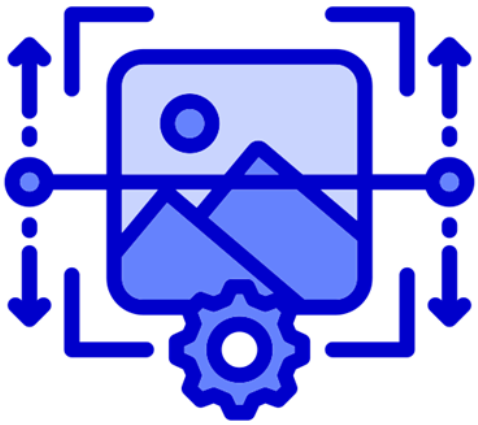
Trabajo Práctico Procesamiento de Imágenes 1

Tecnicatura Universitaria en Inteligencia Artificial.



Docentes:

- Gonzalo Sad
- Julián Alvarez
- Juan Manuel Calle

Integrantes (Grupo 11):

- Marcela Flaibani
- Facundo López Crespo
- Daniela Dito

Índice

Enunciado

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 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 en python, 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 la figura 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.

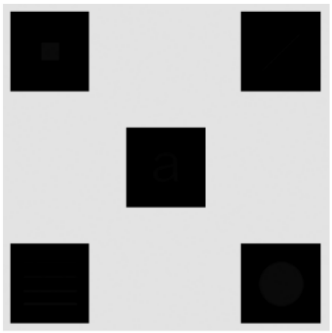


Figura 1: Imagen con detalles en diferentes zonas.

Problema 2 - Corrección de múltiple choice

En la figura 2 se muestra el esquema de un examen múltiple choice de 10 preguntas con cuatro opciones para cada una de ellas (A, B, C, D). La plantilla también tiene un encabezado con tres campos para completar datos personales (Name, Date y Class).

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, se asume que las respuestas correctas son las siguientes:

1. C 2. B 3. A 4. D 5. B 6. B 7. A 8. B 9. D 10. D

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 (no usar como dato las coordenadas de las preguntas) 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 10: 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. Date: deben ser 8 caracteres formando una sola palabra.

iii. Class: un único caracter.

Por ejemplo:

Name: OK
Date: MAL
Class: MAL

Asuma que todos los campos ocupan un solo renglón y que se utilizan caracteres alfanuméricos y barra inclinada " / ".

c. Utilice el algoritmo desarrollado para evaluar las imágenes de exámenes resueltos (archivos examen_<id>.png) e informe cada resultado obtenido.

d. Generar una imagen de salida informando los alumnos que han aprobado el examen (con al menos 6 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.

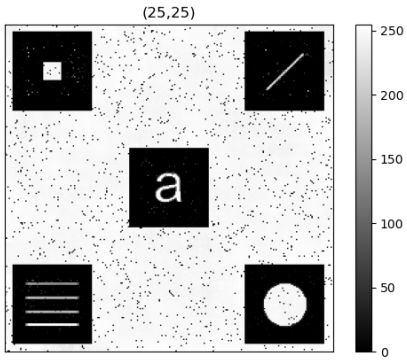
Resolución

Ejercicio 1

Imagen original: Niveles de intensidad de la imagen: [0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 226, 227, 228]
Cantidad de valores de intensidad únicos : 15
La imagen tiene niveles de grises concentrados en rangos estrechos (0, 11) y (226, 228).
Para maximizar el contraste de la imagen, el histograma debería estar más expandido, abarcando el máximo rango posible de valores.
Con la ecualización del histograma se busca que todos los niveles de grises aparezcan con frecuencias similares y que ocupen todo el rango de 0 a 255.

- Realizamos **umbrales y manipulación de intensidades (segmentación, análisis de regiones):**

  - Prueba 1: reemplazar todos los valores de píxeles **mayores que 0 por 255 (blanco),** es decir, cualquier píxel que no sea completamente negro se convierte en blanco.



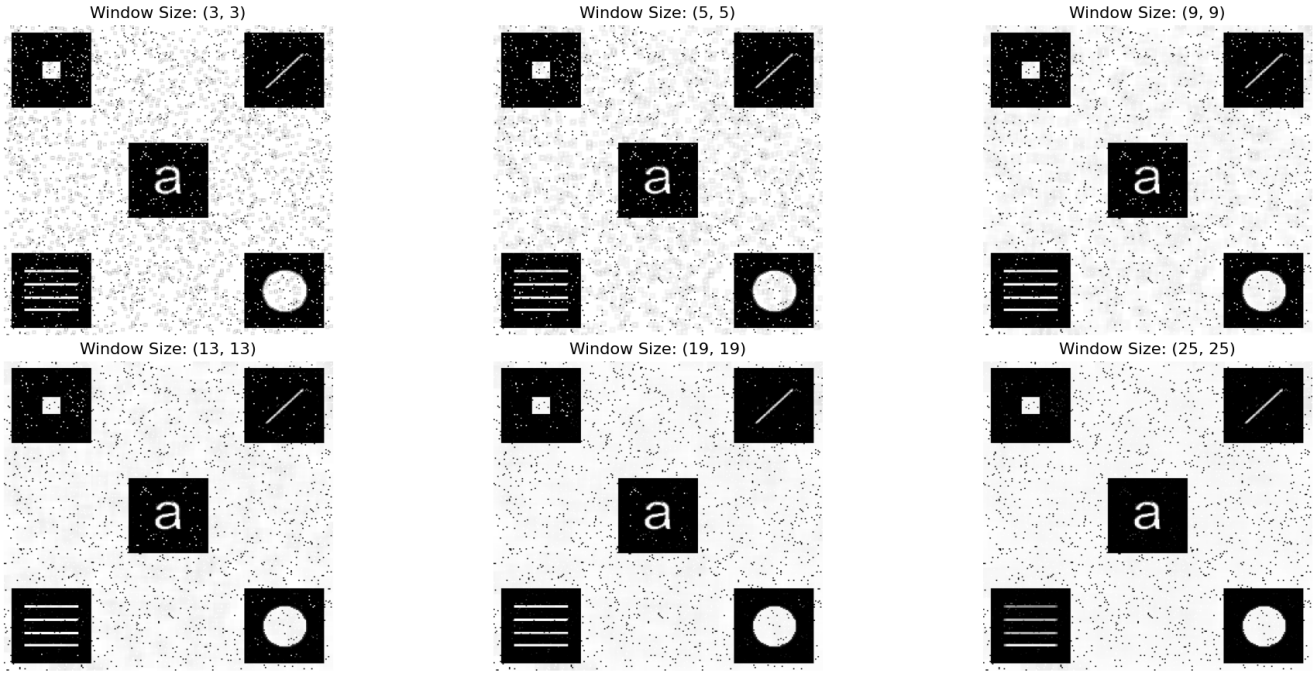
- **Aplicamos la ecualización local del histograma** en la imagen con el objetivo de mejorar el contraste, considerando pequeñas regiones o ventanas alrededor de cada píxel en lugar de la imagen completa, obteniendo una imagen con un mejor contraste, especialmente en áreas donde el contraste era bajo. (La imagen ecualizada se guarda como un archivo .TIFF.)

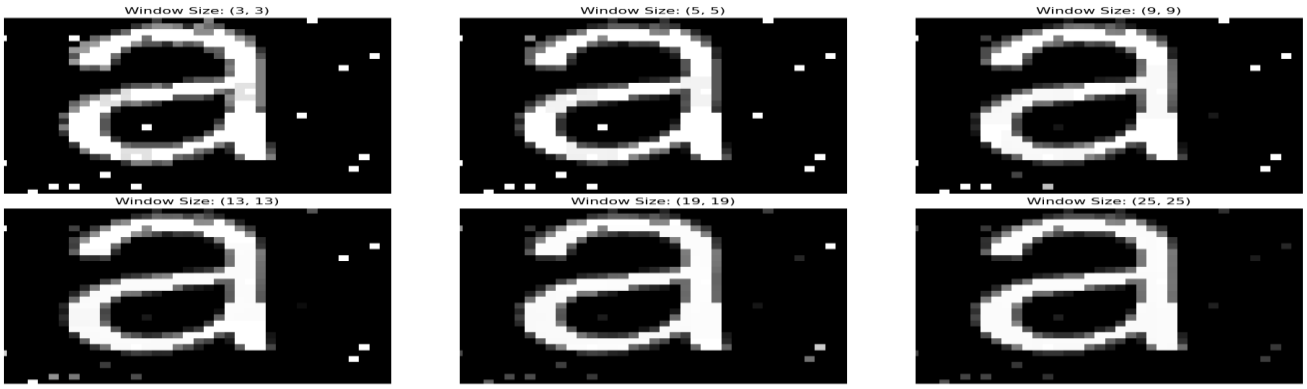
- **Bordes replicados:** Se agrega un borde (cv2.copyMakeBorder()) a la imagen para evitar problemas al aplicar la ventana en los bordes de la imagen. Esto asegura que la ventana siempre tenga el mismo tamaño, incluso cuando esté centrada en los píxeles cercanos a los bordes. Se elige el tipo de borde cv2.BORDER_REPLICATE que replica los píxeles de los bordes de la imagen original para rellenar el área agregada alrededor de la imagen.

- **Análisis del tamaños de ventana:** Se prueban distintos tamaños de ventana para recorrer cada pixel de la imagen.

  - Tamaños de ventana más pequeños **(por ejemplo, 3x3)** pueden resaltar detalles muy locales, pero también pueden generar ruido.

- Tamaños de ventana más grandes (por ejemplo, 25x25) suavizan el resultado, pero pueden perder algunos detalles finos.

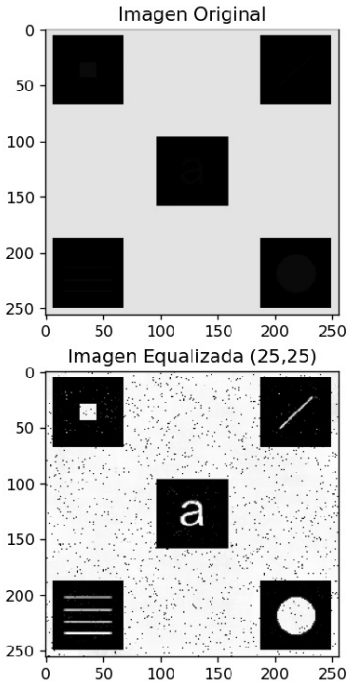


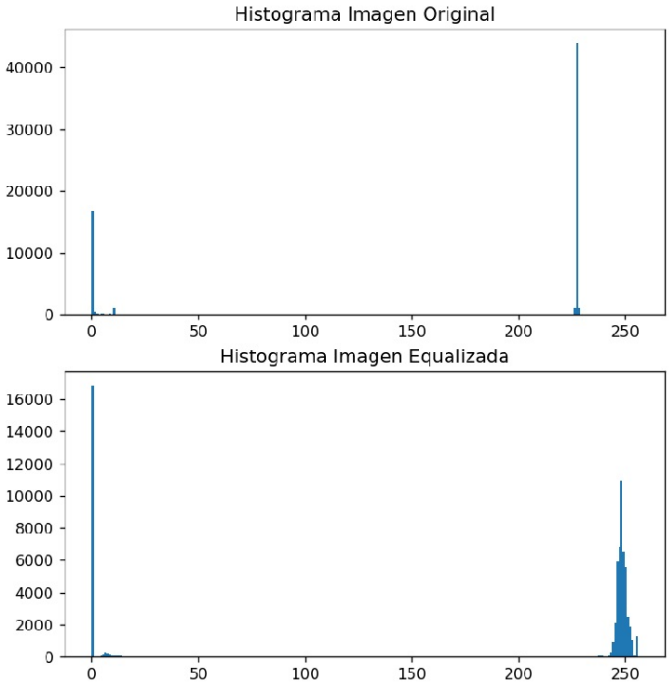


- **Comparación entre la ecualización Local y Global**

Característica	`cv2.equalizeHist(image)`	`local_histogram_equalization(image, window_size)`
Enfoque	Global	Local (ventana alrededor de cada píxel)
Control del tamaño de la ventana	No	Sí, con `window_size`
Contraste	Mejora globalmente	Mejora contrastes locales
Eficiencia	Muy rápida	Más lenta por el cálculo de ventanas locales
Detalles locales	No ajusta bien zonas específicas	Excelente para zonas con diferentes niveles de brillo
Posibles artefactos	Puede generar sobreexposición o ruido	Puede generar ruido si la ventana es muy pequeña

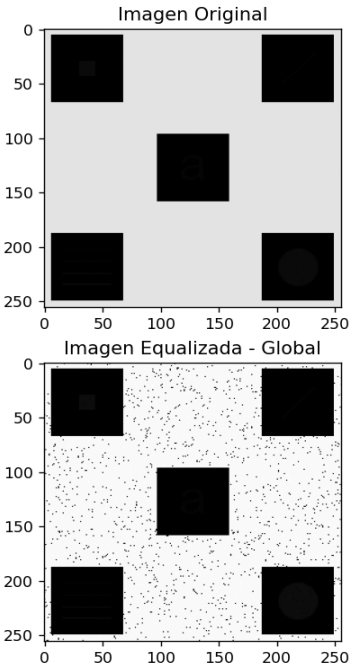
Ecualización Local

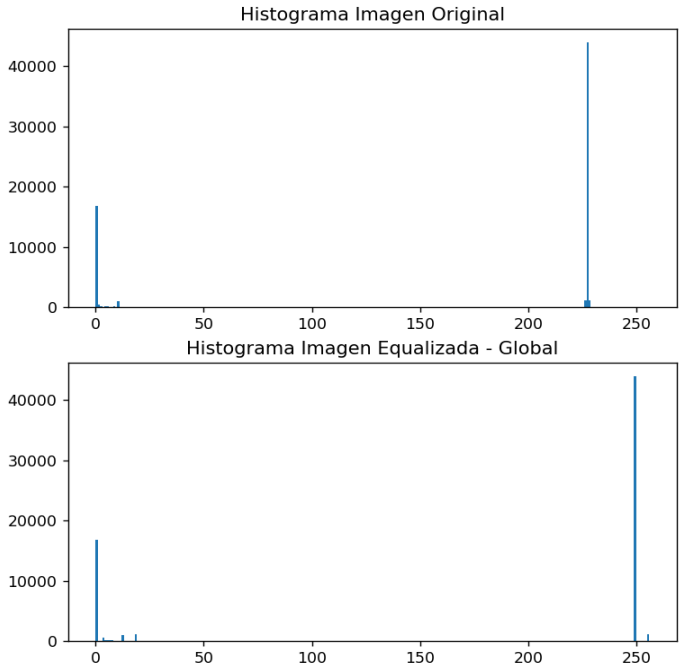




-

Ecualización Global



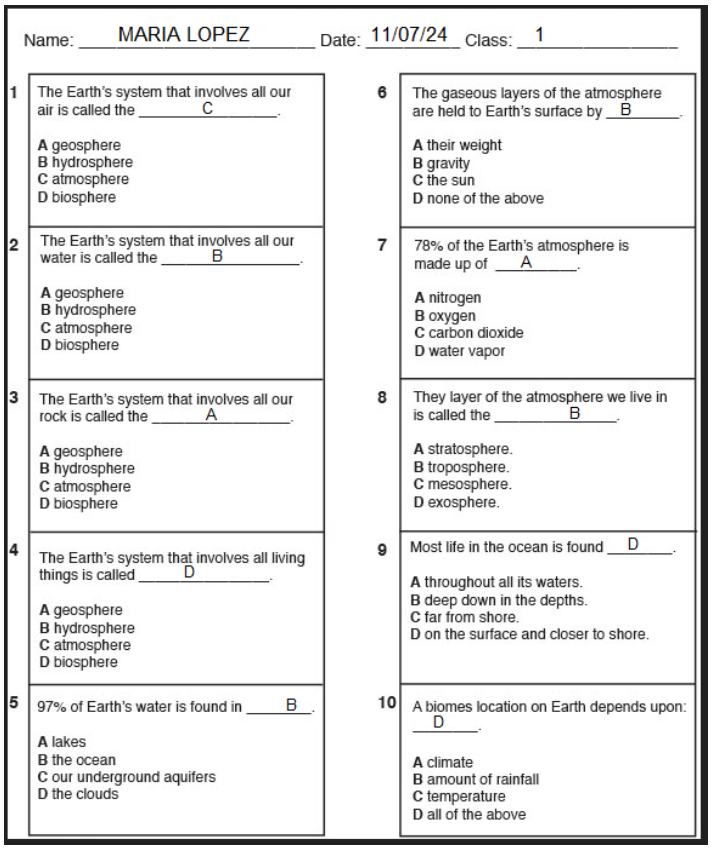


-

Resolución

Ejercicio 2

Se pretende corregir de forma automática, exámenes resueltos en formato de imagen, por medio de un script en python. Las respuestas correctas son: 1. C 2.B 3.A 4.D 5.B 6.B 7.A 8.B 9.D 10.D



Name: MARIA LOPEZ Date: 11/07/24 Class: 1

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

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

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

4 The Earth's system that involves all living things is called D.
A geosphere
B hydrosphere
C atmosphere
D biosphere

5 97% of Earth's water is found in B.
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 B.
A their weight
B gravity
C the sun
D none of the above

7 78% of the Earth's atmosphere is made up of A.
A nitrogen
B oxygen
C carbon dioxide
D water vapor

8 They layer of the atmosphere we live in is called the B.
A stratosphere.
B troposphere.
C mesosphere.
D exosphere.

9 Most life in the ocean is found D.
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: D.
A climate
B amount of rainfall
C temperature
D all of the above

El programa corrige todos los exámenes en una carpeta determinada, muestra los resultados por pantalla y los guarda en un archivo (.csv y/o .txt).

- Identificación de las áreas de la imagen correspondientes a cada pregunta

Función

```
def create_questions(image: np.ndarray) -> tuple:
```

1. Conversión a escala de grises

Convierte la imagen (en formato BGR, es decir, azul, verde, rojo) de color a escala de grises. En una imagen en escala de grises, cada píxel contiene un solo valor de intensidad de luz en lugar de valores separados de color para B, G y R. Esto facilita el procesamiento con imágenes de un solo canal.

2. Detección de bordes

Aplica el algoritmo de detección de bordes de Canny (`cv2.Canny()`) sobre la imagen en escala de grises. Este algoritmo busca cambios bruscos en la intensidad de los píxeles para identificar los bordes en la imagen. El resultado es una imagen binaria para aplicar la transformada de Hough.

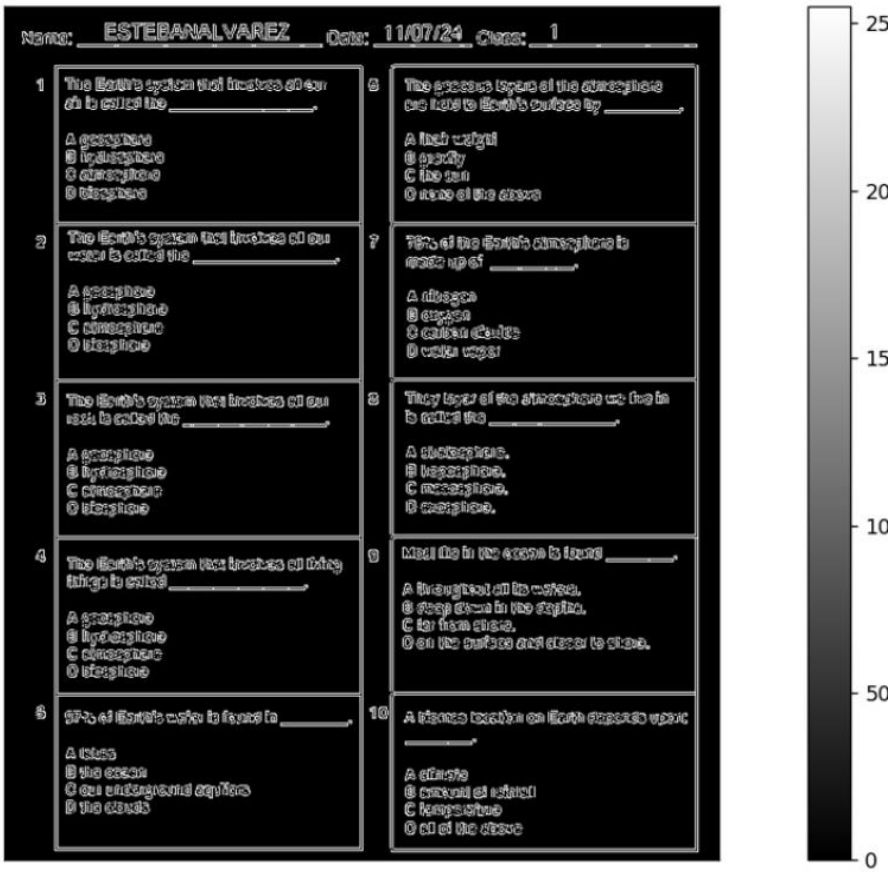
```
edges = cv2.Canny(image_gray, 100, 170, apertureSize=3)
```

- Parámetros:

`100`: Umbral inferior para el gradiente de los bordes. Los píxeles con gradientes por debajo de este valor se descartan.

`170`: Umbral superior para el gradiente de los bordes. Los píxeles con gradientes por encima de este valor son considerados como bordes fuertes.

`apertureSize=3`: Tamaño del filtro de Sobel utilizado internamente para calcular los gradientes. Un tamaño de 3 indica un filtro de 3x3.

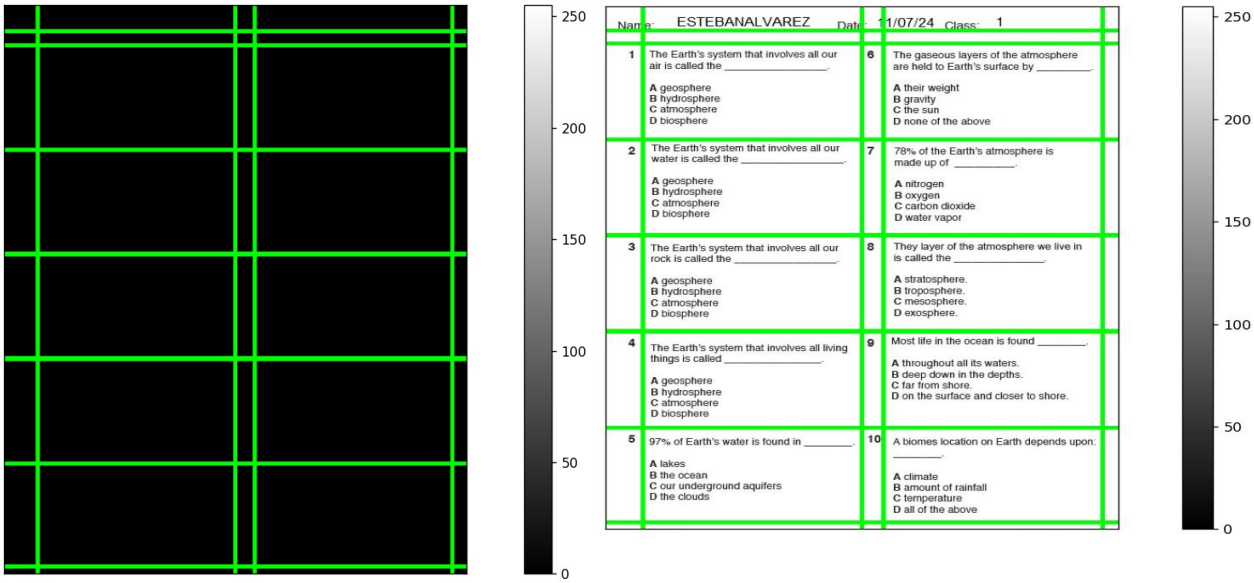


3. Detección de líneas utilizando la Transformada de Hough

Aplica la Transformada de Hough (`cv2.HoughLines()`) para encontrar líneas rectas en una imagen binaria (como la imagen de bordes obtenida con Canny)

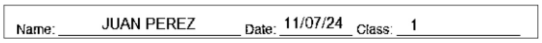
```
lines = cv2.HoughLines(edges, rho=1, theta=np.pi/180, threshold=250)
```

- Parámetros:
  - `edges`: La imagen de bordes sobre la que se buscan líneas.
  - `rho=1`: Resolución en píxeles del parámetro distancia.
  - `theta=np.pi/180`: Resolución angular (en radianes) del parámetro de ángulo.
  - `threshold=250`: Umbral mínimo para considerar que una acumulación corresponde a una línea. Cuanto más alto es el valor, más fuerte tiene que ser la evidencia de una línea.



Con las coordenadas de las líneas, se identifican las coordenadas del encabezado y de cada una de las 10 preguntas y se separan las imágenes.

Encabezado.



Área de una pregunta.

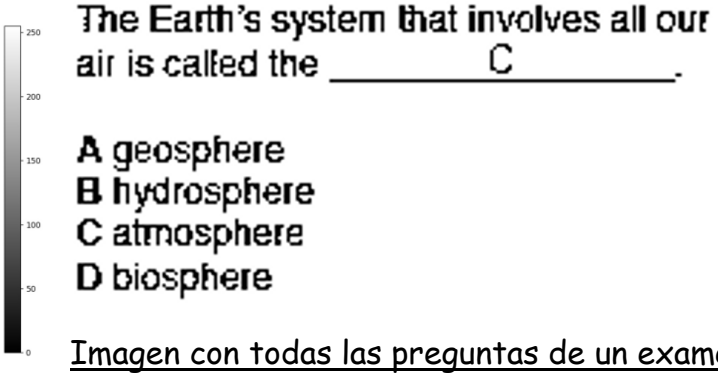
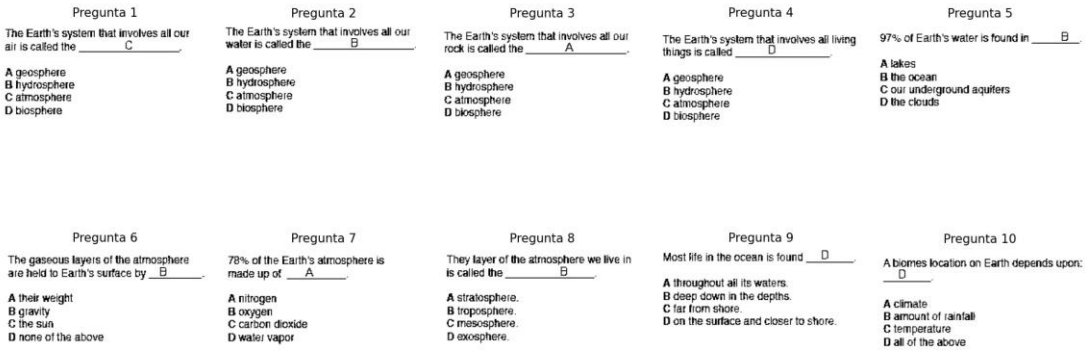


Imagen con todas las preguntas de un examen identificadas.



- Identificación de las líneas debajo de cada respuesta

Función

```
def identify_lines(questions: list) -> tuple:
```

1. Detecta componentes conectados (8-vecinos) dentro de la imagen de cada pregunta (`cv2.connectedComponentsWithStats()`).
2. Se aplican condiciones de filtro basadas en el tamaño del componente. Solo se seleccionan componentes cuyo ancho sea mayor a 50 píxeles y cuya altura sea menor a 3 píxeles (tamaño mínimo de las líneas).
3. Se extrae la región de interés alrededor del componente filtrado usando las coordenadas ajustadas.

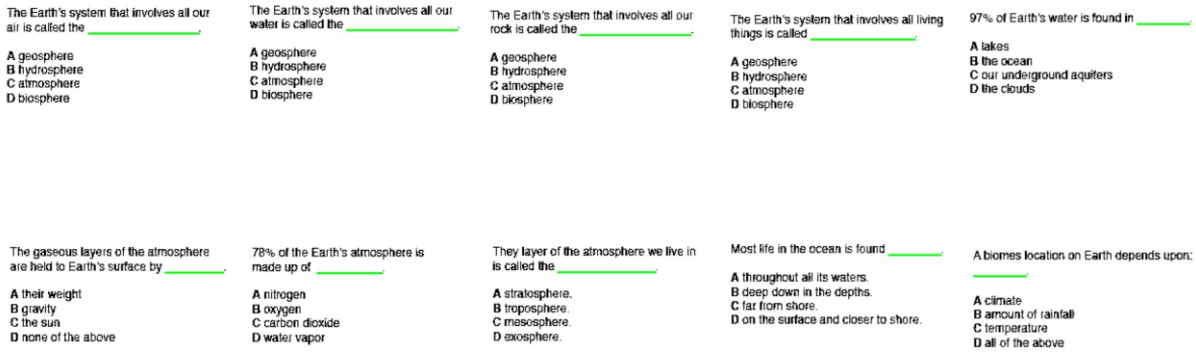
```
num_labels, labels, stats, centroids =
cv2.connectedComponentsWithStats(255 - question, connectivity,
cv2.CV_32S)
```

- Parámetros:
  - `255-question`: La imagen se invierte quedando el fondo negro (píxeles de valor 0) y los objetos de interés en blanco (píxeles de valor 255). Esto se realiza porque la función no detecta bien con la imagen original de fondo blanco y texto negro.
  - `connectivity=8`: Este valor define la conectividad. Un valor de 8 indica que los píxeles se consideran conectados si comparten algún borde o vértice con otro píxel blanco (8-vecinos).
  - `cv2.CV_32S`: Tipo de datos de salida para las etiquetas.

- Retornos:

- `num_labels`: El número de etiquetas o componentes conectados detectados (incluido el fondo como el componente 0).
- `labels`: Una matriz de las mismas dimensiones que la imagen donde cada píxel tiene asignado el número de la etiqueta a la que pertenece.
- `stats`: Una matriz de estadísticas, donde cada fila corresponde a un componente conectado y contiene los siguientes valores: `[x, y, width, height, area]`, es decir, las coordenadas del rectángulo delimitador, su tamaño y el área del componente.
- `centroids`: Las coordenadas del centroide de cada componente conectado.

Imagen con todas las líneas debajo de las preguntas de un examen identificadas.



- Identificación de los datos del encabezado

Función

```
def analyze_header(file: str) -> tuple:
```

1. Para garantizar esta funcionalidad se vuelve a leer como imagen el examen.
2. Mediante el umbralado se construye una imagen binaria en la que el fondo es negro (false) y los elementos de la imagen son blancos (true).
3. Se cuentan los píxeles blancos de manera horizontal para luego filtrar únicamente aquellas filas que portan una significativa cantidad de píxeles blancos.
4. Se determina que el índice del primer elemento del array filtrado. Este índice corresponde con el límite del encabezado, procediéndose así a segmentar la imagen para únicamente retener el encabezado.
5. Para identificar a los campos del encabezado se buscan los respectivos índices horizontales y verticales que los delimitan.
6. Se aplica el algoritmo "connectedComponentsWithStats" de manera tal que se identifican los distintos caracteres que integran cada campo.

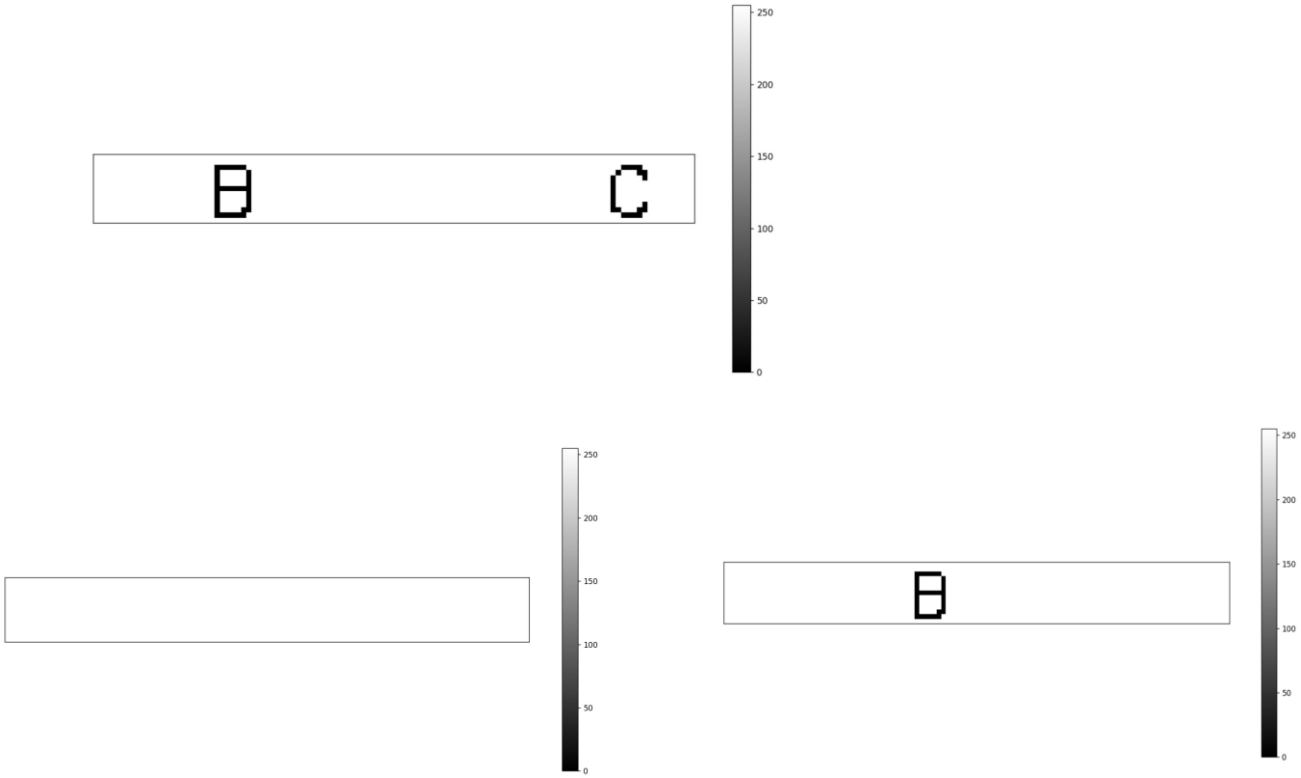
7. Se utiliza la estadística que devuelve este algoritmo para contar la cantidad de caracteres y a su vez la cantidad de palabras presentes en cada campo. Esto último se estima al calcular la cantidad de píxeles que separan horizontalmente a los caracteres continuos, infiriéndose que una distancia mayor a 4 píxeles corresponde con una separación de palabras.
8. En función a los cálculos efectuados en el punto anterior, se analiza si cada campo fue rellenado correctamente. Por último, la función devuelve el resultado del anásis en cuestión y la subimagen que corresponde al campo 'Name'.

- Identificación de las respuestas a cada pregunta

Función

```
def indetify_answers(lines_answer: list) -> list:
```

9. Verifica que en el área exista una sola respuesta. Detecta componentes conectados (8-vecinos) dentro de la zona.



10. Si la imagen tiene un solo componente, identifica la letra de la respuesta, detectando los contornos y el número de hijos (`cv2.findContours()`). En el caso de las letras A y D se diferencian por la relación de áreas entre el padre y el hijo.

A: contorno con 1 hijo. Relación área_hijo/área_padre < 0.65

B: contorno con 2 hijos

C: contorno sin hijos

D: contorno con 1 hijo. Relación área_hijo/área_padre > 0.65

```
contours, hierarchy = cv2.findContours(255-line_answer, cv2.RETR_CCOMP, cv2.CHAIN_APPROX_SIMPLE)
```

Descripción: Esta función encuentra los contornos en una imagen binaria. Los contornos son curvas cerradas que delimitan las formas o los objetos en la imagen. Aquí la imagen invertida (`255 - line_answer`) se usa para detectar objetos blancos sobre fondo negro.

- Parámetros:

○ `255 - line_answer`: La imagen se invierte quedando el fondo negro y los objetos de interés en blanco. Esto se realiza porque la función no detecta bien con la imagen original de fondo blanco y texto negro.

○ `cv2.RETR_CCOMP`: Este modo de recuperación de contornos obtiene todos los contornos y organiza la jerarquía de manera que cada contorno tenga información sobre su contorno padre y sus hijos.

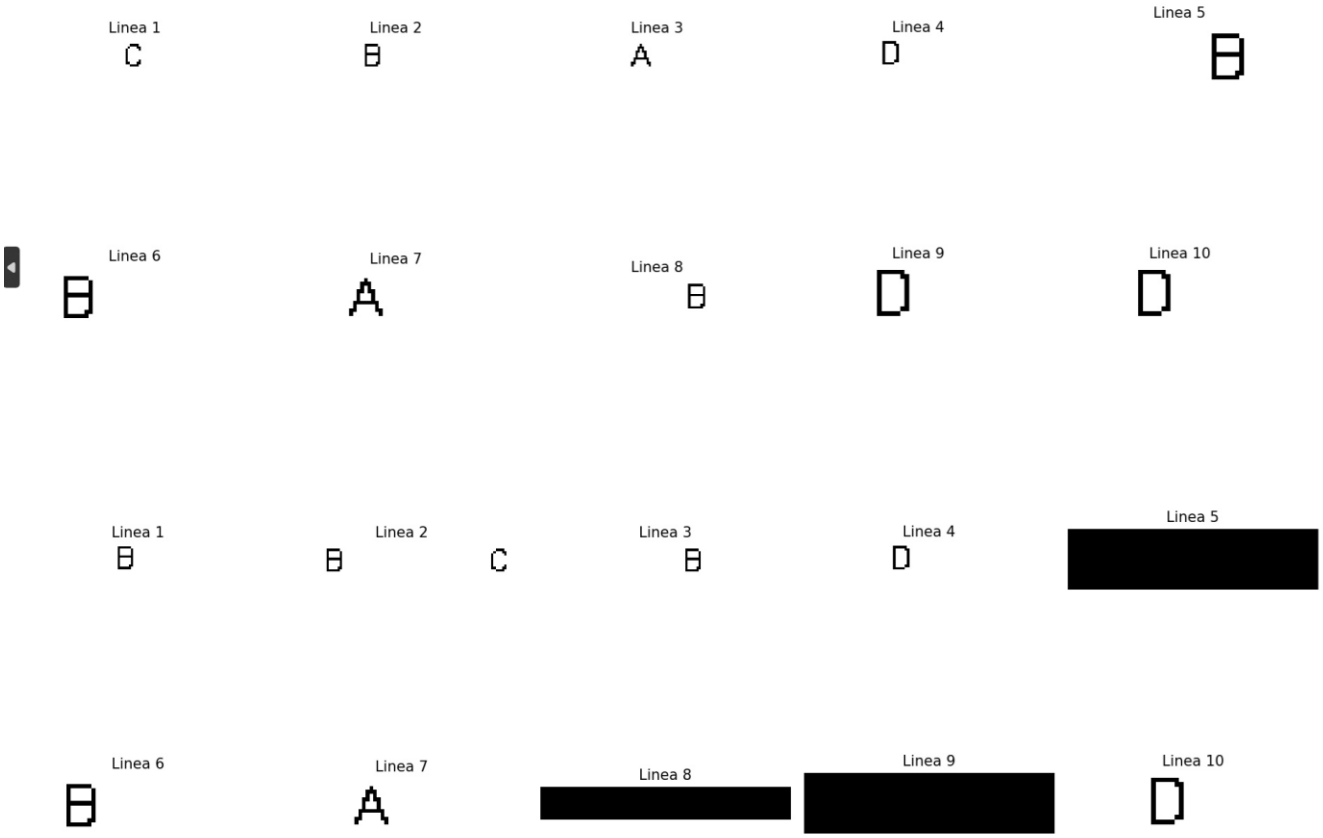
○ `cv2.CHAIN_APPROX_SIMPLE`: Método de aproximación de contornos que guarda solo los puntos esenciales del contorno, lo que reduce el uso de memoria..

- Retornos:

○ `contours`: Lista de contornos encontrados. Cada contorno es un array de puntos.

○ `hierarchy`: Información sobre la relación jerárquica entre los contornos. Es una matriz con 4 valores por contorno: `[next, previous, first_child, parent]`.

Imágenes de detección de letras para distintos exámenes. El área en negro es cuando tiene un número de respuestas distinto de uno.



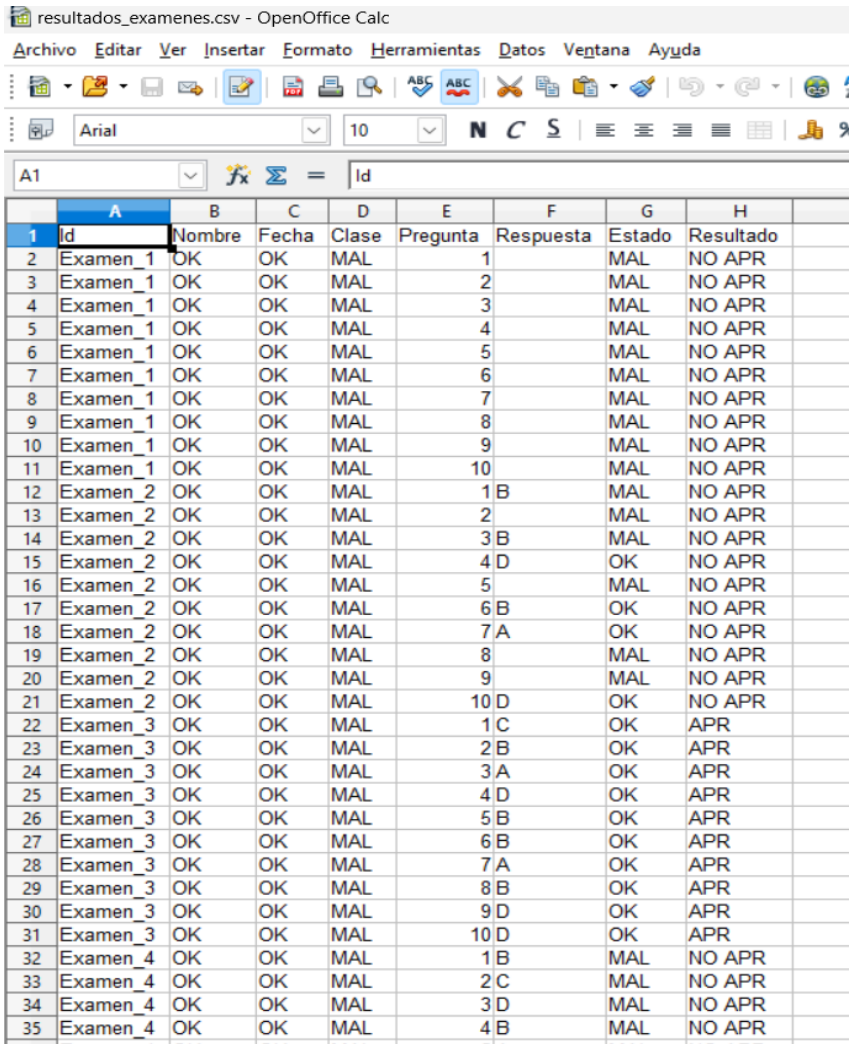
11. Devuelve un array con las respuestas. Las respuestas faltantes o con errores de formato, se devuelven vacías.

- Corrección de los exámenes

1. Se comparan las respuestas de cada exámen con las respuestas correctas.
2. Se evalúa si cada exámen está aprobado.
3. Se muestran los resultados.

Id: Examen_2
Nombre: OK
Fecha: OK
Clase: MAL
Respuestas:
Pregunta 1: MAL
Pregunta 2: MAL
Pregunta 3: MAL
Pregunta 4: OK
Pregunta 5: MAL
Pregunta 6: OK
Pregunta 7: OK
Pregunta 8: MAL
Pregunta 9: MAL
Pregunta 10: OK
Resultado: NO APR

4. Se corrigen todos los exámenes de la carpeta seleccionada por el usuario y se guardan los resultados en formato .csv y .txt

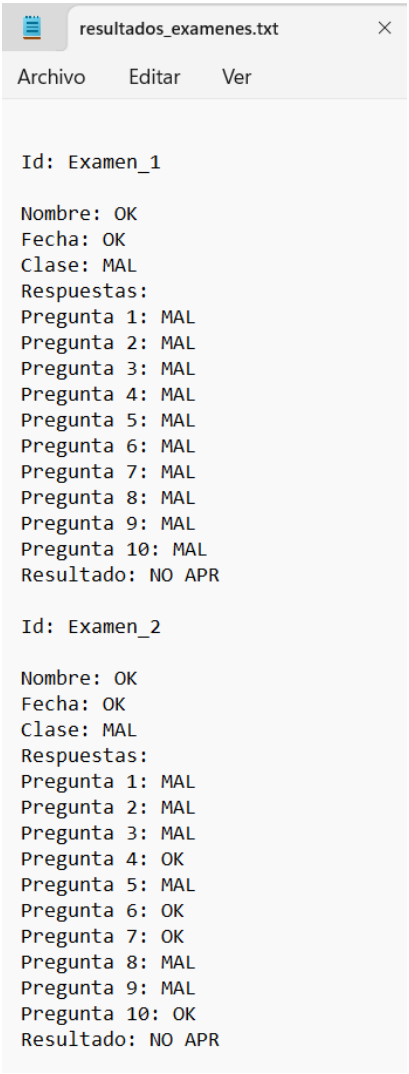

resultados_examenes.csv - OpenOffice Calc

Archivo Editar Ver Insertar Formato Herramientas Datos Ventana Ayuda

Arial 10

A1 Id

	A	B	C	D	E	F	G	H
1	Id	Nombre	Fecha	Clase	Pregunta	Respuesta	Estado	Resultado
2	Examen_1	OK	OK	MAL	1		MAL	NO APR
3	Examen_1	OK	OK	MAL	2		MAL	NO APR
4	Examen_1	OK	OK	MAL	3		MAL	NO APR
5	Examen_1	OK	OK	MAL	4		MAL	NO APR
6	Examen_1	OK	OK	MAL	5		MAL	NO APR
7	Examen_1	OK	OK	MAL	6		MAL	NO APR
8	Examen_1	OK	OK	MAL	7		MAL	NO APR
9	Examen_1	OK	OK	MAL	8		MAL	NO APR
10	Examen_1	OK	OK	MAL	9		MAL	NO APR
11	Examen_1	OK	OK	MAL	10		MAL	NO APR
12	Examen_2	OK	OK	MAL	1	B	MAL	NO APR
13	Examen_2	OK	OK	MAL	2		MAL	NO APR
14	Examen_2	OK	OK	MAL	3	B	MAL	NO APR
15	Examen_2	OK	OK	MAL	4	D	OK	NO APR
16	Examen_2	OK	OK	MAL	5		MAL	NO APR
17	Examen_2	OK	OK	MAL	6	B	OK	NO APR
18	Examen_2	OK	OK	MAL	7	A	OK	NO APR
19	Examen_2	OK	OK	MAL	8		MAL	NO APR
20	Examen_2	OK	OK	MAL	9		MAL	NO APR
21	Examen_2	OK	OK	MAL	10	D	OK	NO APR
22	Examen_3	OK	OK	MAL	1	C	OK	APR
23	Examen_3	OK	OK	MAL	2	B	OK	APR
24	Examen_3	OK	OK	MAL	3	A	OK	APR
25	Examen_3	OK	OK	MAL	4	D	OK	APR
26	Examen_3	OK	OK	MAL	5	B	OK	APR
27	Examen_3	OK	OK	MAL	6	B	OK	APR
28	Examen_3	OK	OK	MAL	7	A	OK	APR
29	Examen_3	OK	OK	MAL	8	B	OK	APR
30	Examen_3	OK	OK	MAL	9	D	OK	APR
31	Examen_3	OK	OK	MAL	10	D	OK	APR
32	Examen_4	OK	OK	MAL	1	B	MAL	NO APR
33	Examen_4	OK	OK	MAL	2	C	MAL	NO APR
34	Examen_4	OK	OK	MAL	3	D	MAL	NO APR
35	Examen_4	OK	OK	MAL	4	B	MAL	NO APR


resultados_examenes.txt

Archivo Editar Ver

```
Id: Examen_1

Nombre: OK
Fecha: OK
Clase: MAL
Respuestas:
Pregunta 1: MAL
Pregunta 2: MAL
Pregunta 3: MAL
Pregunta 4: MAL
Pregunta 5: MAL
Pregunta 6: MAL
Pregunta 7: MAL
Pregunta 8: MAL
Pregunta 9: MAL
Pregunta 10: MAL
Resultado: NO APR

Id: Examen_2

Nombre: OK
Fecha: OK
Clase: MAL
Respuestas:
Pregunta 1: MAL
Pregunta 2: MAL
Pregunta 3: MAL
Pregunta 4: OK
Pregunta 5: MAL
Pregunta 6: OK
Pregunta 7: OK
Pregunta 8: MAL
Pregunta 9: MAL
Pregunta 10: OK
Resultado: NO APR
```

Notas

Las funciones devuelven las imágenes y las coordenadas para que, de manera general, se pueda trabajar sobre las imágenes generadas o directamente en la imagen original, con sus posiciones relativas.

Todas las respuestas son listas correspondientes a todos los exámenes a corregir.

```python
def create_questions(image: np.ndarray) -> tuple:
```

devuelve

```python
return header, header_coords, questions, questions_coords
```

```python
def identify_lines(questions: list) -> tuple:
```

devuelve

```
return lines_answer, lines_answer_coords
```

```
def indetify_answers(lines_answer: list) -> list:
```

devuelve

```
return answers
```

Los gráficos y archivos generados durante el desarrollo, para ir validando los resultados parciales, están comentadas o se eliminaron del código.

- **Durante el desarrollo, los inconvenientes más notorios fueron:**

- En las funciones `cv2.connectedComponentsWithStats()` y `cv2.findContours()`, se presentaron problemas en la detección de componentes y contornos, respectivamente, al trabajar con la imagen original binarizada (con fondo blanco y letras en negro). Al invertir la imagen (255 - imagen), se solucionaron.
- Encontrar los umbrales adecuados para identificar todas las líneas debajo de las respuestas.
- Encontrar los valores adecuados para diferenciar la letra A de la D.

- **Repositorio**

https://github.com/Facunditos/Facunditos-PDI-TUIA-TP1-LopezCrespo-Flaibani-Dito.git