Universidad Nacional de Rosario

Facultad de Ciencias Exactas, Ingeniería y Agrimensura

Trabajo práctico Nº 01

Tecnicatura Universitaria en Inteligencia Artificial

Procesamiento de imágenes (IA4.4)

Integrantes	Legajo	Mail
Martinez Dufour, Caterina	M-7169/2	caterinaxmd7@gmail.com
Porcelli, Fabricio	P-5340/6	fabricioporcelli@gmail.com
Lenarduzzi, Juan		

Fecha de entrega: 22/04/2025

Docentes: Gonzalo, Sad - Julián, Álvarez - Juan Manuel, Calle

ÍNDICE

1. Introducción

En este informe se explicará cómo se abordaron los problemas planteados utilizando diferentes estrategias vistas hasta el momento en la materia. Para cada solución, se comentarán los inconvenientes que surgieron durante el desarrollo, las modificaciones realizadas y la forma en que se resolvieron.

Además, se incluirán imágenes que faciliten la comprensión de los resultados obtenidos y en algunos casos, se presenta un ejemplo de su funcionamiento. El trabajo fue realizado en un repositorio de GitHub, que contenía un archivo README con instrucciones para poder ejecutar el programa paso a paso, dos archivos .py con las soluciones a los problemas dados y los archivos proporcionados por la cátedra.

2. Problema 1 - Ecualización local de histograma

Descripción del problema:

En este problema se trabaja con una imagen en formato .tif que presenta un fondo gris uniforme y cinco cuadrados negros, donde cada uno oculta un objeto. Para poder visualizar estos objetos, se necesita llevar a cabo una ecualización local del histograma usando una ventana (cuadrada o rectangular) que se desplaza pixel a pixel, centrada en cada punto de la imagen hasta recorrerla completamente.

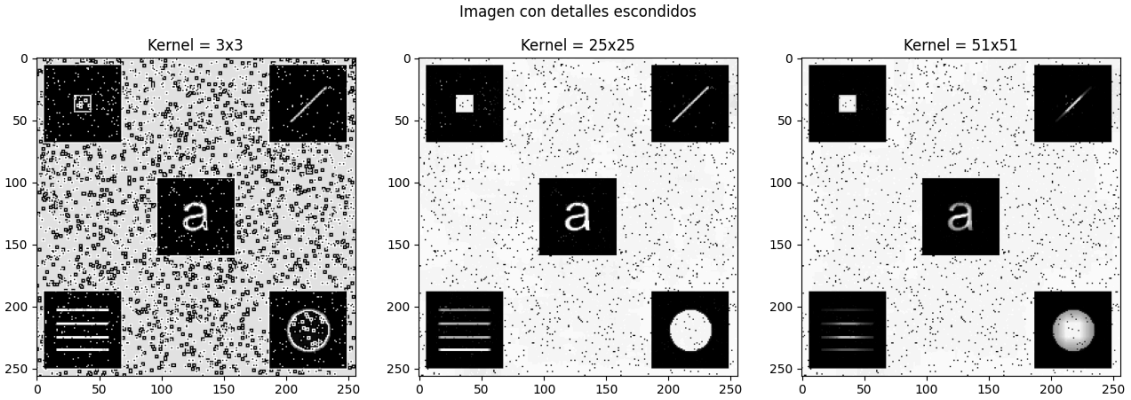
No se utiliza una ecualización global del histograma ya que este no es efectiva para revelar detalles localizados en áreas oscuras como las que contiene esta imagen.

Resolución del problema:

Para resolver este problema se utilizaron las librerías de OpenCV(cv2), Numpy(numpy) y Matplotlib (matplotlib.pyplot). Se trabajó con la imagen en formato .tif en escalas de grises para facilitar el análisis posteriormente y se utilizó una función que realiza la ecualización local del histograma de la siguiente manera:

- Generamos bordes replicados con “cv2.copyMakeBorder”.
- Creamos una imagen vacía del mismo tamaño y tipo de datos que la original pero con todos sus valores en negro (ceros), donde se guardara los resultados de la transformación.
- La imagen se recorrió píxel por píxel. En cada posición, se extrajo una ventana de tamaño M x N centrada en el píxel actual y se aplicó la ecualización del histograma sobre esa ventana, y el valor ecualizado del píxel central se guardó en la imagen de salida.
- El valor ecualizado del píxel central se guarda en la imagen transformada de salida.

Por último, hicimos la evaluación de la imágen con 3 tamaños diferentes:



Como podemos observar, si el tamaño de la ventana es muy pequeño (3x3) la imagen posee mucho ruido y no muestra de forma adecuada todas las figuras. Si la ventana es muy grande (51x51) tiene menor ruido pero los objetos ocultos pierden detalles. La mejor opción es usar una ventana intermedia (25x25), la cual tiene una imagen con menor ruido y se puede apreciar bien los objetos.

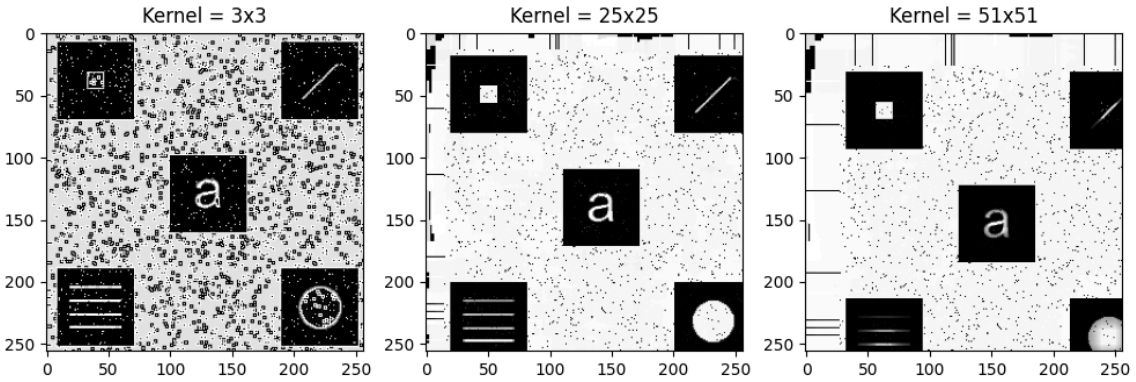
Problemas que tuvimos hasta encontrar la solución:

Las ventanas pueden tener problemas si el tamaño de M x N son pares porque no va a poder encontrar un centro de la ventana, por eso se debe elegir valores impares (debe ser una región simétrica).

Otro problema que tuvimos, es que cuando estamos cerca del borde de la imagen, no hay suficientes píxeles alrededor como para formar una ventana completa, es por eso que agregamos bordes, lo que extiende los bordes de la imagen copiando los valores más cercanos. Antes usábamos:

cv2.copyMakeBorder(img, M, N, M + 1, N + 1, cv2.BORDER_REPLICATE)

Esto estaba mal porque agregaba demasiado borde y además lo hacía de forma asimétrica, desplazando el contenido.



Es por esto que ahora utilizamos:
cv2.copyMakeBorder(img, M//2, M//2, N//2, N//2, cv2.BORDER_REPLICATE)

Usamos M // 2 y N // 2 porque esto hace que, al aplicar una ventana centrada de tamaño M x N, se pueda cubrir completamente la imagen original, incluso en los bordes.

Por ejemplo, en ventana de 3x3, el centro es el píxel de la posición (1, 1) en la ventana. Para que esta ventana esté centrada sobre cada píxel de la imagen, necesitás agregar 1 fila / columna extra en cada dirección (porque 3 // 2 = 1).

3. Problema 2 - Corrección de múltiple choice

Descripción del problema:

El problema planteado consiste en la automatización de la corrección de exámenes tipo multiple choice a partir de imágenes digitalizadas. Estos exámenes cuentan con un formato estructurado que incluye un encabezado con datos personales del alumno (como Nombre, ID, Código y Fecha) y un área principal donde se encuentran las preguntas y las opciones de respuesta.

El desafío radica en diseñar un sistema que pueda procesar las imágenes de estos exámenes y realizar las siguientes tareas de manera eficiente:

1. **Evaluar las respuestas del examen**: Detectar qué opciones fueron marcadas por el alumno y compararlas con una plantilla de respuestas correctas para determinar su desempeño.

2. **Validar los datos del encabezado**: Comprobar que los datos personales cumplan con criterios específicos de formato, como la longitud máxima permitida o la cantidad de caracteres requeridos.

3. **Procesar múltiples exámenes**: Aplicar el algoritmo a un conjunto de imágenes y generar resultados claros para cada uno.

4. **Generar un informe final**: Presentar de manera visual y resumida qué alumnos aprobaron el examen (con al menos 20 respuestas correctas) y cuáles no lo lograron.

Este problema no solo involucra la extracción de información visual mediante técnicas de procesamiento de imágenes, sino también la validación y estructuración de los datos obtenidos. Para resolverlo, se requiere aplicar métodos que combinen análisis geométrico, segmentación de imágenes,

detección de patrones y validación lógica, lo que supone un desafío tanto técnico como metodológico.

El objetivo principal del trabajo es proporcionar una solución automatizada, precisa y escalable que pueda ser utilizada en contextos educativos, mejorando la eficiencia en la corrección de exámenes y reduciendo posibles errores humanos.

Resolución del problema:

Detección y Corrección de Respuestas

Para abordar la primera consigna, se desarrolló un conjunto de funciones que permiten procesar una imagen de un examen multiple choice, segmentar las áreas de interés y determinar si las respuestas marcadas por el alumno son correctas o incorrectas. A continuación, se detalla el enfoque adoptado:

1. Lectura y carga de imágenes

La entrada al sistema son imágenes de exámenes multiple choice en formato digital.

Usamos la librería `OpenCV` para leer las imágenes en escala de grises.

La función `leer_imagen` recibe la ruta de la imagen y devuelve la versión en escala de grises, mientras que `cargar_imagenes` procesa múltiples rutas para manejar conjuntos de exámenes.

2. Segmentación de renglones de respuestas

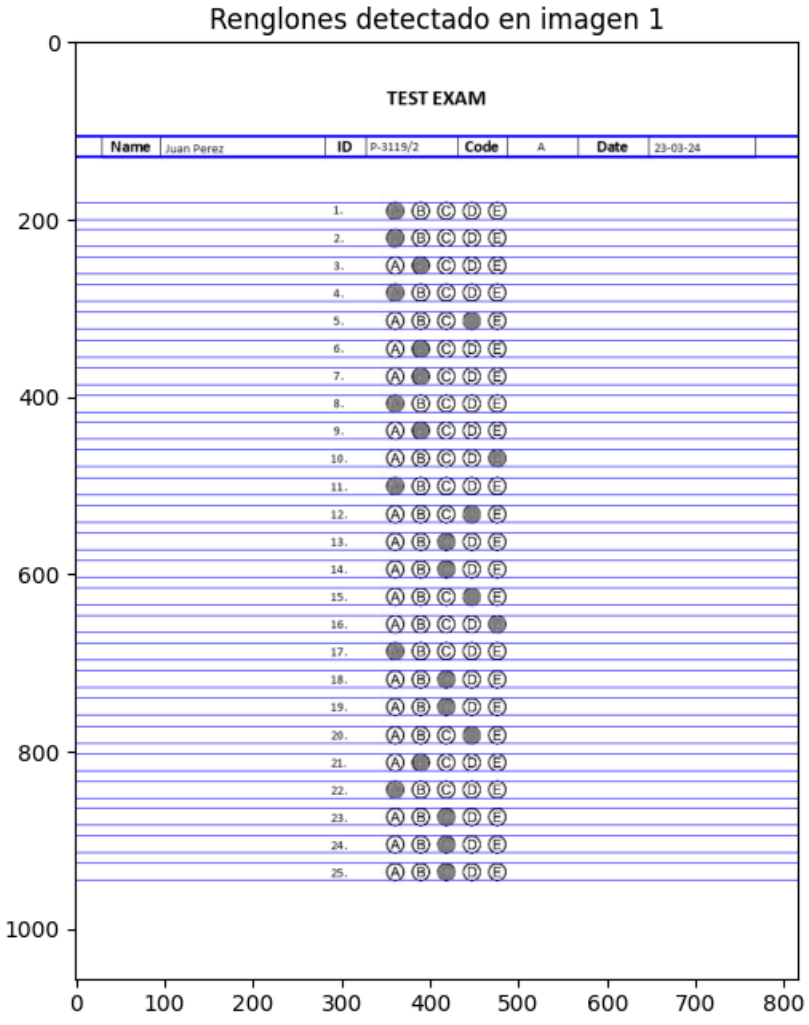
El objetivo aquí es identificar las regiones horizontales (renglones) donde se encuentran las opciones de respuesta.

Binarizamos la imagen para resaltar las marcas negras sobre el fondo blanco.

Calculamos la proyección horizontal, que suma la cantidad de píxeles oscuros en cada fila. Esto genera un perfil de densidad que permite detectar renglones con contenido relevante.

Aplicamos filtros de umbral para identificar rangos de densidad que correspondan a renglones de respuestas y descartamos líneas no relevantes (como líneas de encabezado o espacios vacíos).

Los renglones con opciones de respuesta tienen una densidad de píxeles oscuros mucho mayor que las zonas de fondo blanco o encabezados.



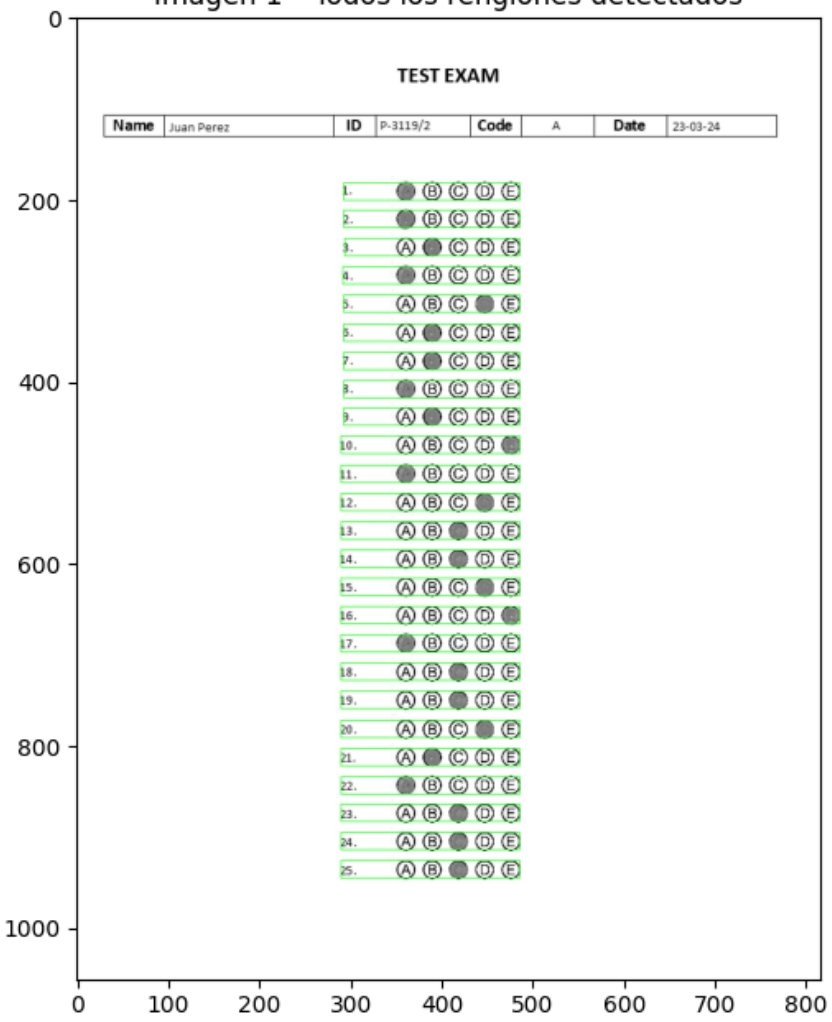
3. Delimitación de las respuestas

Una vez identificados los renglones, delimitamos las columnas donde se encuentran las opciones de respuesta en cada renglón.

Usamos máscaras binarias para detectar las columnas con píxeles oscuros dentro de cada renglón.

Determinamos los límites horizontales (inicio y fin de cada opción) calculando los índices de las primeras y últimas columnas con contenido relevante.

Imagen 1 – Todos los renglones detectados

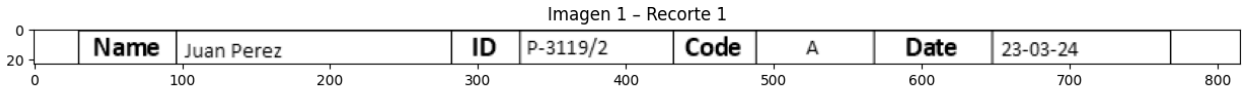


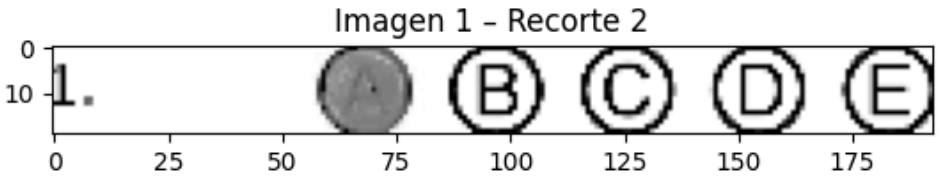
4. Recorte de regiones de interés

Con los límites de los renglones y columnas definidos, realizamos un recorte preciso de las áreas correspondientes a las opciones de respuesta.

Extraemos cada renglón como una subimagen rectangular. Esto se hace tanto para los renglones de opciones de respuesta como para el encabezado del examen.

Este enfoque reduce el tamaño de los datos a procesar, lo que mejora la velocidad y precisión en etapas posteriores.





5. Detección de círculos

Para identificar las marcas realizadas por el alumno, implementamos un método basado en la **Transformada de Hough**.

Aplicamos la función `cv2.HoughCircles`, que detecta círculos en una imagen basada en el gradiente de intensidad.

Ajustamos parámetros como el tamaño mínimo y máximo del radio de los círculos, asegurando que solo se detecten las marcas correspondientes a las opciones de respuesta.

Dibujamos los círculos detectados para verificar visualmente la precisión del algoritmo.

Las marcas realizadas en las opciones de respuesta tienen una forma circular bien definida, lo que las hace ideales para este tipo de detección.

6. Evaluación de respuestas

En esta etapa, determinamos qué opciones han sido marcadas y si cumplen con los criterios válidos.

Analizamos cada círculo detectado para calcular la cantidad de píxeles oscuros en su interior. Si un círculo contiene más de un umbral de píxeles oscuros (250), se considera marcado.

Clasificamos las respuestas según las siguientes reglas:

- **"No respondida"**: Ningún círculo está marcado.
- **Letra**: Solo un círculo está marcado, asignándole la letra correspondiente a su posición.
- **"Múltiples"**: Más de un círculo está marcado.

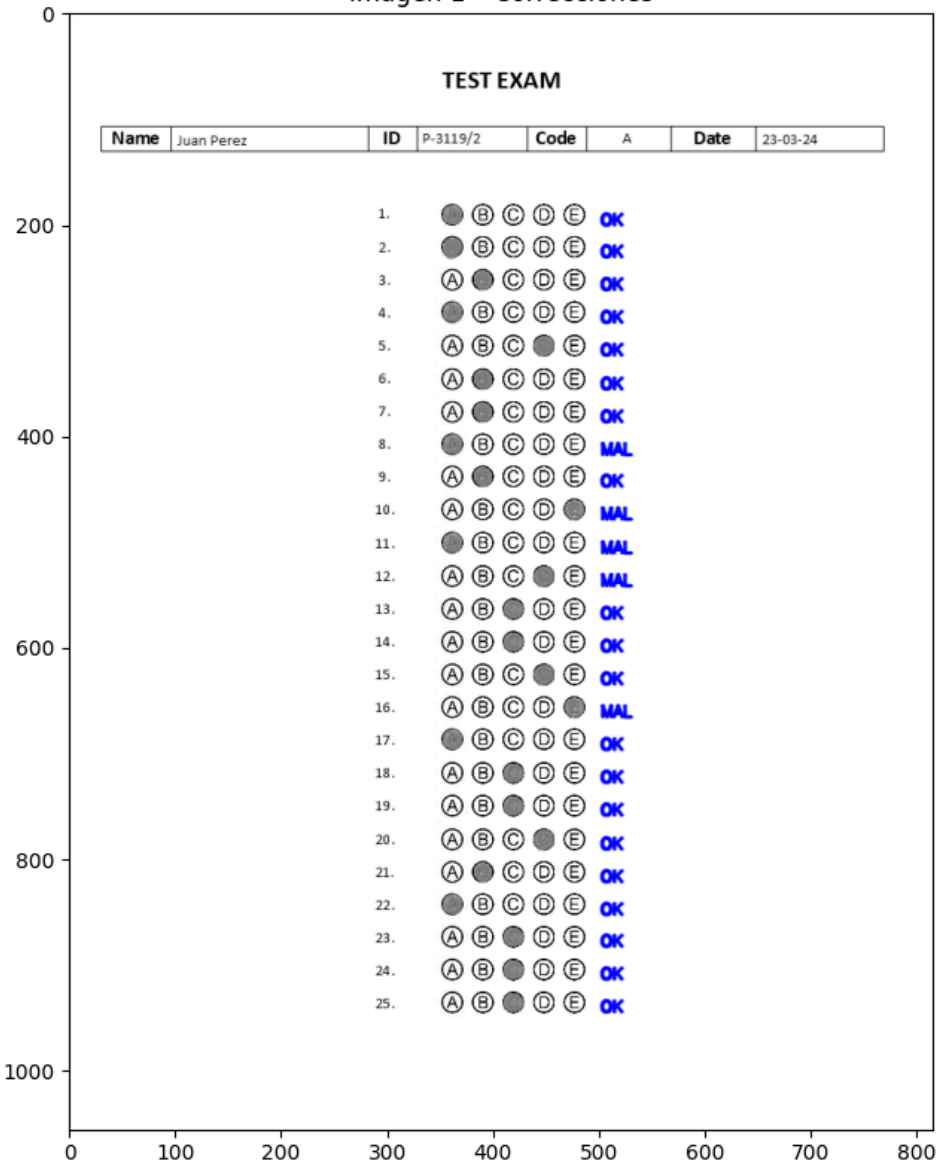
Este enfoque asegura que solo se consideren respuestas válidas y permite identificar errores como múltiples marcas en una misma pregunta.

7. Corrección

Finalmente, las respuestas detectadas se comparan con una plantilla predefinida de respuestas correctas para determinar si cada respuesta es "OK" o "MAL".

Recorrimos la lista de respuestas detectadas y la comparamos índice por índice con la plantilla de respuestas. Guardamos los resultados de la corrección para cada pregunta y cada examen.

Imagen 1 – Correcciones



El procedimiento descrito permitió analizar las imágenes de los exámenes multiple choice para determinar automáticamente las respuestas correctas e incorrectas. El uso de técnicas avanzadas como la Transformada de Hough y la segmentación basada en proyección horizontal garantizó un análisis preciso y escalable.

Validación de Datos del Encabezado

Para validar los datos del encabezado, diseñamos un proceso que asegura que cada campo cumpla con las restricciones establecidas. Este encabezado contiene cuatro campos fundamentales: **Nombre**, **ID**, **Código** y **Fecha**. Cada uno de estos datos debía ser extraído, analizado y validado de manera automática para verificar si cumplía con los criterios definidos.

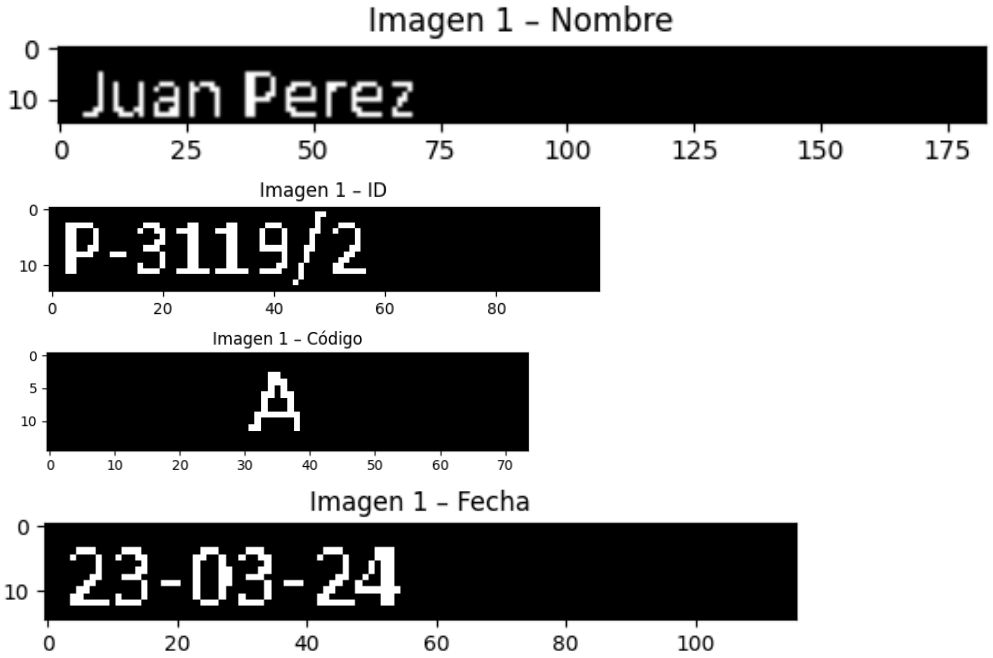
1. Extracción de los campos del encabezado

Se utiliza la función `detectar_datos_encabezado` para extraer las áreas correspondientes a cada campo dentro del encabezado. Este proceso incluye:

- **Binarización**: Convertir la imagen del encabezado a un formato binario invertido para facilitar la detección de texto.

- **Recorte de subregiones**: Con base en coordenadas predefinidas, se extraen las áreas específicas para cada campo (Nombre, ID, Código y Fecha).

Cada uno de estos recortes se almacena en una lista para su análisis posterior.



2. Análisis de los campos

Tras extraer los campos, la siguiente etapa consistió en analizar el contenido de cada uno para validar su formato y longitud. Esta tarea fue llevada a cabo por la función `analizar_datos_encabezado`. Aquí, el algoritmo examinó cada subregión recortada y detectó las letras presentes basándose en las transiciones de píxeles claros a oscuros. Al identificar los bordes de las letras, el sistema pudo calcular cuántos caracteres componían cada campo, así como cuántos espacios existían entre ellos. Esta información fue crucial para validar los datos según las restricciones establecidas.

El campo **Nombre**, por ejemplo, debía contener al menos dos palabras separadas por un espacio y no exceder los 25 caracteres en total. Para lograr esto, se contó el número de letras y espacios detectados en la subregión correspondiente. Si había al menos un espacio y la longitud total del campo no superaba el límite permitido, se consideraba válido. El campo **ID**, en cambio, debía ser una única palabra formada por exactamente 8 caracteres. Aquí, la ausencia de espacios entre letras fue un criterio fundamental para determinar su validez.

En cuanto al campo **Código**, debía ser un único carácter, y para el campo **Fecha**, se requirió que fuera una palabra de 8 caracteres sin espacios. Estas condiciones garantizaron que los datos del encabezado fueran consistentes y se ajustaran al formato esperado.

Finalmente, la función generó un estado para cada campo, mostrando si cumplía con las condiciones requeridas. Por ejemplo, un encabezado válido podría producir una salida como la siguiente:

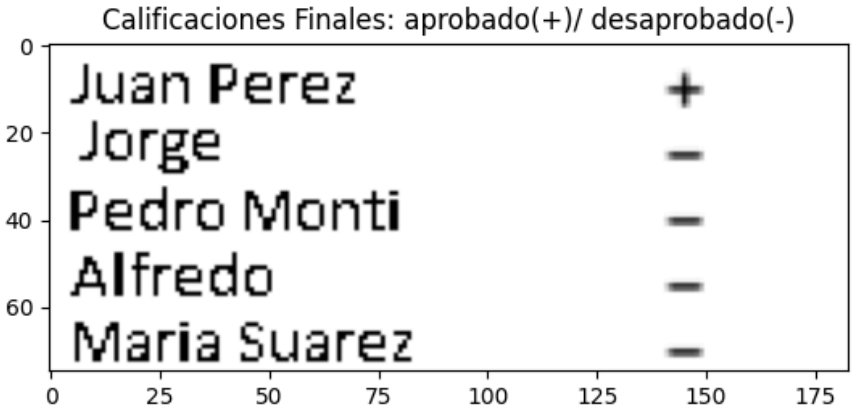
```
Nombre: OK

ID: OK

Código: MAL

Fecha: OK
```

Este proceso permitió no solo detectar errores en los datos del encabezado, sino también localizar rápidamente los campos que requerían corrección. Si bien el algoritmo fue preciso en la mayoría de los casos, algunos desafíos surgieron debido a problemas de segmentación, como caracteres mal detectados en imágenes de baja calidad. Para mitigar esto, se ajustaron los límites de las subregiones y los umbrales de binarización, lo que mejoró significativamente la precisión.

Consolidación de la imagen final

Los recortes de los nombres, ahora marcados con su indicador correspondiente, fueron combinados en una única imagen mediante la función `np.vstack`, que une verticalmente las imágenes de los nombres en un solo archivo.

Por último, si se activaba la opción `mostrar`, la imagen consolidada se visualizaba utilizando `matplotlib`. Esto permitió verificar rápidamente los resultados antes de guardar la imagen final.



4. Conclusión

En este trabajo logramos demostrar cómo, utilizando herramientas de procesamiento de imágenes y librerías de Python, es posible manipular imágenes de manera eficiente y automatizar procesos que históricamente fueron manuales.

Comenzamos con el problema de la ecualización local del histograma, donde implementamos un algoritmo que permitió resaltar detalles ocultos en una imagen mediante el ajuste adaptativo del contraste en diferentes regiones. Este ejercicio nos permitió comprender cómo parámetros como el tamaño de la ventana de procesamiento influyen en los resultados y cómo estas técnicas pueden aplicarse para mejorar la visualización de información en imágenes complejas.

Luego, en el caso del análisis de exámenes multiple choice, desarrollamos un sistema que procesa imágenes, segmenta áreas clave, valida los datos del encabezado y evalúa las respuestas automáticamente. Este enfoque permitió transformar un proceso manual en una solución automatizada, escalable y precisa, capaz de generar informes visuales que resumen el desempeño de los alumnos de manera clara y eficiente.

Ambos problemas evidencian la utilidad de las técnicas de procesamiento digital de imágenes para resolver desafíos prácticos en distintos contextos, optimizando tiempos y reduciendo errores humanos. Esto resalta el potencial de estas herramientas como soluciones modernas y efectivas.