TRABAJO PRÁCTICO N° 1 - Año 2025 - 1° Semestre

Problema 1 - Ecualización local de histograma

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 consiste en definir una ventana cuadrada o rectangular (vecindario) y mover el centro de la misma de píxel en píxel. 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 píxel centrado en la ventana bajo análisis, obteniendo así el valor del píxel correspondiente a la imagen procesada. Luego, se desplaza la ventana un píxel 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 píxeles 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 (*MxN*). Utilice dicha función para analizar la imagen que se muestra en la Figura 1 e informe cuáles son los detalles escondidos en las diferentes zonas de la misma. Luego, desarrolle un análisis sobre la influencia del tamaño de la ventana en los resultados obtenidos.

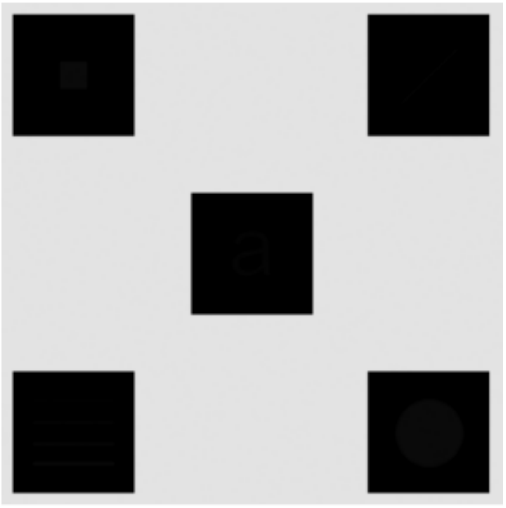

Figura 1: Imagen con detalles ocultos en diferentes zonas.

AYUDA: Con la función **cv2.copyMakeBorder(img, top, bottom, left, right, borderType)**, puede agregar una cantidad fija de píxeles a una imagen, donde *top*, *bottom*, *left* y *right* son valores enteros que definen la cantidad de píxeles a agregar arriba, abajo, a la izquierda y a la derecha, respectivamente. Por otro lado, *borderType* define el valor a utilizar. Por ejemplo, si se utiliza **borderType = cv2.BORDER_REPLICATE**, se replicará el valor de los bordes.

Problema 2 - Correción de Examen Multiple Choice

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

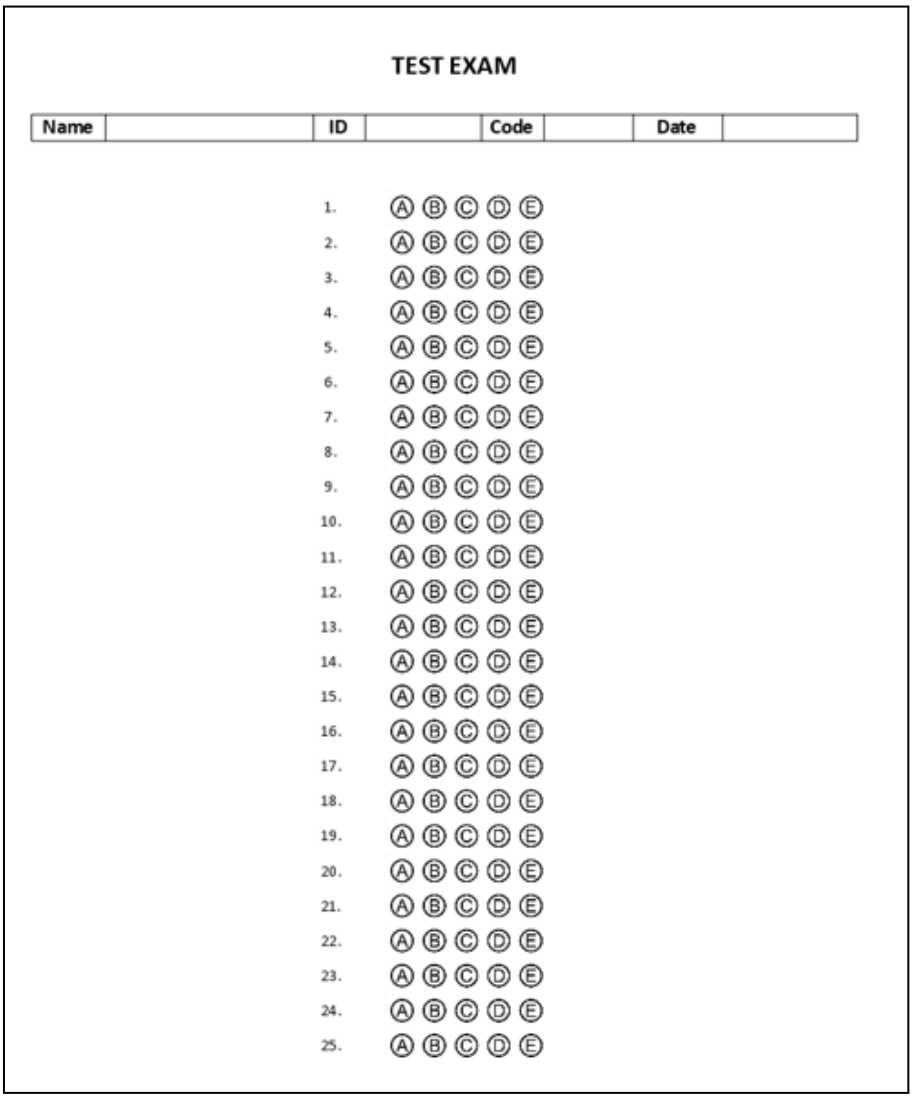

Figura 2: Ejemplo del esquema de un exámen *multiple choice* sin resolver.

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 ello, debe asumir 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 de que alguna respuesta tenga marcada más de una opción, la misma se considera como incorrecta, de igual manera para el caso de que no haya ninguna opción marcada.

El algoritmo a desarrollar debe considerar y resolver los siguientes puntos:

A. Se debe tomar únicamente como entrada la imagen de un examen y mostrar por pantalla cuáles de las **respuestas** son **correctas** y cuáles son **incorrectas**. Por ejemplo, en el terminal se debe visualizar:

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

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

a. **Name:** Debe contener al menos dos palabras y no más de 25 caracteres en total.
b. **ID:** Debe contener sólo 8 caracteres en total, formando una única palabra.
c. **Code:** Debe contener un único caracter.
d. **Date:** Debe contener sólo 8 caracteres en total, formando una única palabra.

Por ejemplo, en el terminal se debe visualizar:

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

Asuma que todos los campos ocupan un único renglón y que se utilizan tanto caracteres alfanuméricos como guión medio *" - "* y barra inclinada *" / "*.

En la Figura 3a se muestra un ejemplo en donde los campos del encabezado 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: Ejemplo de encabezado en donde todos los campos se encuentran bien cargados.

TEST EXAM

Name	Jorge	ID	X45GBK 0755	Code		Date	23-03

Figura 3b: Ejemplo de encabezado en donde hay campos que se encuentran mal cargados.

C. Se debe aplicar el algoritmo desarrollado, de **forma cíclica**, sobre el conjunto de cinco imágenes de cada examen *multiple choices* resuelto (archivos multiple_choice_<id>.png) e informar su desempeño junto con los resultados obtenidos.

D. Se debe generar una imagen de salida que informe aquellos alumnos que han aprobado el examen (con al menos **20 respuestas correctas**) y aquellos alumnos que no lo lograron. Dicha imagen de salida debe contar con el *"crop"* del contenido del campo *Name* del encabezado de cada examen del punto anterior, junto con algún indicador que diferencie a aquellos que correspondan a un examen aprobado de aquellos que se encuentren desaprobados.

AYUDAS:

- Existen varias formas de detectar las celdas donde se marcan las respuestas, una de ellas es mediante la detección de las coordenadas de las líneas verticales y horizontales que alinean dichas celdas. Para ello, una opción posible consiste en umbralizar la imagen con **img_th = img < th** y luego sumar el valor de los píxeles que están en cada columna para detectar las líneas verticales con **img_cols = np.sum(img_th_ones, 0)** y sumar el valor de los píxeles que están en cada fila para detectar las líneas horizontales con **img_rows = np.sum(img_th_ones, 1)**. Luego, dado que en dichas líneas existen muchos más píxeles que en las demás partes del formulario, se puede definir un umbral acorde (uno para las líneas horizontales y otro para las líneas verticales) y de esta forma detectar las posiciones de las mismas. Por ejemplo, utilizando **img_rows_th = mg_rows > th_row**. Tenga presente que las líneas pueden tener más de un píxel de ancho, por lo cual, quizás deba encontrar el principio y el fin de las mismas en **img_rows_th**. A su vez, considere que esta técnica puede ser utilizada para detectar las líneas del encabezado y así obtener sub-imágenes de cada uno de sus campos.

- A partir de las sub-imágenes de los campos detectados, una posible forma de obtener cada caracter dentro de ellos, consiste en la obtención de las componentes conectadas con **cv2.connectedComponentsWithStats(celda_img, 8, cv2.CV_32S)**. Sin embargo, tenga especial cuidado de que no hayan quedado píxeles de las líneas divisorias de la tabla dentro de dicha celda. Una posible forma de evitar este inconveniente, es con la eliminación de aquellas componentes conectadas que posean un área muy pequeña, mediante la definición de un umbral con **ix_area = stats[:, -1] > th_area** y luego aplicar un filtrado con **stats = stats[ix_area, :]**.