Tecnicatura Universitaria en Inteligencia Artificial

Procesamiento de Imágenes

“TRABAJO PRÁCTICO N° 1”

Balverdi, Valentina - (B-6588/9)

Caballero, Franco - (C-7328/8)

Grimaldi, Damián - (G-5977/3)

1° Semestre - Año 2025

Problema 1 - Ecualización local de histograma

Introducción al problema

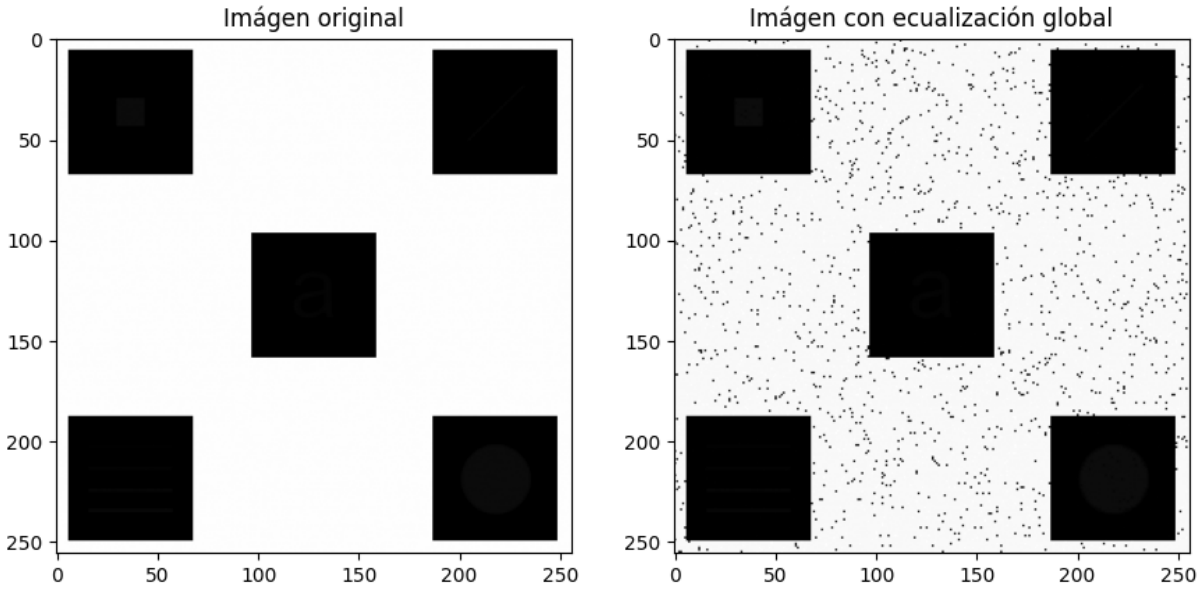
En este ejercicio se analiza una de las técnicas de transformación de intensidad aplicada a imágenes llamada **ecualización local del histograma.** Ésta técnica busca mejorar el contraste en imágenes que presentan detalles difíciles de percibir a simple vista ya que tienen niveles de gris similares al del fondo.

A diferencia de la ecualización global (que ajusta el contraste considerando toda la imagen al mismo tiempo), la ecualización local trabaja por regiones, procesando cada píxel según la distribución de intensidades en su vecindario.

El objetivo del ejercicio fue implementar la ecualización local, aplicarla a una imagen con detalles ocultos, y analizar cómo influye el tamaño y forma de la ventana utilizada en el resultado final.

Desarrollo y resolución

Se trabajó con una imagen en escala de grises la cual presenta zonas con bajo contraste y estructuras poco visibles a simple vista.
Como primer paso de referencia, se aplicó la función **cv2.equalizeHist()** sobre toda la imagen para ver los cambios. Estos fueron mínimos y los detalles finos siguieron ocultos o apenas visibles.



Comparación imagen original e imagen con la ecualización global aplicada.

Para implementar la ecualización local, se creó la función **ecualizacion_local()** que recibe como parámetros una imagen y una tupla con el tamaño de la ventana (cantidad filas, cantidad columnas).

Una de las cosas a tener en cuenta fue cómo tratar los bordes de la imagen. Como la ventana se mueve por cada píxel, en las esquinas o bordes puede quedar “afuera” de la imagen. Para evitar eso, usamos cv2.copyMakeBorder() con el modo cv2.BORDER_REPLICATE, que extiende la imagen replicando los valores de los bordes. Así siempre pudimos aplicar la ventana completa, incluso cerca de los límites.

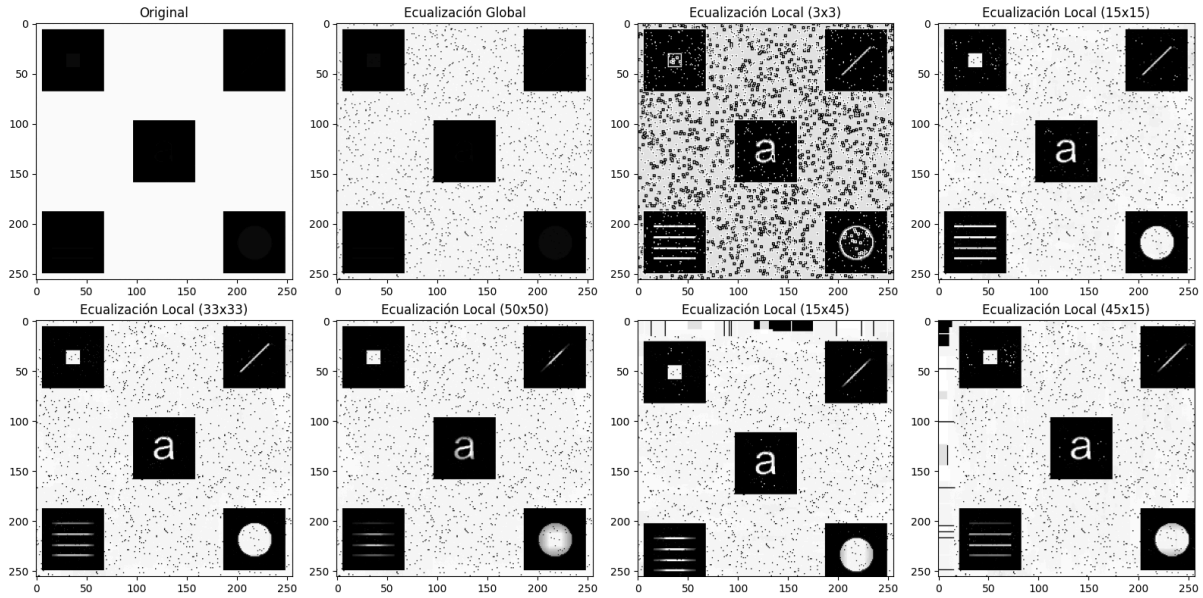
Luego, con dos bucles, se recorre cada píxel de la imágen. En cada iteración se extrae una subimagen del tamaño de la ventana (centrada en el píxel actual), se le aplica cv2.equalizeHist() a esa subimagen, se toma el valor del píxel central de la ventana ecualizada, y ese es el nuevo valor que se asigna al píxel actual en la imagen resultante.

Así se va construyendo una nueva imagen en donde cada píxel fue ajustado de acuerdo al histograma de su vecindario.

```python
def ecualizacion_local(img, tamaño_ventana) -> np.ndarray:
    """
    Realiza la ecualización local de una imagen en escala de grises utilizando un tamaño de ventana específico.
    Args:
        img (np.ndarray): Imagen de entrada en escala de grises.
        tamaño_ventana (tuple): Tamaño de la ventana para la ecualización local (m, n).
    Returns:
        np.ndarray: Imagen ecualizada.
    """

    m,n = tamaño_ventana
    bordes = max(m,n)//2
    top, bottom, left, right = bordes, bordes, bordes, bordes

    filas, columnas = img.shape

    img_copia = img.copy()

    img_bordes = cv2.copyMakeBorder(img_copia, top,bottom,left,right, cv2.BORDER_REPLICATE)

    for fila in range(filas):
        for columna in range(columnas):
            ventana = img_bordes[fila:fila+m, columna:columna+n]
            img_histograma = cv2.equalizeHist(ventana)
            img_copia[fila, columna] = img_histograma[m//2, n//2]

    return img_copia
```

Función resultante.

Finalmente implementamos esta función probando distintos tamaños de ventana, tanto cuadradas como rectangulares. A continuación se muestran los resultados comparativos:



Comparación de todas las imágenes resultantes.

Conclusión

La ecualización local del histograma fue efectiva para mejorar el contraste de forma localizada, revelando detalles que no se visualizaban en la imagen original ni con la aplicación de la ecualización global. Al aplicar una transformación de intensidad basada en el vecindario de cada píxel, se logró que ciertas zonas con bajo contraste muestren contenido que antes estaba oculto.

Durante el desarrollo notamos que el tamaño de la ventana influye en el resultado:

- Con ventanas muy chicas (como 3x3), el contraste mejora pero aparece mucho ruido.
- Con tamaños medios (como 15x15 o 33x33), se obtiene un buen equilibrio: se resaltan los detalles sin deformar tanto la imagen.
- Con ventanas grandes (como 50x50), el resultado es más suave, pero se pierde precisión en los detalles más finos.

También se probaron ventanas rectangulares, que mostraron cómo la forma de la ventana afecta la dirección en que se refuerzan los detalles: por ejemplo, una ventana 15x45 mejora más los cambios horizontales, y una 45x15 los verticales.

La ecualización local permitió analizar cómo la distribución local de niveles de gris afecta la visualización de detalles, y cómo modificar el tamaño de ventana cambia el resultado final. La resolución de éste problema ayudó a comprender mejor las transformaciones de intensidad en imágenes y su aplicación práctica.

Problema 2 - Correción de Exámen Multiple Choice

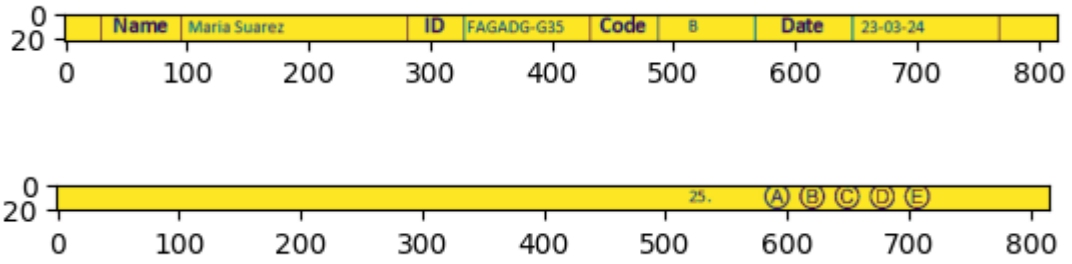
Introducción al problema

En este problema se aborda la automatización del proceso de corrección de exámenes tipo multiple choice, a partir del análisis de imágenes de las planillas de respuesta.

El objetivo es construir un script en python que, además de identificar las respuestas marcadas por el alumno, realice una validación de los campos del encabezado (Nombre, ID, Código y Fecha), y genere un imagen que diferencia a los alumnos aprobados de los desaprobados.

Desarrollo y resolución

Se parte de una imagen en escala de grises de cada examen.
El primer paso consiste en segmentar dos zonas principales, el encabezado y el conjunto de preguntas, para la posterior resolución de los puntos pedidos. Esto se logra utilizando la función **deteccion_renglones(img)**, que recibe como parámetro la imagen de un examen y detecta líneas horizontales oscuras. Devuelve una tupla con la imagen del encabezado y una lista con cada renglón de preguntas del examen.



Para detectar estos renglones se aplicó una técnica de análisis de proyección horizontal sobre imágenes binarizadas. Esto permitió identificar las líneas divisorias mediante los cambios de intensidad fila a fila.

Con el resultado obtenido, se prosiguió con el análisis del encabezado, que consiste en verificar si éste cumple con las condiciones especificadas. Para eso, se utilizó la función **segmentar_encabezado()**, que recibe la imagen del encabezado y la segmenta en celdas usando detección de bordes verticales, mediante la aplicación de un filtro espacial con máscara vertical. Luego, se analizó la proyección vertical para ubicar las líneas divisorias de las celdas. Una dificultad fue ajustar los umbrales de detección que permitan una segmentación robusta sin perder partes importantes.
Aquí se utilizó además la función **agrupar_columnas()**, ya que en varias imágenes las líneas verticales se encontraron borrosas o duplicadas, y eso dificultaba la correcta segmentación.

Posteriormente, la función **detectar_caracteres()** se aplicó a cada celda obtenida. En ella, se utilizan técnicas de umbralización, filtrado por área y etiquetado de componentes conectados, que permiten identificar los caracteres escritos. Además, se analiza la separación entre caracteres para detectar espacios (por ejemplo, para validar si el nombre tenía más de una palabra).
Nuevamente, se presentó la dificultad de ajustar cuidadosamente los umbrales para evitar que se pierdan letras o se confunda el texto con ruido.

Los datos obtenidos en esa etapa se utilizaron luego en la función **validacion_encabezado()**, que verifica si en cada tipo de campo se cumplen las condiciones definidas en el enunciado. Devuelve un string indicando si el contenido de cada campo es correcto o incorrecto.

```
name : OK
id : MAL
code : OK
date : OK
```

Una vez completado lo requerido con el encabezado, se trabajó sobre la detección y corrección de las respuestas de los exámenes.
Con la función **detectar_respuesta_marcada()**, cada pregunta se analizó por separado. Se detectan las burbujas a partir de contornos con forma circular, se filtran por área y aspecto, y se calcula el nivel de relleno interno. Se considera como marcada la burbuja con mayor cantidad de píxeles blancos. Si hay más de una opción marcada, o ninguna, también se detecta.

Una de las principales dificultades en esta etapa fue ajustar el umbral relativo (85% del máximo) de forma que funcione bien tanto para burbujas claramente marcadas como para marcas tenues o parciales. También se debieron evitar falsas detecciones causadas por sombreado de fondo o ruido.

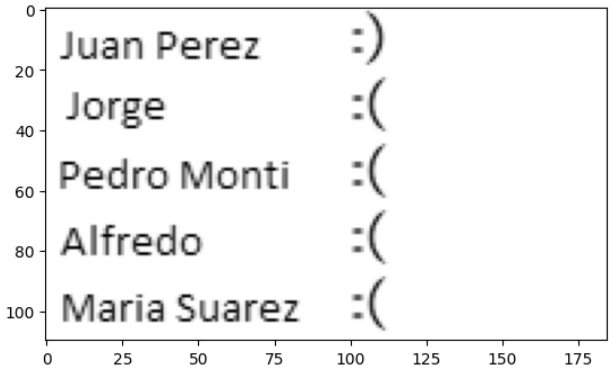
Luego, con la lista de respuestas detectadas para cada pregunta, se utiliza la función **validacion_respuestas()**,que compara cada respuesta marcada con las respuestas correctas provistas por el enunciado. Esta función evalúa si hay una única opción marcada y si esta coincide con la respuesta correcta. En caso afirmativo, la marca como correcta y aumenta el contador. Si hay más de una respuesta o ninguna marcada, se considera como incorrecta. Además, imprime por consola si cada pregunta fue respondida correctamente o no.

```
Pregunta 7: OK
Pregunta 8: MAL
Pregunta 9: OK
Pregunta 10: MAL
Pregunta 11: MAL
Pregunta 12: MAL
```

Con el total de respuestas correctas obtenidas, se determina si el examen está aprobado o no. Según el enunciado, una calificación se considera aprobada si el alumno tiene al menos 20 respuestas correctas.

A medida que se procesan las distintas imágenes de examen, se almacenan los nombres de los alumnos junto con su estado (aprobado o desaprobado). Estos nombres fueron obtenidos previamente del encabezado, y se conservaron en forma de recortes de imagen para usarlos en el informe final.

Finalmente, se genera una imagen resumen con la función **informe_final()**. Esta imagen concatena los recortes de nombres en una sola figura, y se le agrega al lado un indicador **:)** si el alumno aprobó o **:(** si desaprobó. La imagen se guarda y muestra en pantalla.



Conclusión

El script realizado permitió resolver el problema propuesto. Se combinaron técnicas de procesamiento de imágenes, como umbralización, detección de bordes, proyecciones horizontales y verticales, filtrado espacial, detección de contornos y etiquetado de componentes conectados, para abordar tanto el análisis del encabezado como la detección de respuestas, logrando así, automatizar el proceso de corrección de exámenes multiple choice.

Conclusión final

La realización del Trabajo Práctico N°1 permitió aplicar las técnicas de procesamiento de imágenes trabajadas en clase abordando dos problemas con características y objetivos diferentes: la mejora del contraste en imágenes mediante técnicas de ecualización local del histograma, y la corrección automatizada de exámenes multiple choice a partir del análisis de imágenes escaneadas.

En ambos casos se trabajó con imágenes en escala de grises y se aplicaron técnicas como umbralización, filtrado espacial, detección de bordes y etiquetado de componentes conectados, integrando también el uso de bibliotecas como Numpy, OpenCV y Matplotlib.

En el primer problema, se logró implementar una función de ecualización local capaz de resaltar detalles ocultos en distintas regiones de una imagen, y se analizó el efecto del

tamaño de ventana sobre el resultado. Este ejercicio permitió profundizar en el manejo de máscaras, bordes, padding y operaciones por ventana móvil.

En el segundo problema, se desarrolló un sistema completo para la corrección automática de exámenes, desde la segmentación de encabezado y preguntas hasta la validación de respuestas y la generación de un informe visual. Este desafío implicó un trabajo más complejo de segmentación, detección de bordes y validación de condiciones lógicas, enfrentando además variabilidad en las imágenes.

A lo largo del trabajo, se presentaron dificultades técnicas como líneas borrosas, detección parcial de caracteres, líneas o burbujas, que fueron resueltas mediante el ajuste de umbrales, pruebas y diseño del código.

finalmente, el trabajo no solo permitió consolidar conceptos teóricos vistos, sino también ejercitar la resolución práctica de problemas reales utilizando estas herramientas ahora conocidas.