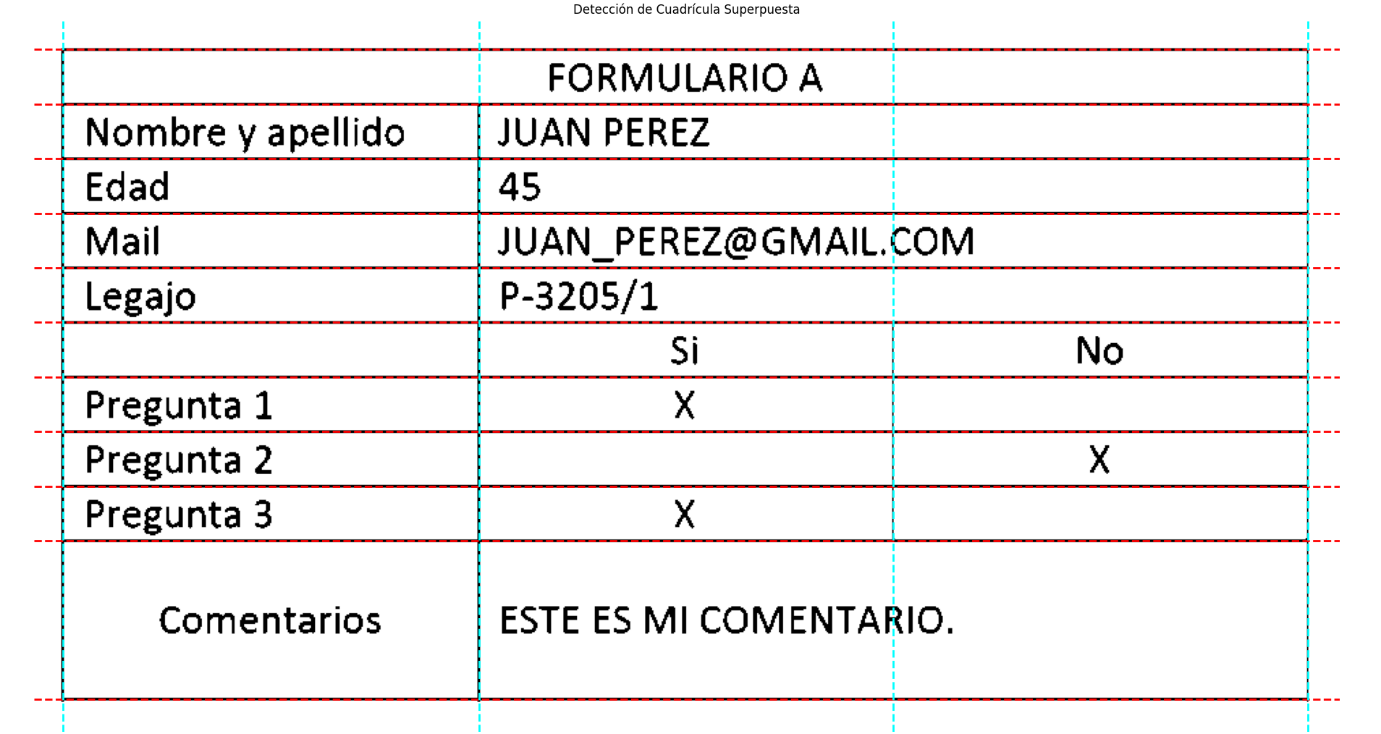
Para resolverlo, aplicamos el método de OTSU, que permitió establecer un umbral óptimo y uniforme, logrando una separación más precisa entre los píxeles blancos y negros. En el caso particular de las líneas verticales, algunas eran más largas que el recuadro del formulario y la última (de izquierda a derecha) resultaba más fina, esto no generó problemas significativos.

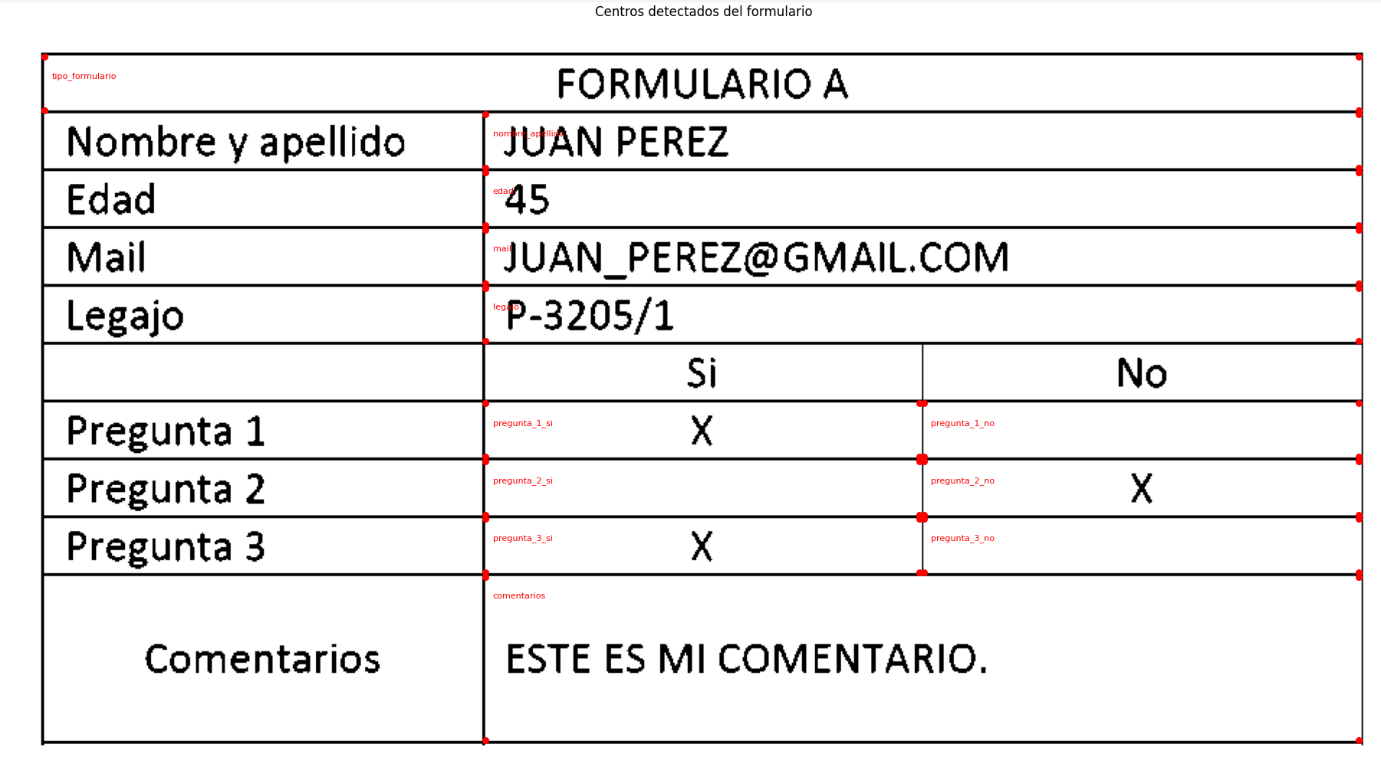
Para detectar las celdas realizamos una proyección horizontal y vertical de la imagen. Definimos un umbral en base a un percentil alto, ya que las líneas que queríamos identificar eran las que presentaban mayor cantidad de píxeles negros en la proyección.

El principal problema apareció en la proyección vertical, especialmente con la línea que separa las respuestas en la sección de preguntas. Dado que esta línea no era las más largas de la imagen, tuvimos que ajustar el percentil a un valor más bajo y visualizar los resultados mediante sliders, hasta encontrar un valor adecuado que evitara falsos positivos.

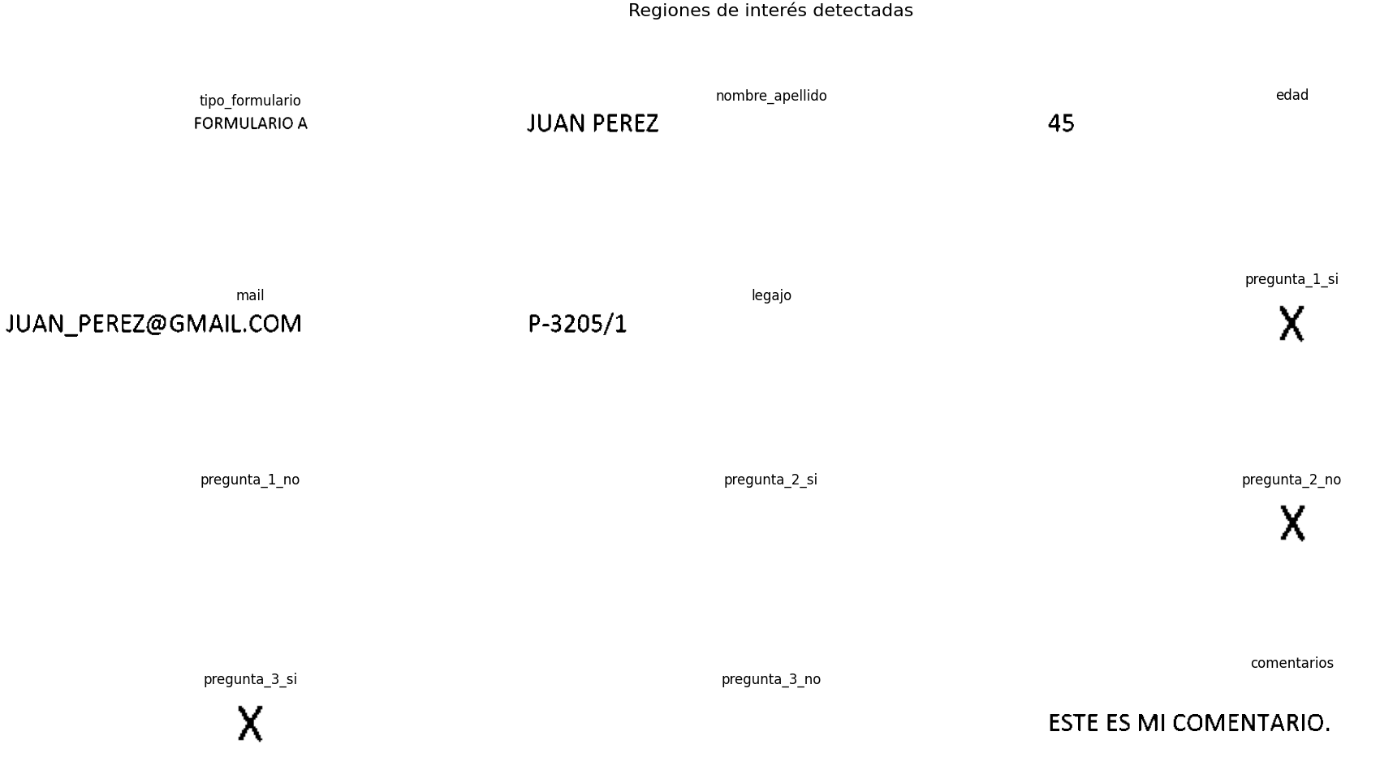


Para determinar las coordenadas de cada celda, analizamos los cambios entre píxeles negros y blancos utilizando las diferencias entre índices consecutivos con NumPy.  
Identificamos los “saltos” cuando la diferencia era mayor a uno, lo que correspondía a los espacios en blanco entre celdas.  
Debido a que el grosor de las líneas podía ser irregular, tomamos como referencia la media entre índices consecutivos (para líneas horizontales o verticales).



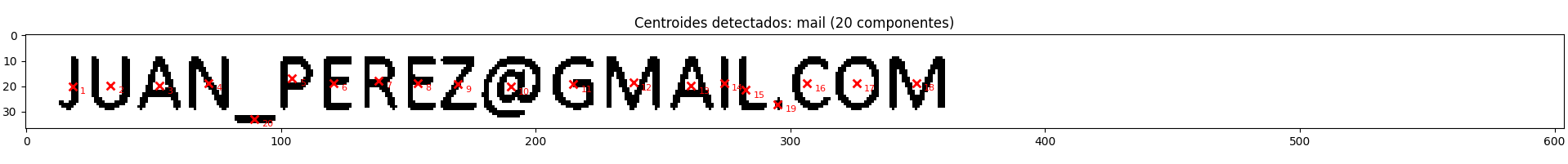
Posteriormente, calculamos las coordenadas exactas de cada celda y aplicamos un pequeño desplazamiento (1–2 píxeles) para evitar incluir las líneas negras dentro de la región de interés (ROI). Esto permitió obtener únicamente el contenido interior de cada celda. 

Con las coordenadas ya definidas, construimos un diccionario que almacena el array de la ROI correspondiente a cada celda del formulario.



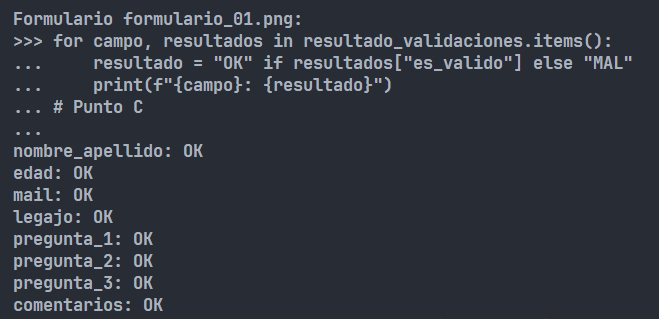
A continuación, utilizamos la función connectedComponentsWithStats de OpenCV para identificar los componentes conectados dentro de cada ROI.  
La detección de letras individuales funcionó correctamente, pero al intentar detectar palabras completas surgió un problema: no existía una función directa para agrupar letras en palabras.

Para resolverlo, empleamos los centroides de los objetos detectados. Calculamos las distancias entre centroides consecutivos y usamos una mediana de referencia para identificar posibles espacios entre palabras.  
Sin embargo, asumimos erróneamente que las distancias entre letras con espacios siempre serían mayores que entre letras consecutivas, lo cual no se cumplía en todos los casos.



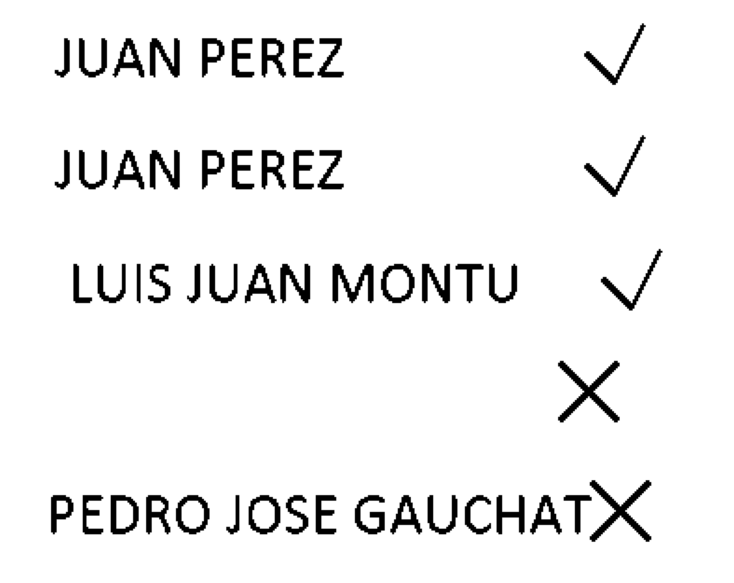
La solución fue definir un umbral constante por celda para distinguir palabras, determinado de forma empírica mediante pruebas manuales hasta encontrar valores adecuados.

Una vez que logramos contar letras y palabras correctamente, desarrollamos una función genérica de validación que aplicamos a cada formulario.  
Los resultados se almacenaron en un diccionario, que luego imprimimos por pantalla cumpliendo así con el punto A del enunciado.



Para el siguiente paso, realizamos los crops correspondientes al campo “Nombre y Apellido”.  
El primer inconveniente fue que las imágenes tenían dimensiones distintas, lo que impedía unirlas correctamente en un mismo canvas. Lo solucionamos tomando la imagen de mayor tamaño como base para generar un canvas en blanco, sobre el cual insertamos los recortes de cada formulario.

El segundo problema ocurrió en el formulario 4, donde el campo de nombre estaba vacío. En ese caso, la función connectedComponentsWithStats devolvía únicamente el centroide del fondo y otro con valor NaN, provocando un error al intentar ubicar la tilde o la cruz.  
Resolvimos este inconveniente utilizando el centroide del fondo como referencia para posicionar el indicador correspondiente.



Finalmente, con los diccionarios de resultados ya estructurados, creamos el archivo CSV solicitado. Generamos una lista con los encabezados y otra con las filas de resultados, y luego empleamos la función de escritura de CSV en Python para guardar los datos.  
Este paso se ejecutó correctamente, sin presentar inconvenientes.