PROCESAMIENTO DIGITAL DE IMÁGENES I
Trabajo Práctico Nº1
Año 2023

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.

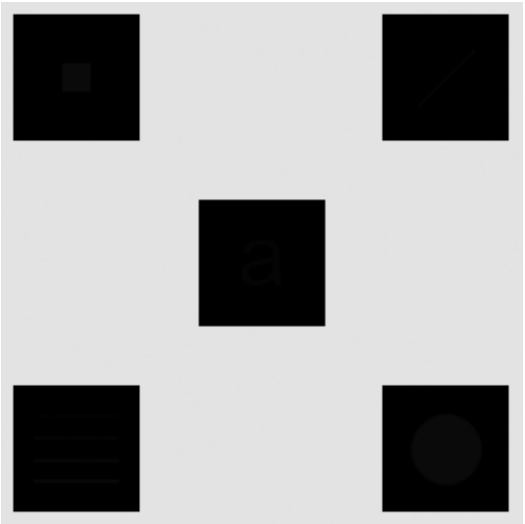

Figura 1 - Imagen con detalles en diferentes zonas.

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 – Validación de formulario

En la Figura 2 se muestra el esquema de un formulario (imagen formulario_vacio.png), con sus respectivos campos. La primera fila define el tipo de formulario (A, B o C), luego hay 4 campos para completar datos personales, luego 3 preguntas que deben responderse por SI o por NO, y por último un campo de comentarios libres.

FORMULARIO A		
Nombre y apellido		
Edad		
Mail		
Legajo		
	Si	No
Pregunta 1		
Pregunta 2		
Pregunta 3		
Comentarios		

Figura 2 – Esquema del formulario.

Se tiene una serie de formularios completos, en formato de imagen, y se pretende validar cada uno de ellos, corroborando que cada uno de sus campos cumpla con las siguientes restricciones:

1. Nombre y apellido: Debe contener al menos 2 palabras y no más de 25 caracteres en total.
2. Edad: Debe contener 2 o 3 caracteres.
3. Mail: Debe contener 1 palabra y no más de 25 caracteres.
4. Legajo: 8 caracteres formando 1 sola palabra.
5. Preguntas: se debe marcar con 1 caracter una de las dos celdas SI y NO. No pueden estar ambas vacías ni ambas completas.
6. Comentarios: No debe contener más de 25 caracteres.

Asuma que todos los campos ocupan un solo renglón (el de comentarios también), y que se utilizan solo las siguientes letras mayúsculas, números y símbolos:

A B C D E F G H I J K L M N O P Q R S T U V W X Y Z 0 1 2 3 4 5 6 7 8 9 @ . - /_

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.

FORMULARIO A		
Nombre y apellido	JUAN PEREZ	
Edad	45	
Mail	JUAN_PEREZ@GMAIL.COM	
Legajo	P-3205/1	
	Si	No
Pregunta 1	X	
Pregunta 2		X
Pregunta 3	X	
Comentarios	ESTE ES MI COMENTARIO.	

(a)

FORMULARIO A		
Nombre y apellido	JORGE	
Edad	4500	
Mail	JORGE @GMAIL.COM	
Legajo	X45ASLAB W45	
	Si	No
Pregunta 1		
Pregunta 2	X	x
Pregunta 3		xx
Comentarios	ESTE ES UN COMENTARIO MUY MUY LARGO.	

(b)

Figura 3 – Ejemplos de carga de formularios. (a) Todos los campos bien cargados. (b) Todos los campos mal cargados.

Desarrolle un algoritmo para validar los campos del formulario. Debe tomar como entrada la imagen del mismo y mostrar por pantalla el estado de cada uno de sus campos. Por ejemplo:

Nombre y apellido:	OK
Edad:	OK
Mail:	MAL
Legajo:	MAL
Pregunta 1:	OK
Pregunta 2:	MAL
Pregunta 3:	OK
Comentarios:	OK

Utilice el algoritmo desarrollado para evaluar las imágenes de formularios completos (archivos formulario_xx.png) e informe los resultados obtenidos.

AYUDAS:

1) Existen varias formas de detectar las celdas donde se cargan los datos, una de ellas es detectando las coordenadas de las líneas verticales y horizontales que dividen el formulario en filas y columnas. 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 columnas **img_cols = np.sum(img_th_ones,0)** y sumar el valor de los pixels que están en cada fila para detectar las filas **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 filas y otro para las columnas) y detectar así las posiciones de las mismas. Por ejemplo: **img_rows_th = img_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**.
2) Una vez obtenida las celdas, una posible forma de obtener los caracteres dentro de la misma, es obteniendo las componentes conectadas dentro de la misma: **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,:]**.