

Procesamiento de Imágenes Digitales

Trabajo práctico N°1 - Informe.

López Ceratto, Julieta : L-3311/1
Dimenna, Valentín : D-43366/4
Onega, Miranda Pilar : O-1779

Facultad de Ciencias Exactas, Ingeniería y
Agrimensura
Tec. Universitaria en Inteligencia Artificial
2024

Índice

Introducción.....	2
Ejercicio 1: Ecualización Local del Histograma.....	3
Solución.....	3
Variando parámetros.....	5
Ejercicio 2: Corrección del Examen.....	8
A. Encabezado.....	8
Primer paso: obtener campos.....	8
Segundo paso: corregir campos.....	8
B. Corrección de preguntas.....	10
Primera parte: recortar preguntas.....	10
Segunda Parte: obtener y detectar letra.....	11
Tercera parte: corregir preguntas.....	15
Cuarta parte: armar imagen de salida.....	16
Ejecución del programa.....	19
Funciones utilizadas.....	20

Introducción

Este informe corresponde al trabajo práctico n°1 de la asignatura Procesamiento de Imágenes Digitales de la carrera Tecnicatura Universitaria en Inteligencia Artificial.

El trabajo consta de dos ejercicios: “Ecuilización Local de Histograma” y “Corrección de Múltiple Choice”.

En este informe se explica cómo el equipo encaró cada ejercicio, con imágenes paso a paso y comentarios u observaciones.

Ejercicio 1: Ecualización Local del Histograma.

Solución.

Para la resolución de este ejercicio, como primer paso se define la función que se solicita:

```
def ecualizacion_hist(img, M, N)
```

Donde:

- Img : imagen en escala de grises.
- M: alto de la ventana.
- N: ancho de la ventana.

Para ecualizar la imagen de forma local, se debe primeramente obtener la forma de la imagen original y definir bordes a agregar a una copia de la imagen original con el fin de que la ventana de ecualización no sobresalga de dicha imagen recibida.

```
#Obtener forma de la imagen
h, w = img.shape

# Definir bordes
top = bottom = M // 2
left = right = N // 2

# Agregar bordes
img_border = cv2.copyMakeBorder(img, top=top, bottom=bottom,
left=left, right=right, borderType=cv2.BORDER_REPLICATE)
```

Luego, se crea una imagen llamada 'img_result' que será la imagen donde se vuelque la ecualización obtenida.

```
# Crear imagen en para almacenar resultados
img_result = img.copy()
```

Luego, se itera por cada píxel de la imagen con borde, pero usando los rangos de la imagen original; de esta forma la ventana nunca se encontrará por fuera de la imagen con borde. En cada píxel iterado, se calcula el histograma de la ventana, se obtiene la distribución acumulada del mismo y se normaliza dicha distribución; luego se lo pasa a tipo uint8 para asegurarse de que no se obtengan valores de píxeles no deseados. Como siguiente paso, se obtiene el valor del píxel correspondiente en el histograma normalizado y se guarda ese valor del píxel en 'img_result'

```
# Iteración por cada píxel de la imagen original
for i in range(h):
    for j in range(w):
        # Definir los límites de la ventana
        max_i = i + M
        max_j = j + N

        # Extraer la ventana
        ventana = img_border[i:max_i-1, j:max_j-1]

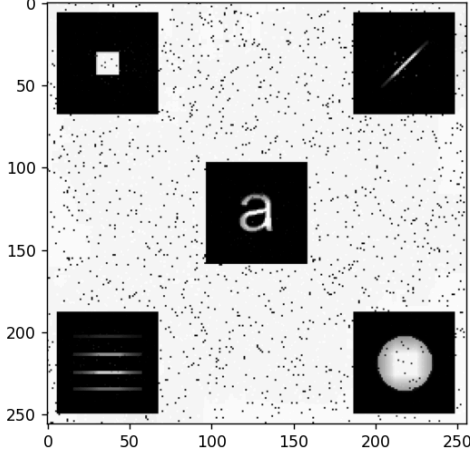
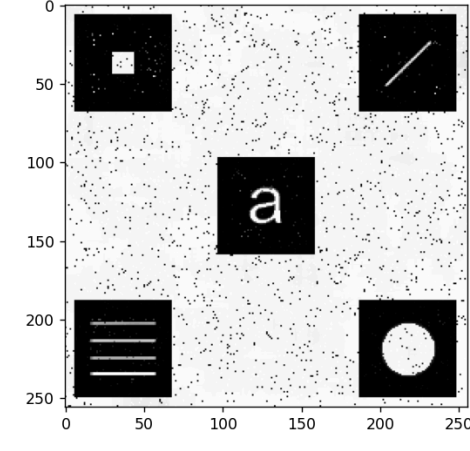
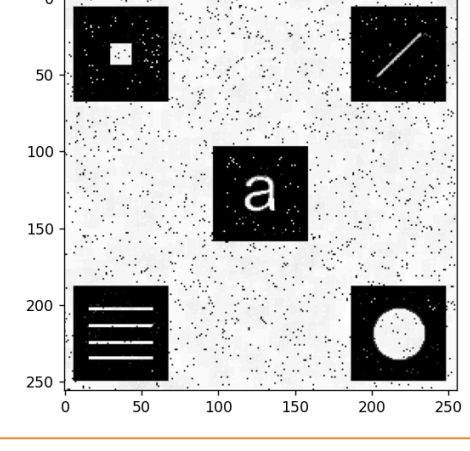
        # Calcular histograma
        hist = cv2.calcHist([ventana], [0], None, [256], [0,
256])

        # Calcular la distribución acumulada (CDF)
        cdf = hist.cumsum()
        cdf_norm = (cdf - cdf.min()) * 255 // (cdf.max() -
cdf.min())
        cdf_norm = cdf_norm.astype('uint8') # Asegurarse de que
sea tipo uint8

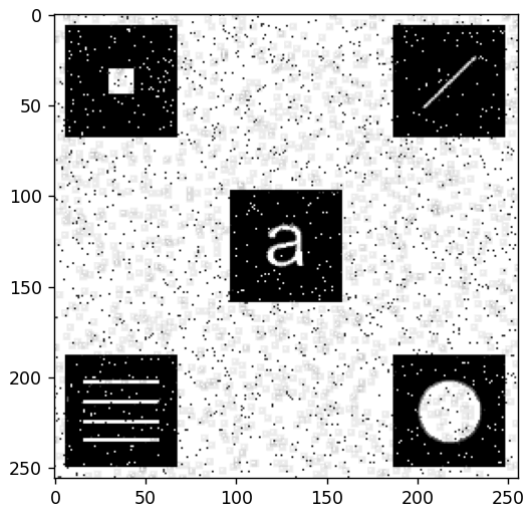
        # Obtener el valor del píxel para la cdf normalizada
        pix_norm = cdf_norm[img_border[i+top, j+right]]
        img_result[i, j] = pix_norm
    #img_result = cv2.medianBlur(img_result)
```

Variando parámetros.

Observamos cómo varía el resultado de la ecualización local según la variación en los tamaños de la ventana.

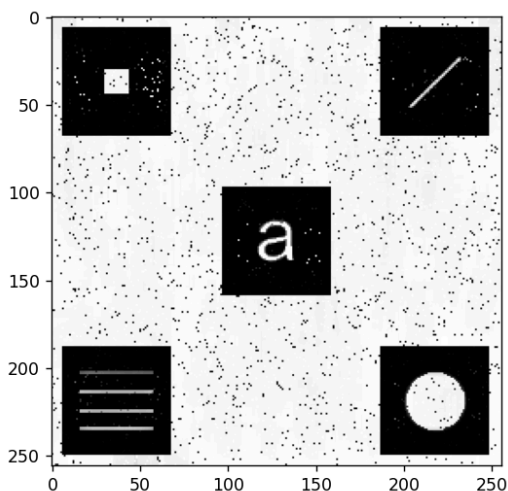
Resultado	Interpretación
<p>M y N = 50.</p> 	<p>Se obtiene poca definición de las figuras escondidas en los cuadrados y presenta un ruido propio de la granularidad del filtro; con una ventana tan grande, se calcula el histograma de una región extensa, lo que puede hacer que pequeñas variaciones en la intensidad de los píxeles se propaguen y aparezcan como bloques o patrones irregulares.</p>
<p>M y N = 25</p> 	<p>Se obtiene más definición de las figuras contenidas dentro de los cuadrados, aunque se pueden ver 3 cosas:</p> <ul style="list-style-type: none"> - Persiste el ruido del caso anterior; - La figura del cuadrado inferior izquierdo aún no está completamente descubierta. - Se presenta un ruido visual dentro de los cuadrados negros: En imágenes con áreas uniformes, la ecualización local puede amplificar pequeñas variaciones en la intensidad de los píxeles, generando lo que parece ser "ruido" o pixelación.
<p>M y N = 15</p> 	<p>Ahora la figura dentro del cuadrado inferior izquierdo se ve clara, pero a costa de incrementar ese ruido dentro de los cuadrados negros.</p>

M y N = 5.



El ruido dentro de los cuadrados negros ya se hace mucho más intenso, incluso afectando a la claridad de las figuras. Ahora aparecen píxeles grises en el fondo, esto se debe a que cuando se aplica una ventana pequeña en la ecualización local, lo que permite que los detalles pequeños o variaciones en las intensidades se mantengan y se resalten; sin embargo está capturando detalles por demás en la variación de los píxeles.

M = 35, N = 15.

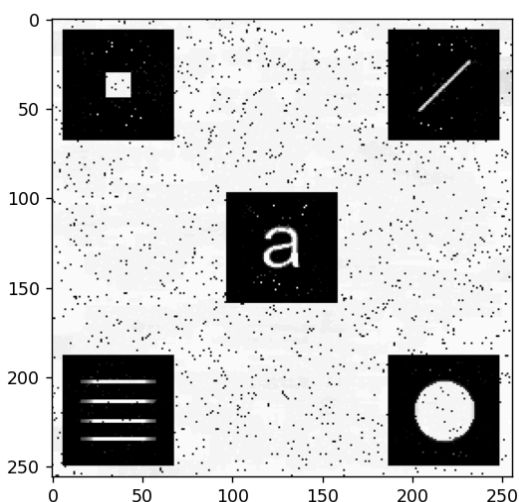


Se puede observar que a valores distintos de M y N se trasladan los efectos anteriores según dicho valor.

En este caso, como N, que corresponde al ancho de la ventana, es bajo, se presenta la detección de detalles pequeños a lo ancho de la ventana y no así con el M (alto).

Esto se ve también en la poca detección de las líneas horizontales desde arriba hacia abajo.

M = 15, N = 35.



Sucede lo inverso a lo anterior. Nótese cómo ahora la figura de las líneas presenta más definición de arriba hacia abajo pero no a lo ancho.

Observaciones.

Podemos observar que a **mayor tamaño de M y N** tiende a ser una ecualización global del histograma, no logrando la visualización correcta de las figuras dentro de los cuadrados.

Por otro lado, un **M y N muy pequeños** hace que se produzca ruido dentro de los cuadrados e incluso píxeles grises en el fondo debido a que se capturan variaciones muy pequeñas en la imagen.

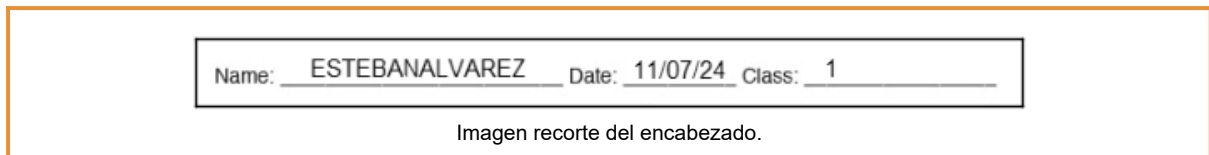
Finalmente, un **M y N desiguales** hace que las características vistas anteriormente se propaguen a lo alto y ancho según el valor que adopten M y N.

Ejercicio 2: Corrección del Examen.

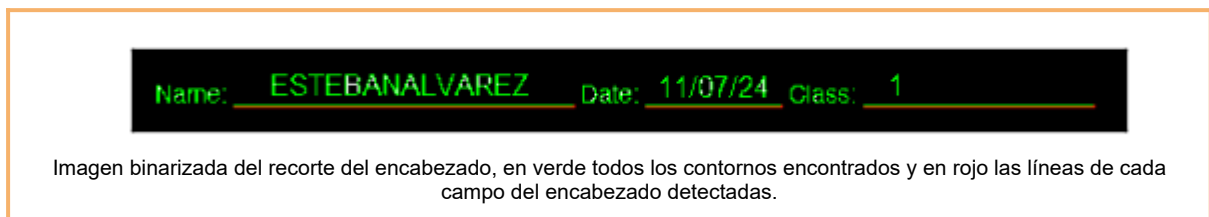
A. Encabezado.

Primer paso: obtener campos.

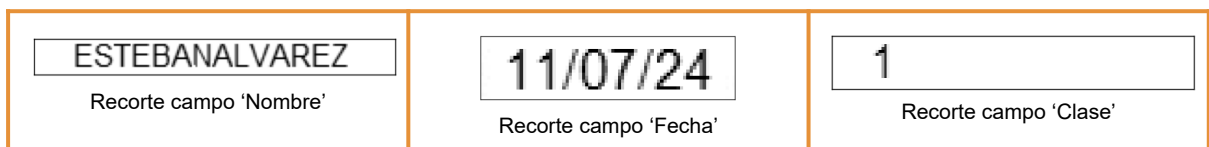
Para comenzar, se busca obtener recortes de los campos del examen. Primero hacemos un recorte manual de la parte superior del examen.



Luego, buscamos líneas horizontales -sobre una imagen binarizada del recorte- cuyo ancho mínimo sea 70 píxeles, obteniendo así las 3 líneas del campo del encabezado.



Finalmente, dividimos el encabezado según las coordenadas x de las líneas para el ancho y para las coordenadas y: $[y-20: y+h-2]$ ya que de esta manera abarcamos los datos del campo y además no sale la línea del encabezado en el recorte.



Segundo paso: corregir campos.

En este paso se utiliza la herramienta de componentes conectadas de forma 8 (es decir con píxeles conectados horizontal, vertical y diagonalmente). A esta herramienta se le pasa una imagen binarizada del campo, obteniendo así un componente por letra/carácter de la imagen. Luego, se arma una lista que almacene los bounding boxes de las componentes para obtener así las coordenadas de las mismas.

Si no se encuentran componentes conectados se devuelve el tipo de campo y la palabra 'Mal' (ej: 'nombre: Mal').

Según el tipo de campo, se hacen cosas distintas:

Nombre.

Para el nombre, primero se cuenta si la cantidad de letras no superan las 25, de superarlas, se devuelve 'Nombre: mal'; caso contrario, se pasa a chequear si el campo contiene 2 palabras de la siguiente forma:

```
for x, y, w, h in bounding_boxes[1:]:
    if x - (ux + uw) > 3:          # Si la distancia entre la
        letra actual y la anterior
        palabras += 1              # es mayor a 3, empezó una
        nueva palabra

    ux, uy, uw, uh = x, y, w, h
```

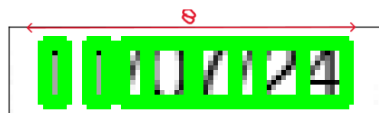
Es decir, chequea si la distancia entre el punto final x de una letra i con el primer punto x de una letra j es mayor a 3 píxeles; si es así, quiere decir que comienza una nueva palabra y se suma el conteo de palabras en 1.



Finalizando el conteo de palabras, en caso de que las palabras sean = 2; si es así devuelve 'Nombre: OK', caso contrario, devuelve 'Nombre: Mal'.

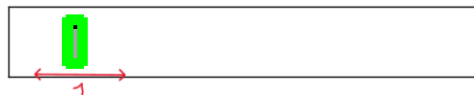
Fecha.

Para la fecha, se verifica que las componentes sean = 9 (si bien la fecha debe tener 8 caracteres, como el fondo es una componente más, se suma uno a la condición). De ser así, devuelve 'Fecha: OK', caso contrario, devuelve 'Fecha: Mal'.



Clase.

Para la clase, simplemente corrobora que la cantidad de componentes sea = 2 (pasa lo mismo con el fondo que en el caso de la fecha).



B. Corrección de preguntas.

Esta sección se dividió en cuatro partes: recortar las preguntas, obtener la letra - o espacio en blanco- contestada, corregir el examen propiamente dicho, devolver imagen aprobado/desaprobado por alumno.

Primera parte: recortar preguntas.

Para este punto se inició encontrando las líneas horizontales y verticales por las cuales están encerradas las preguntas. Para esto se obtuvo primero la imagen binarizada del examen para detectar más fácilmente las líneas.

Name:	JUAN PEREZ	Date:	11/07/24	Class:	1
1	The Earth's system that involves all our air is called the _____. C	6	The gaseous layers of the atmosphere are held to Earth's surface by _____. B		
	A geosphere B hydrosphere C atmosphere D biosphere		A their weight B gravity C the sun D none of the above		
2	The Earth's system that involves all our water is called the _____. B	7	78% of the Earth's atmosphere is made up of _____. A		
	A geosphere B hydrosphere C atmosphere D biosphere		A nitrogen B oxygen C carbon dioxide D water vapor		
3	The Earth's system that involves all our rock is called the _____. A	8	The layer of the atmosphere we live in is called the _____. B		
	A geosphere B hydrosphere C atmosphere D biosphere		A stratosphere. B troposphere. C mesosphere. D exosphere.		
4	The Earth's system that involves all living things is called _____. D	9	Most life in the ocean is found _____. D		
	A geosphere B hydrosphere C atmosphere D biosphere		A throughout all its waters. B deep down in the depths. C far from shore. D on the surface and closer to shore.		
5	97% of Earth's water is found in _____. B	10	A biome's location on Earth depends upon _____. D		
	A lakes B the ocean C our underground aquifers D the clouds		A climate B amount of rainfall C temperature D all of the above		

Luego Calculamos la cantidad de pixeles blancos en cada fila y columna para detectar las líneas horizontales y vertical y usamos la función "reducir_lineas_pixel" para eliminar líneas de ruido y eliminar líneas detectadas que están muy cerca entre sí.

```
img_th_ones = img_th // 255

img_cols = np.sum(img_th_ones, axis=0)
img_rows = np.sum(img_th_ones, axis=1)

vertical_threshold = 300
horizontal_threshold = 450

vertical_lines = np.where(img_cols > vertical_threshold)[0]
horizontal_lines = np.where(img_rows > horizontal_threshold)[0]
```

Líneas Detectadas:

Lineas detectadas

Name: <u>JUAN PEREZ</u> Date: <u>11/07/24</u> Class: <u>1</u>	
1	The Earth's system that involves all our air is called the <u>C</u> . A geosphere B hydrosphere C atmosphere D biosphere
2	The Earth's system that involves all our water is called the <u>B</u> . A geosphere B hydrosphere C atmosphere D biosphere
3	The Earth's system that involves all our rock is called the <u>A</u> . A geosphere B hydrosphere C atmosphere D biosphere
4	The Earth's system that involves all living things is called <u>D</u> . A geosphere B hydrosphere C atmosphere D biosphere
5	97% of Earth's water is found in <u>B</u> . A lakes B the ocean C our underground aquifers D the clouds
6	The gaseous layers of the atmosphere are held to Earth's surface by <u>B</u> . A their weight B gravity C the sun D none of the above
7	78% of the Earth's atmosphere is made up of <u>A</u> . A nitrogen B oxygen C carbon dioxide D water vapor
8	The layer of the atmosphere we live in is called the <u>B</u> . A stratosphere. B troposphere. C mesosphere. D exosphere.
9	Most life in the ocean is found <u>D</u> . A throughout all its waters. B deep down in the depths. C far from shore. D on the surface and closer to shore.
10	A biomes location on Earth depends upon: <u>D</u> . A climate B amount of rainfall C temperature D all of the above

Por último recortamos por columna delimitada por cada par de líneas verticales cada rectángulo de cada pregunta y lo guardamos en una lista de subimagenes.

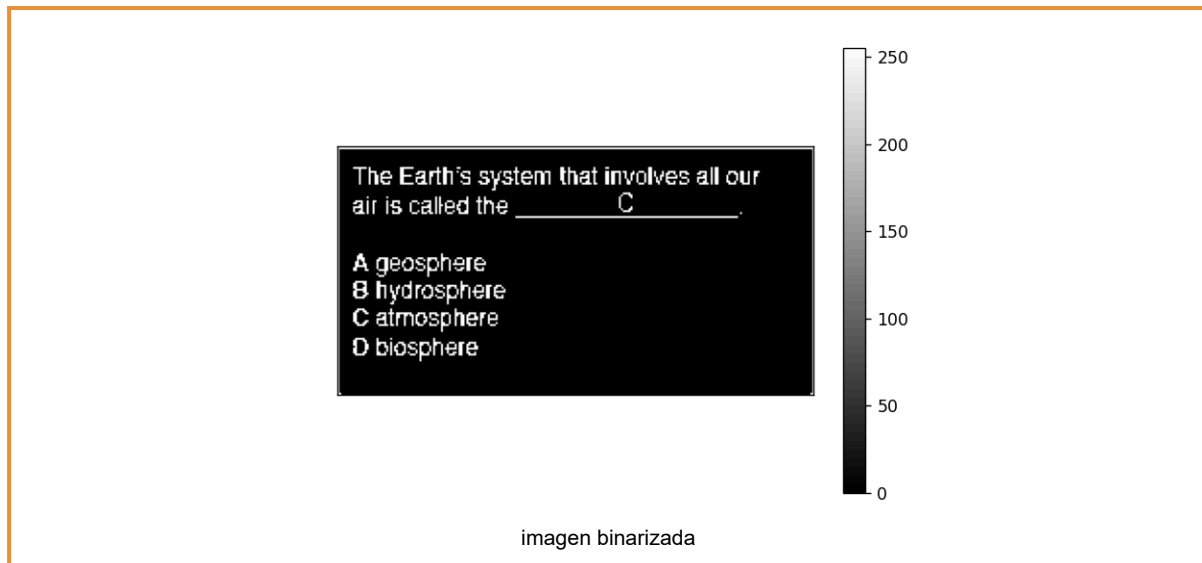
```
question_images = []
for j in range(0,4,2):
    for i in range(len(horizontal_lines_single_pixel)-1):
        top = horizontal_lines_single_pixel[i]
        bottom = horizontal_lines_single_pixel[i + 1]
        left = vertical_lines_single_pixel[j]
        right = vertical_lines_single_pixel[j + 1]
        # Cortamos cada pregunta
        sub_image = img[top:bottom, left:right]
        question_images.append(sub_image)
return question_images
```

The Earth's system that involves all our air is called the C.
A geosphere
B hydrosphere
C atmosphere
D biosphere

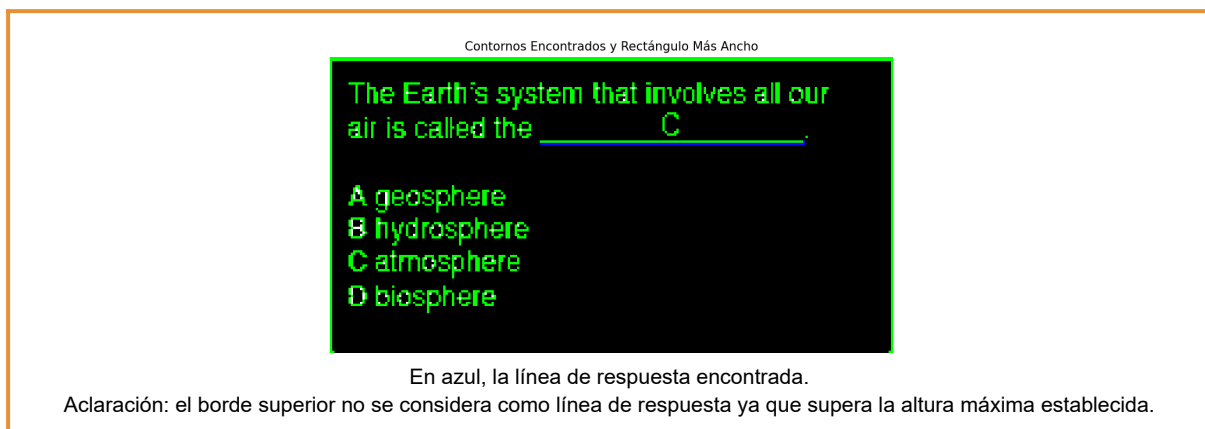
Segunda Parte: obtener y detectar letra.

Obtener la letra contestada.

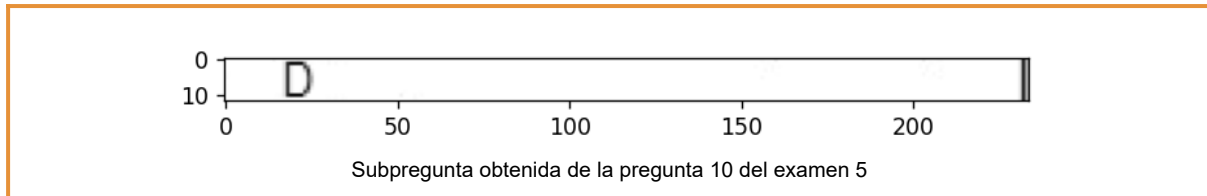
Para este punto, se encaró como primera tarea encontrar la línea para completar la pregunta. Para esto, se obtiene la imagen binarizada de la pregunta, esto va a permitir una mejor detección de contornos.



Luego, se busca el contorno con mayor ancho dentro de la pregunta entre todos los contornos encontrados; añadiendo además la condición de que el alto del contorno no sea mayor a 3. Es decir, entre todos los contornos, se busca aquel que corresponda a una línea.

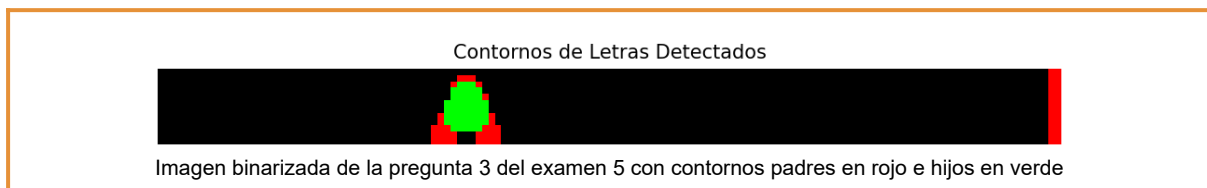


Finalmente se recorta la imagen de la pregunta en las coordenadas de la línea de respuesta haciendo que la coordenada de inicio y sea $y - 14$ ya que es el alto observado aproximado de los espacios de respuesta. De esta forma, obtenemos la letra -o espacio en blanco- contestada.



Detectar letra.

Para detectar la letra, al igual que el paso anterior, se obtiene la imagen binarizada: Como segundo paso, se obtienen los contornos aunque a diferencia del paso anterior, se utiliza la opción <RETR_TREE> de forma tal que se obtengan los contornos con sus jerarquías. Luego, se inicializa un diccionario 'contornos' que va a tener por como key el contorno padre y como value, sus contornos hijos - o lista vacía de no tenerlos- Como tercer paso, para cada contorno padre encontrado se buscan sus contornos hijos y se lo agrega al diccionario. De esta forma obtenemos un diccionario de contornos padre-hijos y también todos los contornos encontrados. Nótese que se obtiene un contorno padre por letra.

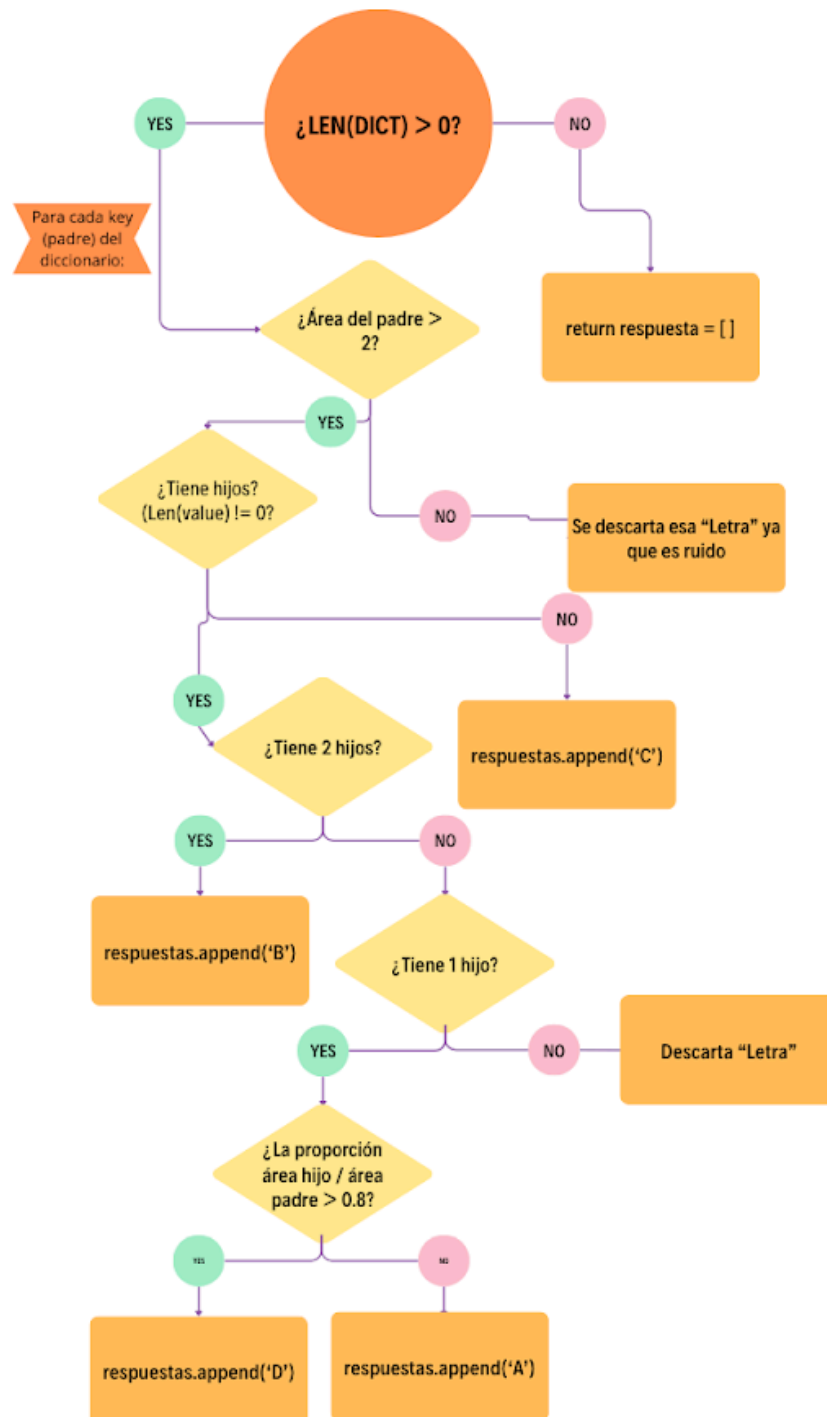


Clasificar letra.

Como tercer paso, para cada contorno padre encontrado, se analiza si tiene hijos y, de tenerlos, la relación con los mismos.

Antes de esto, se filtran los contornos padres que tengan un área menor a 2 para descartar ruidos.

Ahora bien, podemos realizar la siguiente clasificación:



De esta manera, obtenemos una lista “respuestas” con las letras contestadas en una pregunta.

Tercera parte: corregir preguntas.

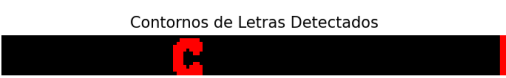
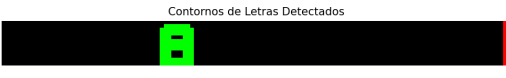
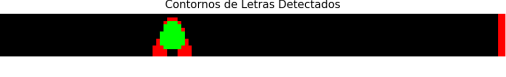

Corrección de pregunta.



Para este último paso de la sección del ejercicio, antes de comparar las letras encontradas con las respuestas correctas, se chequean dos condiciones: si la lista de respuestas tiene más de una letra, se devuelve 'Mal' automáticamente, mismo caso si la lista de respuestas no tiene ninguna letra.

Ahora bien, si la lista tiene una letra, se la compara con la lista de respuestas correctas por índice de pregunta y se imprime 'OK' si es igual a esta última y 'Mal' de no serlo.

Vemos un caso práctico.

Veamos el caso del examen 2 y 5:



Pregunta 1 (examen 4)	Detalle paso	Pregunta 2 (examen 4)
	Imagen binarizada	
padres: 2, hijos : 0	Detracción padre-hijo	padres: 2, hijos : 2
Se descarta el borde de la derecha por el filtro del área y se obtiene...	Filtro	Se descarta el borde de la derecha por el filtro del área y se obtiene...
respuestas = ['C']	Lista respuestas	respuestas = ['B']
Pregunta 1: OK	Respuesta	Pregunta 2: OK
Pregunta 3 (examen 4)		Pregunta 4 (examen 4)
	Imagen binarizada	
padres: 2, hijos : 1	Detracción padre-hijo	padres: 3, hijos : 1
Se descarta el borde de la derecha por el filtro del área, se ve que el área del hijo (verde) es menos del 0.8 del área del padre y se obtiene...	Filtro	Se descarta el borde y el punto de la derecha por el filtro del área, se ve que el área del hijo (verde) es más que el 0.8 del área del padre y se obtiene...
respuestas = ['A']	Lista respuestas	respuestas = ['D']
Pregunta 3: MAL	Respuesta	Pregunta 4: MAL

Pregunta 2 (examen 2)		Pregunta 5 (examen 2)
	Imagen binarizada	
padres: 3, hijos : 2	Detracción padre-hijo	padres: 1, hijos : 0
Se ve que la primera figura tiene 2 hijos por lo tanto es letra B, la segunda tiene 1, por lo tanto es letra C. Se obtiene...	Filtro	Se descarta el borde de la derecha por tamaño de área y se obtiene...
respuestas = ['B', 'C']	Lista respuestas	respuestas = []
Pregunta 2: MAL	Respuesta	Pregunta 5: MAL

Y así sucesivamente para cada pregunta del examen.

Cuarta parte: armar imagen de salida.

Para armar la imagen de salida, se hace uso de una función que genera una imagen representando el resultado de la imagen. Esta función recibe la nota del alumno y la imagen del encabezado 'nombre' de su examen. Si la nota es < 6 genera una imagen con un círculo rojo, caso contrario, genera un círculo verde. Esta imagen de resultado tiene la misma altura que la imagen del encabezado nombre del alumno, además tiene un ancho fijo de 50 píxeles.

 <p>Imagen de resultado para un examen no aprobado.</p>	 <p>Imagen de resultado para un examen aprobado.</p>
--	---

Luego se usa la función `resultados_exámenes` que recibe como entrada una lista con las ubicaciones de las imágenes de los exámenes.

Esta función ejecuta todas las funciones vistas en los apartados anteriores para cada examen y además arma la imagen de salida.

A medida que ejecuta las funciones anteriores de los exámenes va almacenando la altura necesaria de la imagen de salida obteniendo la altura de los encabezados 'nombre' de cada examen y agregándole 2 píxeles mas por examen.

```
h_img_salida += h+2
```

Donde h es la altura de un encabezado de un examen.

Los 2 píxeles extra se utilizarán luego para trazar una línea negra horizontal divisoria entre nombre y nombre.

Otro dato que se almacena es aquel encabezado nombre con el mayor ancho, esto servirá para dos cuestiones:

- Por un lado para obtener la posición x donde trazar la línea vertical negra divisoria entre los nombres de los exámenes y las fotos de los resultados.
- Por otro lado para obtener la posición x+3 donde insertar la imagen resultado.

A medida que se corrige cada examen se va almacenando en un diccionario con clave (i = numero de examen), value = (nombre_alumno, nota).

Una vez corregidos todos los exámenes se crea una imagen a color con las dimensiones:

```
img_salida = np.ones((h_img_salida, w_max_nombre + 53, 3),
dtype=np.uint8) * 255
```

Es decir, una imagen que tenga la altura almacenada en las iteraciones y el ancho del mayor ancho entre los encabezados de los nombres + 53 píxeles (50 del ancho de la imagen resultado y 3 del espacio dejado para trazar la línea vertical y el espacio entre la línea vertical y el inicio de la imagen resultado).

Luego se traza la línea vertical negra divisoria con las coordenadas obtenidas anteriormente:

```
# Dibujar la línea vertical en la posición deseada
img_salida[0:h_img_salida, w_max_nombre + 2] = 0 # Línea vertical
en negro
```

Posteriormente se pasa a iterar por el diccionario y se va completando la imagen de salida de la siguiente forma:

```
y_inicial = 0
l = 0
# Colocar nombres y resultados
for nombre_alumno, resultado in dict_nombre_resultado.values():
    h_alumno, w_alumno, _ = nombre_alumno.shape
```

y_inicial es la posición y de inserción del encabezado del nombre del alumno, naturalmente empezará por la posición 0.

En cada iteración se obtiene primero la altura y anchura del encabezado del alumno.

A continuación, se inserta en la posición y_inicial (que al principio es 0 pero luego no): altura del encabezado del alumno, 0 (todos los encabezados se insertan en x=0): ancho del encabezado: el encabezado nombre.

```
# Colocar la imagen del nombre
img_salida[y_inicial:y_inicial + h_alumno, 0:w_alumno] =
nombre_alumno
```

En el siguiente paso, se inserta la imagen resultado de la siguiente forma:

```
# Colocar la imagen del resultado
img_salida[y_inicial:y_inicial + h_alumno, w_max_nombre +
3:w_max_nombre + 3 + resultado.shape[1]] = resultado
```

Donde las coordenadas y siguen la misma lógica que para el encabezado nombre (recordar que tanto el encabezado como la imagen resultado tienen la misma altura), y la coordenada x va a corresponder a $w_max_nombre + 3$ (q píxel luego de la línea vertical) hasta la posición x: $w_max_nombre + 3 + resultado.shape[1]$ (o igual, $w_max_nombre+53$).

Como tercer elemento a colocar en la iteración, se presenta la línea horizontal negra divisoria de nombres. Aunque para esto nos aseguramos que no se trate del último alumno de la lista:

```
if l != len(dict_nombre_resultado)-1:
    # Dibujar la línea horizontal
    img_salida[y_inicial + h_alumno:y_inicial + h_alumno + 1,
:] = 0
```

La posición y se corresponde al final del nombre del alumno y el ancho abarca la totalidad de la imagen.

Como último paso en la iteración, se actualizan los valores:

```
y_inicial += h_alumno + 2 # Ajustar la posición para el
siguiente alumno
l += 1
```

donde $y_inicial$ va a pasar a ser la altura del encabezado del alumno + 2 píxeles y será la próxima posición a insertar el siguiente encabezado nombre, y l es el número de alumno.

Finalmente, se agregan bordes de 1 píxel negro a la imagen de salida como cuestión estética:

```
img_salida = cv2.copyMakeBorder(img_salida,1,1,1,1,
cv2.BORDER_CONSTANT)
```

Obtenemos así una imagen del siguiente estilo que deja ver que nadie aprobó el examen:

ESTEBANALVAREZ	✖
MARIA	✖
MARIA LOPEZ	✔
LUCAS FERNANDEZ	✖
JUAN PEREZ	✔

Ejecución del programa.

Para utilizar el programa corrector de exámenes (tp_1_problema_2.py), una vez ejecutado el mismo le aparecerá por consola la siguiente solicitud:

```
Ingrese la lista de respuestas correctas para corregir este examen. Deben ser ingresadas con una "," de separación (Ej: D,C):
```

Debe escribir, como dice el enunciado, todas las respuestas correctas del/los examen/es a corregir. Por ejemplo:

```
Ingrese la lista de respuestas correctas para corregir este examen. Deben ser ingresadas con una "," de separación (Ej: D,C): 'C', 'B', 'A', 'D', 'B', 'B', 'A', 'B', 'D', 'D'
```

No debe haber espacios, la única separación entre letra y letra debe ser la ",", y debe ser la respuesta en mayúscula.

Luego, le aparecerá esta solicitud:

```
Ingrese la ubicación del archivo examen o los arhivos examen, conteste "q" cuando haya terminado:
```

Por cada examen que tenga, debe proveer una ruta al mismo en cada solicitud (es una solicitud por examen). Por ejemplo, en caso de que tenga 3 exámenes será algo como esto en consola:

```
Ingrese la ubicación del archivo examen o los arhivos examen, conteste "q" cuando haya terminado: src\examen_1.png
Ingrese la ubicación del archivo examen o los arhivos examen, conteste "q" cuando haya terminado: src\examen_2.png
Ingrese la ubicación del archivo examen o los arhivos examen, conteste "q" cuando haya terminado: src\examen_3.png
```

Para que comience a corregir los exámenes, debe contestar 'q' en la solicitud:

```
Ingrese la ubicación del archivo examen o los arhivos examen, conteste "q" cuando haya terminado: q
```

Una vez contestada q, el programa comenzará a correr solo. Mostrándole:

- Validación de cada encabezado.
- Corrección de pregunta.
- Nota del examen.
- Imagen con el resultado de los exámenes (aprobado o no aprobado)

Funciones utilizadas.

EJERCICIO 1: ECUALIZACIÓN DEL HISTOGRAMA.		
FUNCIÓN.	EXPLICACIÓN.	USO
<code>ecualizacion_hist(img, M, N)</code>	Esta función devuelve una imagen ecualizada localmente según los valores de largo y ancho de una ventana MxN.	<ul style="list-style-type: none"> - <code>img</code>: imagen en escala de grises. - <code>M</code>: largo de la ventana. - <code>N</code>: ancho de la ventana.
EJERCICIO 2: CORRECCIÓN DE EXAMEN.		
FUNCIÓN.	EXPLICACIÓN.	USO
<code>binarize(img: np.array, tr: int = 150, maxv: int = 255) -> np.array</code>	Esta función recibe una imagen y la binariza según valores de <code>tr</code> y <code>maxv</code> recibidos.	<ul style="list-style-type: none"> - <code>tr</code> : thresh. - <code>maxv</code>: máximo valor de imagen de salida.
<code>user()</code>	Esta función interactúa con el usuario. Solicita la lista con las respuestas correctas del examen y el path de la imagen de cada examen.	
Parte 1: encabezado.		
FUNCIÓN.	EXPLICACIÓN.	USO
<code>obtener_campos(examen: np.array) -> tuple[np.array, np.array, np.array]</code>	Devuelve 3 imágenes de los campos a analizar.	<ul style="list-style-type: none"> - <code>examen</code> : imagen en escala de grises del examen; - <code>nombre</code>: imagen con el nombre; - <code>fecha</code>: imagen con la fecha; - <code>clase</code> : imagen con la clase.
<code>obtener_datos_de_campos(campo: np.array, tipo: str) -> str</code>	Corrige los campos según el tipo de campo.	<ul style="list-style-type: none"> - <code>campo</code>: imagen del campo. - <code>tipo</code>: tipo de campo del que se trata (nombre, fecha, clase).
Parte 2: recorte de preguntas.		
FUNCIÓN.	EXPLICACIÓN.	USO
<code>recortar_preguntas(img: np.array) -> list</code>	Recorta pregunta dada una imagen del examen, devuelve una lista de imágenes de preguntas.	<ul style="list-style-type: none"> - <code>img</code>: imagen del examen;

Parte 3: Detección de Respuestas y corrección de preguntas.		
FUNCIÓN.	EXPLICACIÓN.	USO
<code>lineas_horizontales(img_bin: np.array) -> tuple</code>	Encuentra líneas horizontales con el ancho máximo en la imagen y con un alto máximo de 3.	- <code>img_bin</code> : imagen binarizada;
<code>recortar_pregunta(pregunta: np.array, linea_preg: tuple) -> np.array</code>	Recorta la parte de respuesta del examen.	- <code>pregunta</code> : imagen escala de grises de pregunta del examen; - <code>linea_preg</code> : línea horizontal de la pregunta del examen;
<code>detectar_letra(sub_preg: np.array) -> tuple[dict, tuple]</code>	Detecta el/los caracter/es dentro del área de respuesta, dividiéndolos por padres con sus hijos. Devuelve un diccionario padre-hijo y los contornos encontrados.	- <code>sub_pregunta</code> : área de respuesta de la pregunta;
<code>clasificar_letra(dict_contorno: dict, contornos: tuple) -> list</code>	Clasifica qué letra es la de la respuesta, y devuelve una lista con las letras encontradas dentro de una respuesta.	- <code>dict_contornos</code> : diccionario clave contorno padre, valor contornos hijos; - <code>contornos</code> : total de contornos encontrados;
<code>corregir_pregunta(respuesta: list, i: int, respuestas_correctas: list) -> str</code>	Corrige la pregunta según la letra contestada. Si está bien contestada devuelve 'Ok', caso contrario (no contestó bien, no contestó o contestó más de una letra), devuelve 'Mal'.	- <code>respuesta</code> : letras encontradas en la pregunta; - <code>i</code> : número de pregunta; - <code>respuestas_correctas</code> : lista con las respuestas correctas del examen contra la que se comparará las respuestas dadas;
<code>corregir_examen(preguntas: list, respuestas_correctas: list) -> None</code>	Corrige todas las preguntas de un examen.	- <code>examen</code> : imagen escala de grises del examen completo; - <code>respuestas_correctas</code> : lista con las respuestas correctas del examen contra la que se comparará las respuestas dadas en el examen.
<code>img_resultado(nombre_alumno: np.array, nota: int) -> np.array</code>	Esta función arma una imagen de círculo rojo o verde con el mismo alto que la imagen del nombre del alumno según la nota recibida.	- <code>nombre_alumno</code> : imagen nombre del alumno. - <code>nota</code> : nota obtenida en el examen.
<code>resultados_exámenes(list_path)</code>	Corrige todos los exámenes.	- <code>list_path</code> : una lista con las ubicaciones de los archivos de los exámenes.