PROCESAMIENTO DIGITAL DE IMÁGENES I
Trabajo Práctico N°1
Año 2024 - Primer 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 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.

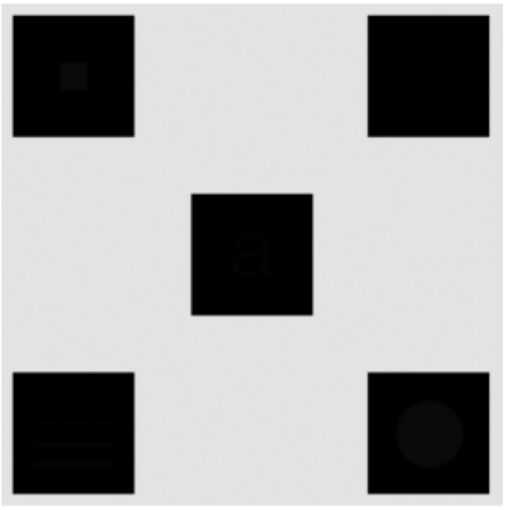

Figura 1 - Imagen con detalles en diferentes zonas.

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.

PROBLEMA 2 - Corrección de 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, E). La plantilla también tiene un encabezado con cuatro campos para completar datos personales (Name, ID, Code y Date).

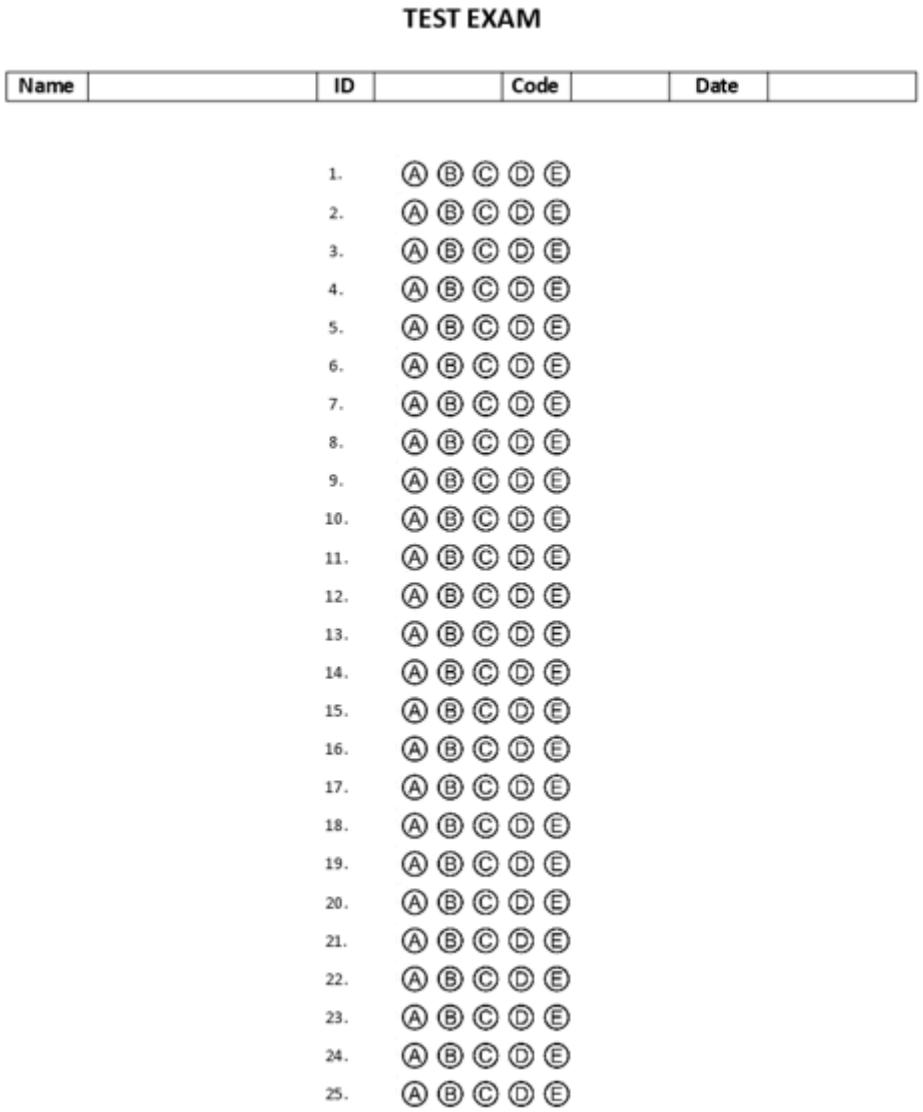

Figura 2 - Esquema del exámen.

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 = img<th** y luego sumar el valor de los pixels que están en cada columna para detectar las líneas verticales **img_cols = np.sum(img_th_ones,0)** y sumar el valor de los pixels que están en cada fila para detectar las líneas horizontales **img_rows = np.sum(img_th_ones,1)**. Luego, dado que en dichas líneas existen muchos más pixels 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 detectar así las posiciones de las mismas. Por ejemplo: **img_rows_th = mg_rows>th_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,:]**.