PROCESAMIENTO DIGITAL DE IMÁGENES I

Trabajo Práctico N°1

Año 2024 - Primer Semestre

Grupo 8:
Cancio José
García Julián
Herrera Francisca

INTRODUCCIÓN

Este trabajo práctico consiste en la resolución de dos problemas relacionados con el procesamiento de imágenes.

PROBLEMA 1 - Ecualización local de histograma

Enunciado

La técnica de ecualización del histograma se puede extender para un análisis local, es decir, se puede realizar una ecualización local del histograma. El procedimiento sería definir una ventana cuadrada o rectangular (vecindario) y mover el centro de la ventana de pixel a pixel. En cada ubicación, se calcula el histograma de los puntos dentro de la ventana y se obtiene de esta manera, una transformación local de ecualización del histograma. Esta transformación se utiliza finalmente para mapear el nivel de intensidad del pixel centrado en la ventana bajo análisis, obteniendo así el valor del pixel correspondiente a la imagen procesada. Luego, se desplaza la ventana un pixel hacia el costado y se repite el procedimiento hasta recorrer toda la imagen.

Esta técnica resulta útil cuando existen diferentes zonas de una imagen que poseen detalles, los cuales se quiere resaltar, y los mismos poseen valores de intensidad muy parecidos al valor del fondo local de la misma. En estos casos, una ecualización global del histograma no daría buenos resultados, ya que se pierde la localidad del análisis al calcular el histograma utilizando todos los pixels de la imagen.

Desarrolle una función para implementar la ecualización local del histograma, que reciba como parámetros de entrada la imagen a procesar, y el tamaño de la ventana de procesamiento (M x N). Utilice dicha función para analizar la imagen que se muestra en Fig. 1 e informe cuales son los detalles escondidos en las diferentes zonas de la misma. Analice la influencia del tamaño de la ventana en los resultados obtenidos.

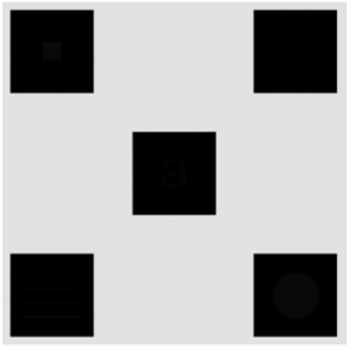


Figura 1 - Imagen con detalles en diferentes zonas.

AYUDA:

Con la siguiente función, puede agregar una cantidad fija de pixels a una imagen: cv2.copyMakeBorder(img, top, bottom, left, right, borderType), donde top, bottom, left y right son valores enteros que definen la cantidad de pixels a agregar arriba, abajo, a la izquierda y a la derecha, respectivamente, y borderType define el valor a utilizar.
Por ejemplo, borderType= cv2.BORDER_REPLICATE replica el valor de los bordes.

Comentarios sobre la resolución

El código con la resolución del problema se encuentra en el archivo tp1_pdi_ej1.py, dentro de la carpeta del repositorio.

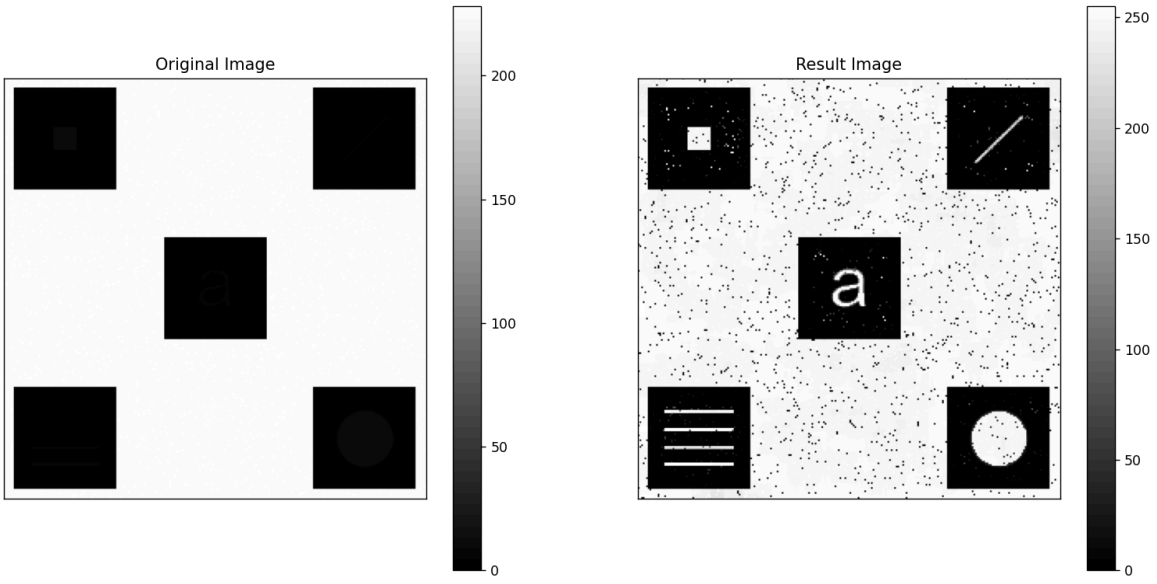
El mismo implementa una función para realizar ecualización local del histograma en una imagen, buscando mejorar el contraste y la distribución de intensidades en la imagen dada.

Primero, se define una función imshow para mostrar imágenes, con la opción de elegir si mostrar la imagen en una nueva figura, con título, en escala de grises o a color, entre otras opciones.

Luego, se define la función ecual_local_hist, que aplica ecualización local del histograma a nuestra imagen. Esta función recibe como entrada la imagen y el tamaño del kernel que se utilizará para definir la región de interés (ROI) en la que se aplicará la ecualización. La ecualización se realiza recorriendo la imagen píxel por píxel, calculando la ecualización del histograma para la ROI correspondiente y asignando el valor ecualizado al píxel central de la ROI en la imagen resultante.

Después de definir las funciones, se define el tamaño del kernel y se aplica la ecualización local del histograma a la imagen a tratar, llamada ‘Imagen_con_detalles_escondidos.tif’.

Finalmente, se muestra la imagen original y la imagen resultante en una misma figura para comparar los resultados de la ecualización. Este proceso nos ha permitido descubrir los objetos ‘escondidos’ en la imagen original, como se puede apreciar a continuación:



PROBLEMA 2 -Corrección de multiple choice

Enunciado

El problema planteado consiste en crear un algoritmo que permita corregir en forma automática una serie de evaluaciones de tipo multiple choice.

En la Figura 2 se muestra el esquema de una plantilla de respuestas del examen de 25 preguntas con cinco opciones para cada una de ellas (A, B, C, D, E). La plantilla también tiene un encabezado con cuatro campos para completar datos personales (Name, ID, Code y Date).

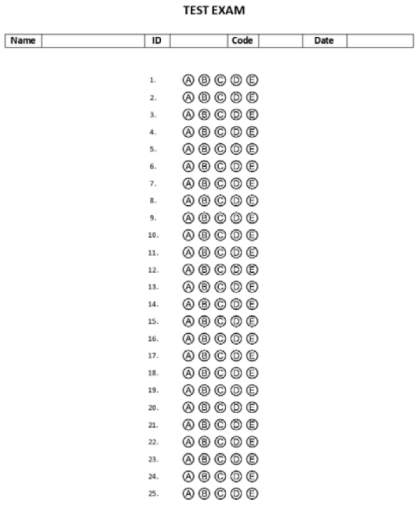


Figura 2 - Esquema del examen

Se tiene una serie de exámenes resueltos, en formato de imagen, y se pretende corregirlos de forma automática por medio de un script en python. Para esto, asuma que las respuestas correctas son las siguientes:
1. A 2. A 3. B 4. A 5. D 6. B 7. B 8. C 9. B 10. A 11. D 12. A 13. C 14. C 15. D 16. B 17. A 18. C 19. C 20. D 21. B 22. A 23. C 24. C 25. C

En el caso que alguna respuesta tenga marcada más de una opción, la misma se considera como incorrecta, de igual manera si no hay ninguna opción marcada.

El algoritmo a desarrollar debe resolver los siguientes puntos:

a. Debe tomar únicamente como entrada la imagen de un examen y mostrar por pantalla cuáles de las respuestas son correctas y cuáles incorrectas. Por ejemplo:

Pregunta 1: OK
Pregunta 2: MAL
Pregunta 3: OK
.....
Pregunta 25: OK

b. Con la misma imagen de entrada, validar los datos del encabezado y mostrar por pantalla el estado de cada campo teniendo en cuentas las siguientes restricciones:

i. Name: debe contener al menos dos palabras y no más de 25 caracteres.
ii. ID: 8 caracteres formando una sola palabra.
iii. Code: un único caracter.
iv. Date: 8 caracteres formando una sola palabra.

Por ejemplo:

Name: OK
ID: OK
Code: MAL
Date: MAL

Asuma que todos los campos ocupan un solo renglón y que se utilizan caracteres alfanuméricos, guión medio " - " y barra inclinada " / ".

En la Figura 3a se muestra un ejemplo donde los campos del formulario están todos correctamente cargados, mientras que en la Figura 3b se muestra otro ejemplo donde todos los campos están mal cargados.

TEST EXAM

Name	Juan Perez	ID	P-3119/2	Code	A	Date	23-03-24

Figura 3a - Todos los campos bien cargados

TEST EXAM

Name	Jorge	ID	X45GBK 0755	Code		Date	23-03

Figura 3b - Todos los campos mal cargados

c. Utilice el algoritmo desarrollado para evaluar las imágenes de exámenes resueltos (archivos multiple_choice_x.png) e informe los resultados obtenidos.

d. Generar una imagen de salida informando los alumnos que han aprobado el examen (con al menos 20 respuestas correctas) y aquellos alumnos que no. Esta imagen de salida debe tener los "crop" de los campos Name del encabezado de todos los exámenes del punto anterior y diferenciar de alguna manera aquellos que corresponden a un examen aprobado de uno desaprobado.

AYUDAS:

1) Existen varias formas de detectar las celdas donde se marcan las respuestas, una de ellas es detectando las coordenadas de las líneas verticales y horizontales que alinean dichas celdas. Para ello, una opción es primero umbralar la imagen img_th = imgth_row. Tenga en cuenta que las líneas pueden tener más de un pixel de ancho, por lo cual, quizás deba encontrar el principio y el fin de las mismas en la variable img_rows_th. Esta misma técnica se puede utilizar para detectar las líneas del encabezado y obtener subimagenes de los campos del mismo.

2) Con las subimagenes de los campos detectados, una posible forma de obtener los caracteres dentro de los mismos, es obteniendo las componentes conectadas dentro: cv2.connectedComponentsWithStats(celda_img, 8, cv2.CV_32S). Tenga especial cuidado que no hayan quedado pixels de las líneas divisorias de la tabla dentro de la celda. Una posible forma de evitar este problema, es eliminar las componentes conectadas de área muy chica, definiendo un umbral: ix_area = stats[:,-1]>th_area y luego stats = stats[ix_area,:].

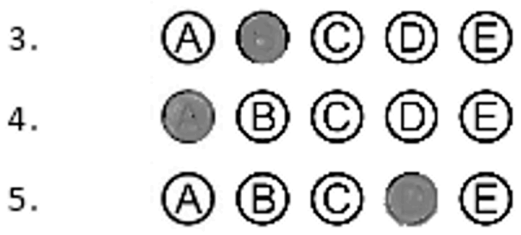
Comentarios sobre la resolución

Detección de respuestas

El código con la resolución del problema se encuentra en el archivo tp1_pdi_ej2.py, dentro de la carpeta del repositorio.

Se realizan a continuación algunos comentarios con el desarrollo del código y las dificultades presentadas.

En primer lugar se analiza cómo detectar cuándo una opción es elegida y cuándo no.
Por ejemplo en la siguiente imagen la opción elegida para la pregunta 3 es la B, para la 4 la A y la 5 la D.
En cualquier caso la opción elegida siempre va a estar rellena en gris.



Sabiendo esto analizamos el valor promedio de los píxeles que conforman cada una de las ventanas de color verde según se puede observar a continuación.

Se observa que para las opciones marcadas dicho valor oscila alrededor de 143, y que para las opciones no marcadas varía en torno a 175 (ya sea A, B, C, D o E).

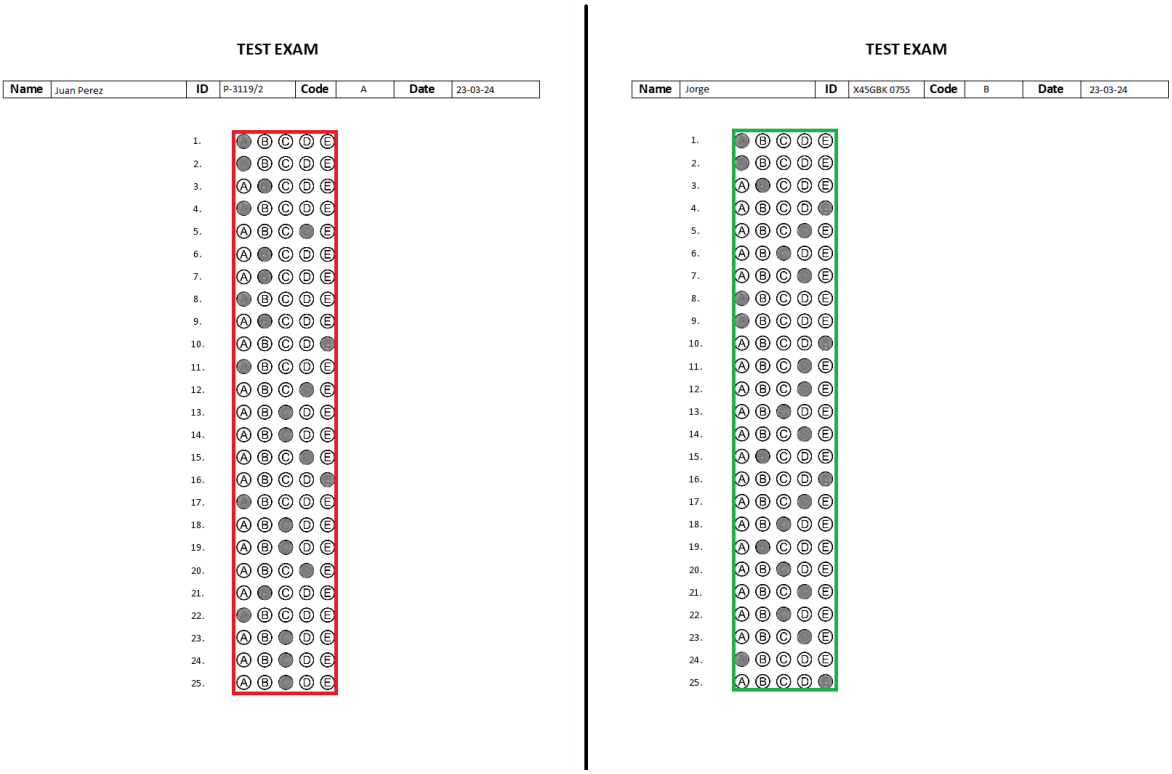
El tamaño de las ventanas es fijo, de 21x21 píxeles, de manera que los círculos queden circunscriptos.
Podemos entonces detectar cuáles son las respuestas elegidas.

Como la separación entre las opciones es regular, podemos establecer un paso que permita mediante un bucle doble guardar los valores en una lista de listas llamada lista_respuestas.

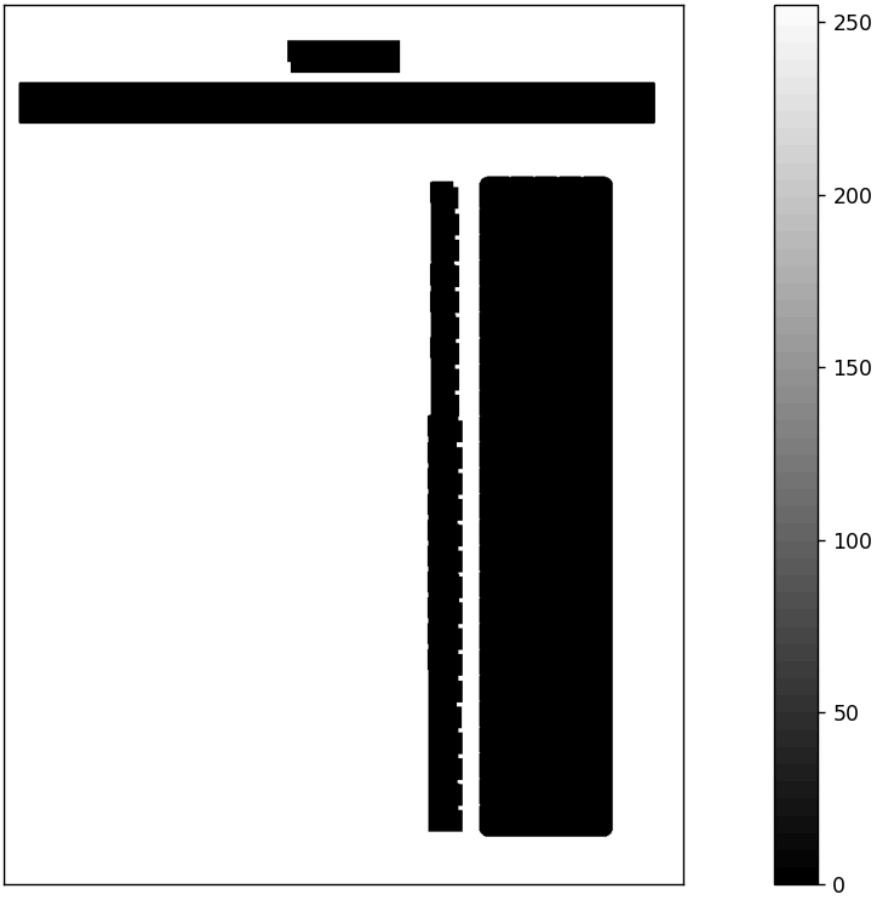
Como son 25 las preguntas y 5 las opciones, lista_respuestas será una lista de 25 sublistas, donde cada sublista tendrá 5 elementos, que pueden ser 0 si la opción no fue elegida, o 1 si lo fue.

Localización de las opciones

Se observa en la siguiente imagen que la posición de las opciones varía según el examen. Por lo tanto debemos determinar a partir de qué posición (r,c) debemos comenzar el bucle doble para detectar las respuestas, es decir, cuál es la coordenada de la esquina superior izquierda de los rectángulos rojo y verde.



La distribución de los datos dentro de la hoja del examen nos permite pensar que puede aplicarse una transformación a cada imagen de manera de obtener lo siguiente:



De esta manera podremos identificar cuál es el objeto que corresponde a las opciones y de esta manera obtener sus coordenadas.

Para generar la imagen anterior se crea la función unir(image, window_size=(25,25)), donde se binariza la imagen que se ingresa como parámetro, y luego se le aplica un kernel (por defecto de 25x25 píxeles), el cual asigna un 0 (negro) al píxel analizado si existe al menos un píxel oscuro dentro de la región de interés (en adelante roi). El tamaño del kernel por defecto es el que permite conseguir la unión de los datos en objetos como se ve en la imagen anterior.

Habiendo hecho esto queda por encontrar las coordenadas del objeto que contiene las opciones, lo cual se realiza creando la función find_blocks(image), que utiliza la función cv2.connectedComponentsWithStats.

La función cv2.connectedComponentsWithStats devuelve entre otros datos las coordenadas de la caja contenedora de los objetos que encuentra, y también las áreas de los objetos.

Si bien sabemos que la posición del objeto de las opciones varía según el examen, el área no lo hará o las variaciones serán mínimas. Además, los demás objetos de la hoja presentan áreas muy menores en comparación con el área del objeto de las opciones. Todo esto nos sirve para obtener las coordenadas del objeto 'opciones', al poder identificarlo a través del valor de su área, el cual se encontrará dentro de un rango específico diferente al resto de los objetos.

Se escribe en código lo antes desarrollado dentro de la función respuestas(image), que retorna la lista 'lista_respuestas'.

Control de las respuestas

Se crea una lista de listas llamada 'lista_correctas', análoga a 'lista_respuestas', pero que como su nombre lo indica contiene las opciones que son correctas, marcadas con un 1.

De esta manera es simple comparar entre las sublistas de cada lista, y determinar si las respuestas son correctas.

Este método evita realizar validaciones adicionales como controlar que no haya más de una respuesta elegida o que no se haya elegido ninguna.

Se crea la función check_ans_1(image), la cual contiene la lista 'lista_correctas', y llama a la función respuestas(image). La función retorna la impresión por pantalla de las respuestas, de la siguiente forma:

Pregunta 1: OK
Pregunta 2: MAL
Pregunta 3: OK
.....
Pregunta 25: OK

Validación del encabezado

Utilizamos nuevamente la función cv2.connectedComponentsWithStats para obtener el número de caracteres y palabras de name, id, code y date.

Como el encabezado ocupa siempre la misma posición en todas las hojas de examen, podemos establecer de manera fija las posiciones para obtener las subimágenes de name, id, code y date.

Para determinar el número de caracteres simplemente se aplica la función cv2.connectedComponentsWithStats sobre la imagen binarizada. Para evitar que considere a los puntos o acentos como objetos se establece un umbral th_area = 2 píxeles cuadrados.

En cambio, para determinar el número de palabras, se aplica la función cv2.connectedComponentsWithStats sobre la imagen transformada con la función unir, y con un kernel de 3x3. De esta manera se logra unir las letras que conforman una palabra.
Se muestra a continuación la imagen que retorna unir(imagen,(3,3)), cuando la subimagen es Juan Perez.

Se escribe en código lo desarrollado dentro de la función valida_header(image), que retorna una salida del siguiente tipo:

Name: OK
ID: OK
Code: MAL
Date: MAL

Función general

Finalmente se crea una función general llamada corrector, que tiene como parámetro el nombre de la imagen con el examen que queremos corregir, la cual llama a las funciones valida_header y check_ans_1, y retorna las salidas vistas anteriormente.

En conclusión se obtiene una función robusta ya que permite adaptarse a todas las variantes de exámenes, comprobándose el 100% de efectividad en las validaciones y correcciones.

Generación de imagen de salida

Para construir la imagen de salida necesitábamos tanto los recortes de los nombres de los alumnos como el conteo de las respuestas correctas por evaluación, con un requisito posterior que valide que la condición sea ‘Aprobado’ si el número de respuestas correctas es de 20 o superior (de lo contrario, el alumno desaprueba el examen).
La función procesar_y_mostrar_resultados recibe una lista con los nombres de las evaluaciones de los alumnos (llamada file_list) y es la que se encarga de llamar a las demás funciones: obtiene tanto los resultados de los exámenes como los recortes de los nombres, e imprime todo.

El proceso se lleva a cabo en varias etapas:

Primero, se procesan las imágenes de los exámenes para determinar cuántas respuestas correctas tiene cada estudiante (a través de la función iterativa procesar_examenes), que a su vez hace uso de check_ans_2 para retornar el número de respuestas correctas por examen:

```python
def procesar_examenes(file_list):
    resultados_examenes = {}  # diccionario contenedor de los resultados de los exámenes

    for file in file_list:
        # cargamos la imagen
        imagen = cv2.imread(file, cv2.IMREAD_GRAYSCALE)
        num_correctas = check_ans_2(imagen)  # check_ans recibe la imagen
        nombre_alumno = file.split('.')[0]  # obtenemos nombre del alumno
        resultados_examenes[nombre_alumno] = num_correctas

    return resultados_examenes
```

Luego, se extraen los nombres recortados de las imágenes (haciendo uso de la función llamada extract_names) y se imprimen junto con una etiqueta que indica si el estudiante aprobó o desaprobó el examen, según el número de respuestas correctas. Esto último se lleva a cabo mediante la implementación de la función llamada print_cropped_names_with_condition, que recibe como argumentos la lista de imágenes de los nombres recortados y los resultados de los exámenes.

En detalle, lo que hace la función print_cropped_names_with_condition es calcular el tamaño necesario de la imagen resultante basándose en el número de nombres y el ancho del nombre más largo, más un espacio adicional para la etiqueta de condición. Luego, recorre los nombres recortados y para cada uno, concatena verticalmente su región en la imagen resultante.

Mientras se realiza esta concatenación, la función también agrega la etiqueta de condición (Aprobado o Desaprobado) en función del número de respuestas correctas obtenidas por el estudiante, consultando el diccionario ’resultados_examenes’ para esto: el mismo contiene pares clave-valor donde la clave es el nombre del archivo (que se asume como el nombre del alumno) y el valor es el número de respuestas correctas obtenidas por ese alumno en su examen respectivo.

Finalmente, muestra la imagen resultante que contiene los nombres de los estudiantes y su estado de aprobación. La imagen se muestra utilizando la biblioteca matplotlib, con algunas configuraciones para ajustar el tamaño y desactivar los ejes.

En lo que respecta a las dificultades presentadas en este punto: Inicialmente, intentamos reutilizar la función check_ans_1 para determinar si un estudiante aprobaba o no el examen, pero la misma devolvía por pantalla detalles innecesarios de cada pregunta para todos los alumnos. Por este motivo es que definimos crear check_ans_2, que solo cuenta respuestas correctas, simplificando el proceso y mejorando la legibilidad del código.

Conclusiones

PROBLEMA 1

Nuestra implementación del código ha mejorado notablemente la imagen de entrada mediante el ajuste preciso del tamaño de la ventana de procesamiento (kernel), resaltando así los detalles ocultos en diversas áreas de la imagen. Sin embargo, es importante destacar que, aunque la ecualización local puede realzar la visualización de los detalles, también puede llevar a una amplificación del ruido en ciertas regiones y una pérdida de contraste si no se aplica debidamente.

Durante nuestro análisis, observamos que la modificación del tamaño del kernel podría generar problemas, como la aparición de bloques negros no deseados (ruido) o la reducción del contraste en los detalles ocultos. Por ende, la adaptación precisa del tamaño del kernel fue fundamental para optimizar el proceso y obtener buenos resultados para esta imagen específica.

Esto destaca la importancia de ajustar los parámetros según la situación o el problema en cuestión.

PROBLEMA 2

El código desarrollado demostró ser efectivo bajo las condiciones establecidas, mostrando robustez incluso ante cambios en la posición de las soluciones. No obstante, se identifican ciertos riesgos que podrían comprometer su desempeño, como la alteración en la posición del encabezado, la sustitución del grisado por una cruz para marcar la opción correcta, o incluso variaciones en la intensidad del sombreado. Estos aspectos constituyen posibles escenarios donde el algoritmo no funcionaría correctamente.