TRABAJO PRÁCTICO N° 1 - Año 2025 - 2° 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 (archivo *Imagen_con_objetos_ocultos.tiff*) 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.

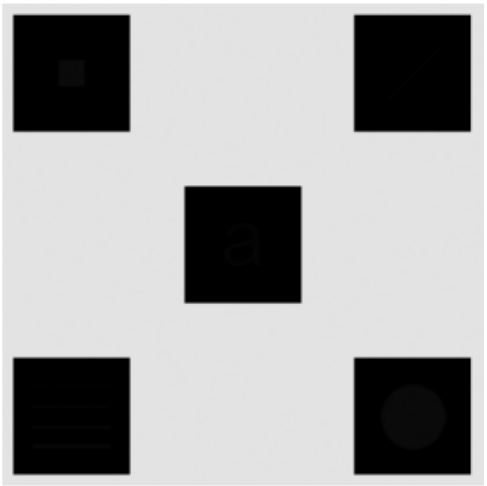

Figura 1: Imagen con detalles ocultos en diferentes zonas.

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 - Validación de formulario

En la Figura 2 se muestra el esquema de un ***formulario*** (archivo *formulario_vacio.png*) con sus respectivos campos (*Nombre y Apellido*, *Edad*, *Mail*, *Legajo*, *Pregunta 1*, *Pregunta 2*, *Pregunta 3* y *Comentarios*) y encabezado. En este último se define el tipo de formulario (*A*, *B* o *C*) y por su parte, cada campo se completa bajo ciertas restricciones que se detallarán a continuación.

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

Figura 2: Ejemplo del esquema de un formulario.

Se tiene una serie de formularios con información, en formato de imagen, y se pretende validarlos de forma **automática** por medio de un *script* en Python. Para ello, debe considerar las siguientes restricciones de cada campo:

a. **Nombre y apellido:** Debe contener un mínimo de dos palabras y no más de 25 caracteres en total.
b. **Edad:** Debe contener 2 o 3 caracteres consecutivos (no deben existir espacios entre ellos).
c. **Mail:** Debe contener una palabra y no más de 25 caracteres.
d. **Legajo:** Debe contener sólo 8 caracteres en total, formando una única palabra.
e. **Preguntas 1, 2 y 3:** En cada pregunta debe haber una única celda marcada (referenciadas a las columnas de *Si* y *No*), con un único caracter. Es decir, en cada pregunta, ambas celdas no pueden estar marcadas ni ambas celdas pueden quedar vacías.
f. **Comentarios:** Debe contener al menos una palabra y no más de 25 caracteres.

Asuma que todos los campos ocupan un sólo renglón (incluso el de comentarios), y que sólo se utiliza los siguientes caracteres:

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 @ . - /_

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

A. Se debe tomar únicamente como entrada la imagen de un formulario y mostrar por pantalla cuáles de los **campos** son **correctos (OK)** y cuáles son **incorrectos (MAL)**. Por ejemplo, en el terminal se debe visualizar:

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

En la Figura 3a se muestra un ejemplo en donde los campos están todos correctamente completos, mientras que en la Figura 3b se muestra otro ejemplo donde todos los campos están incorrectamente completos.

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.	

Figura 3a: Ejemplo de formulario en donde todos los campos se cargaron correctamente.

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.	

Figura 3b: Ejemplo de formulario en donde todos los campos se cargaron incorrectamente.

B. Se debe aplicar el algoritmo desarrollado, de **forma cíclica**, sobre el conjunto de cinco imágenes de formularios completos (archivos *formulario_<id>.png*) e informar resultados en función del tipo de formulario (*A*, *B* o *C*).

C. Se debe generar una **única imagen de salida** que informe aquellas personas que han completado correctamente el formulario y aquellas personas que lo han completado de forma incorrecta. Dicha imagen de salida debe contar con el *"crop"* del contenido del campo *Nombre y Apellido* de cada formulario del punto anterior, junto con algún indicador que diferencie a aquellos que correspondan a un formulario correcto de aquellos que sean incorrectos.

D. Se debe generar un archivo **CSV** para almacenar los resultados de cada validación. En el mismo, cada fila (registro) debe corresponder al conjunto de resultados de un formulario procesado y debe responder a las siguientes pautas:

a. Debe tener un identificador único (ID) en la primera columna cuyo valor se debe corresponder con el ID utilizado en el nombre del archivo (*formulario_<id>.png*).
b. Debe tener una serie de columnas correspondientes a los distintos campos del formulario (respetando el orden): *Nombre y Apellido, Edad, Mail, Legajo, Pregunta 1, Pregunta 2, Pregunta 3* y *Comentarios*.
c. Cada celda debe registrar el resultado de la validación, utilizando únicamente los valores posibles: *OK* o *MAL*

AYUDAS:

- Existen varias formas de detectar las celdas donde se completan los datos, 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 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, :]**.