Universidad Nacional de Rosario

Facultad de Ciencias Exactas, Ingeniería y Agrimensura

IA4.4 Procesamiento de Imágenes

Trabajo Práctico N°1

Tecnicatura Universitaria en Inteligencia Artificial

Integrantes (Grupo 16):

Alsop, Agustín (A-4651/7)

Asad, Gonzalo (A-4595/1)

Castells, Sergio (C-7334/2)

Hachen, Rocío (H-1184/3)

Docentes:

Gonzalo Sad

Julián Alvarez

Juan Manuel Calle

Fecha: 21/10/2024

1. INTRODUCCIÓN

En este trabajo se realizan dos ejercicios sobre diferentes imágenes, aplicando las técnicas aprendidas en clase de transformación, filtrado, detección de bordes y segmentación.

El primer ejercicio, consistió en aplicar la ecualización local del histograma sobre una imagen para extraer detalles escondidos. El segundo ejercicio consistió en analizar las resoluciones de algunos exámenes, evaluarlos y analizar si fueron completados correctamente en su encabezado. Durante el trabajo se explican las diferentes funciones que fueron creadas para las resoluciones, junto con capturas de imágenes intermedias que fueron obtenidas durante el proceso de análisis.

2. Problema 1 – Ecualización Local de Histograma

2.1 Descripción

El problema consistió en desarrollar una función de Python que implemente la ecualización local del histograma, recibiendo como parámetros de entrada una imagen a procesar (provista con el enunciado) y el tamaño de la ventana de procesamiento (M x N). La imagen sin procesar se ve a continuación:

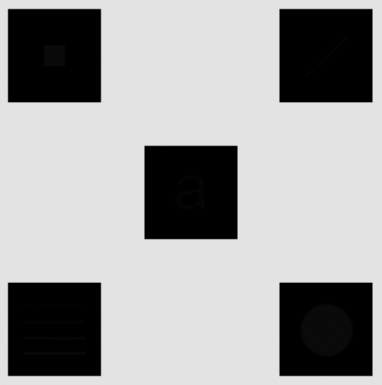


Figura 1: Imagen con detalles en diferentes zonas.

Se debió utilizar la función desarrollada para encontrar detalles ocultos escondidos en las diferentes zonas de la misma, analizando la influencia de la ventana en los resultados obtenidos.

2.2 Resolución

Para la resolución del problema fue definida una función `localHistEq`, que implementa la ecualización local del histograma. La función recibe tres parámetros:

- `Img(np.ndarray)`: Imagen a procesar. Debe ser ingresada en escala de grises.
- `Width(int)`: Ancho de la ventana de procesamiento. Debe ser un entero positivo impar.
- `Height(int)`: Alto de la ventana de procesamiento. Debe ser un entero positivo impar.

La función retorna un `np.ndarray`, que es la imagen resultante luego de aplicar la ecualización local del histograma.

`localHistEq` realiza primero una validación de los parámetros ingresados y retorna un mensaje de error si alguno de ellos no cumple con los requisitos. A continuación, agrega un borde replicado los extremos de la imagen alrededor de la misma, para permitir el

procesamiento de píxeles cercanos a los bordes y asegurando que todos los píxeles puedan ser ecualizados sin perder información en los extremos. La función luego inicializa una imagen vacía de las mismas dimensiones que la imagen original para almacenar el resultado final del procesamiento, que consiste en recorrer cada píxel de la imagen original, extrayendo una ventana local centrada en cada uno de ellos y aplicándoles una ecualización global.

Se define una lista con diferentes tamaños de ventanas para la ecualización local y la iteramos, ejecutando la función `localHistEq` en cada iteración parametrizándola con cada tamaño de ventana. Finalmente, los resultados son mostrados en una misma imagen.

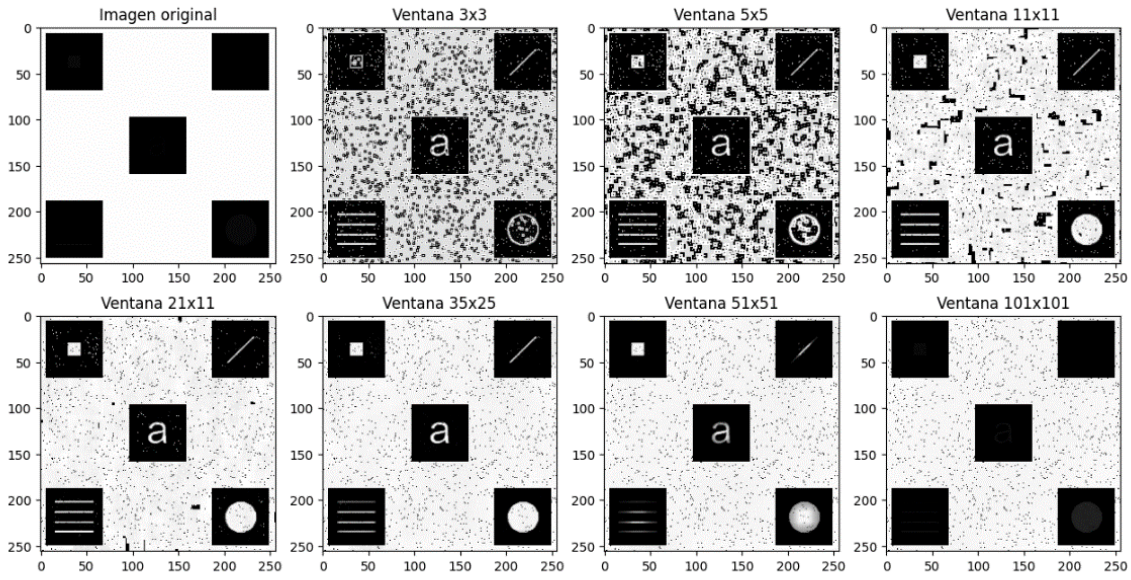


Figura 2: Imagen ecualizada localmente con diferentes tamaños de ventana.

Durante el desarrollo se presentaron dos problemas, tomando como aprendizaje que se debía poner atención en algunos detalles:

- La ventana de procesamiento debe tener un tamaño M x N con M y N impares, para así tener su centro en un único punto de muestra. Esto nos garantizó que la modificación de la señal se realice de manera simétrica alrededor de este centro, evitando desplazamientos de fase y distorsiones.
- Agregar un borde replicando alrededor de la imagen antes de ecualizar, para no perder información en los bordes.

2.3 Conclusiones

Con un tamaño de ventana pequeño se pudieron divisar las formas ocultas dentro de la imagen, siendo de izquierda a derecha y de arriba abajo: un cuadrado relleno, una línea recta inclinada a 45 grados, una letra ‘a’ minúscula, cuatro rayas paralelas horizontales y un círculo relleno. Aplicar una ecualización del histograma consigue tener un contraste mayor, logrando que los niveles de intensidad sean igualmente probables. Los cuadrados oscuros tenían objetos dentro con niveles de intensidad ligeramente diferentes a los del recuadro, que fueron “resaltados” mediante la ecualización.

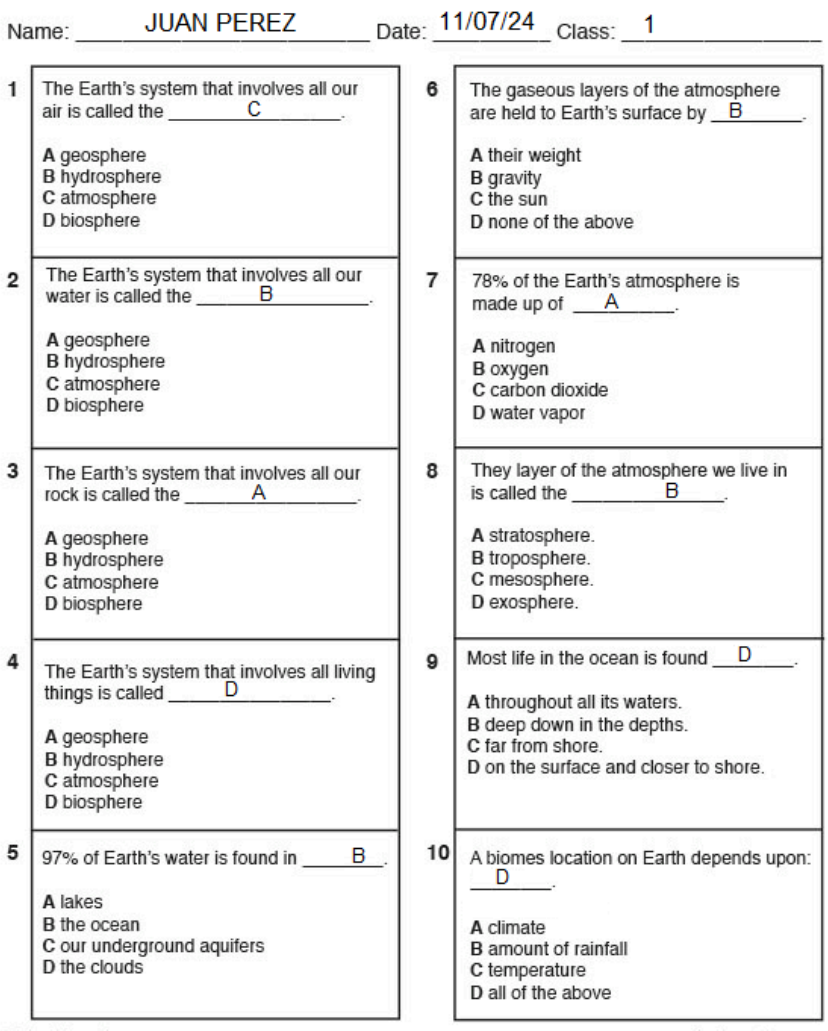
Alterar el tamaño de la ventana afecta en qué tan definidos se encuentran los objetos dentro de los cuadrados. Con un tamaño de ventana pequeña se genera mucho ruido sobre la imagen, haciéndola parecer más granulosa. Al aumentar el tamaño de ventana, la ecualización se realiza en áreas más amplias, suavizando algunos detalles y reduciendo el ruido. Sin embargo, de elegir un tamaño de ventana demasiado grande, los detalles se suavizan demasiado, al punto en que pueden hacer desaparecer los detalles ocultos en los cuadrados.

Elegir el tamaño de ventana para este ejercicio se hace una tarea subjetiva, donde se debe hacer un balance entre los detalles ganados y la cantidad de ruido. Es por eso que se concluye que el tamaño de ventana ideal para este ejercicio podría ser el de 35 x 25, donde las propiedades buscadas se preservan de la mejor manera.

3. Problema 2 – Corrección de *Multiple Choice*

3.1 Descripción

El problema consistió en desarrollar un script en Python que corrigiera automáticamente un conjunto de cinco exámenes. Los exámenes son imágenes con un encabezado con datos personales (nombre, fecha y clase) y un cuerpo con diez preguntas de *multiple choice* con cuatro opciones para cada una de ellas: A, B, C y D.



Name: JUAN PEREZ 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

Figura 3: Esquema del examen.

Teniendo en cuenta que las respuestas correctas deberían ser:

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

El algoritmo debe:

1. Tomar como entrada la imagen de un examen y mostrar por pantalla cuáles de las respuestas eran correctas y cuáles incorrectas (incluyen aquellas donde se eligieron más de una opción o donde no se eligió ninguna).
2. Validar los datos del encabezado y mostrar por pantalla el estado de cada campo, teniendo en cuenta que:
    a. *Name*: debe contener al menos dos palabras y no más de 25 caracteres.
    b. *Date*: deben ser 8 caracteres formando una sola palabra.
    c. *Class*: debe ser un único carácter.
3. 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 correspondían a un examen aprobado de uno desaprobado.

3.2 Resolución

Se implementó un algoritmo que recorre todas las imágenes de los exámenes y las procesa por etapas, donde cada función desarrollada extrae secciones o características específicas y calcula resultados, entregándolos finalmente en una única imagen de salida.

3.2.1 Función imShow

Función para visualizar imágenes. Simplemente parametriza una figura de la librería `matplotlib`. La función recibe los parámetros:

- `img(np.ndarray)`: Imagen a visualizar.
- `new_fig(bool)`: Si es Verdadero, se creará una nueva ventana de imagen.
- `title(str)`: Título de la imagen, por default es None.
- `color_img(bool)`: Si es Verdadero, la imagen será considerada a color. De lo contrario, será considerada en escala de grises.
- `blocking(bool)`: Si es Verdadero, la ejecución del código se interrumpirá hasta que la ventana de la imagen sea cerrada.
- `colorbar(bool)`: Si es Verdadero, se visualizará la escala de colores a la derecha de la imagen.
- `ticks(bool)`: Si es Verdadero, xticks e yticks son deshabilitados.

La función no retorna nada.

3.2.2 Función letterAnswer

Función para identificar una letra a partir de una imagen y retornarla. Las letras identificadas pueden ser A, B, C o D, correspondiente a los cuatro resultados posibles a una pregunta del examen. La función recibe el parámetro:

- `Letter_box(np.ndarray)`: Imagen a procesar.

`letterAnswer` recibe la sección de la imagen donde se encuentra la respuesta del estudiante a una pregunta y primero la convierte en escala de grises, luego cuenta la cantidad de píxeles más oscuros guardándolos en una variable y posteriormente aplica un umbralado sobre la misma. Si la cantidad de píxeles más oscuros supera cierto valor, asume que el estudiante ingresó más de una letra e invalida su respuesta. Si no se detectaron píxeles oscuros, asume que el estudiante no respondió la pregunta. En caso de que las condiciones anteriores no se cumplan, busca contornos en la imagen usando `findContours` de `OpenCV` y los dibuja sobre la imagen. Si la cantidad de contornos es de 2 o 4, la función identifica que la respuesta del estudiante fue C o B respectivamente y retorna un `str` con su respuesta. Sin embargo, si la cantidad de contornos es de 3, busca componentes conectados usando `connectedComponentsWithStats` de `OpenCV` y detecta la respuesta del estudiante en base al área de una de las componentes conectadas, retornando un `str` con las respuestas A o D. En caso de existir más de 4 componentes conectadas la respuesta se considera inválida.

3.2.3 Función lineDetector

Función para identificar líneas rectas en una imagen que recibe como parámetro, retornando una lista con las coordenadas de las líneas identificadas. La función recibe como parámetros:

- `src(np.ndarray)`: Imagen a procesar.
- `th(int)`: Umbral a utilizar en la función HoughLines.

`lineDetector` recibe la imagen de un examen completo, la transforma a escala de grises y luego, mediante la función `Canny` de `OpenCV`, identifica los bordes de la misma. A continuación, a través de la función `HoughLines` de `OpenCV` se detectan las líneas rectas de la imagen, si ninguna recta es detectada se baja el umbral para conseguirlo. Las líneas obtenidas por `HoughLines` se encuentran en coordenadas polares, sin embargo, es más fácil usarlas en coordenadas cartesianas. Por ese motivo se iteran todas las líneas obtenidas y se convierten al sistema cartesiano mediante un proceso algorítmico, generando para cada recta una lista de tuplas que contienen sus coordenadas de inicio de fin. La información de las rectas es guardada en una lista que luego es iterada por una última vez, donde se buscan líneas muy cercanas entre sí. Si las rectas están demasiado

cercanas, significa que son ambos lados de una recta en el examen, por lo que se promedian sus coordenadas y se obtiene una recta intermedia entre ambas, guardándola en la lista final de rectas y eliminando las rectas cercanas entre sí.

Finalmente, se retorna la lista con información de las rectas en la imagen estructurada como `list[list[tuple]]`.

3.2.4 Función lineOrientation

Función para clasificar líneas entre horizontales y verticales, retornando una tupla con las rectas clasificadas. La función recibe como parámetro:

- `line_list(list[list[tuple]])`: Lista con coordenadas cartesianas de rectas identificadas en una imagen.

`lineOrientation` recibe una lista con las coordenadas cartesianas de rectas identificadas en una imagen y la itera. Si detecta que la ordenada al eje "y" de la primera coordenada de la recta tiene un valor específico, determina que la recta es horizontal y la agrega a una lista de rectas horizontales, caso contrario se evalúa el valor de la componente "x" y se determina si se la agrega a la lista de rectas verticales o si se descarta (línea oblicua). A continuación, ordena las rectas dentro de las listas de acuerdo a sus coordenadas.

La función retorna una tupla de listas que contienen las rectas ordenadas y clasificadas, donde el primer elemento de la tupla es una lista de listas de coordenadas de rectas horizontales y el segundo elemento de la tupla contiene lo mismo, pero para rectas verticales.

3.2.5 Función questionROIDetector

Función para identificar las secciones de preguntas de un exámen, basado en intersecciones entre líneas horizontales y verticales, retornando una lista con los recortes de las preguntas. La función recibe los parámetros:

- `v_lines(list[list[tuple]])`: Lista de listas de coordenadas de líneas verticales.
- `h_lines(list[list[tuple]])`: Lista de listas de coordenadas de líneas horizontales.
- `img(np.ndarray)`: Imagen a partir de la cual se obtendrán las ROIs.
- `show(bool)`: Si es verdadero, se visualizarán los ROIs obtenidos.

`questionROIDetector` recibe la imagen de un examen y las coordenadas de sus rectas horizontales y verticales ordenadas. Recorre las coordenadas de las rectas verticales delimitando un área entre una recta y la que le sigue. A continuación, delimita áreas

iterando las rectas horizontales, obteniendo las secciones donde están ubicadas las preguntas. Hace un recorte de la ROI y la agrega a una lista, repitiendo el proceso hasta recorrer toda la imagen. Las preguntas del examen son recorridas de arriba-abajo y de izquierda-derecha, en ese orden.
La función retorna una `list[np.ndarray]` que contiene recortes del examen donde se encuentran las preguntas.

3.2.6 Función headerDetector

Función para identificar la sección del encabezado de una imagen, basado en sus líneas horizontales y retornando un recorte con el mismo. La función recibe los parámetros:

- `h_lines(list[list[tuple]])`: Lista de listas de coordenadas de líneas horizontales.
- `img(np.ndarray)`: Imagen a partir de la cual se obtendrá la ROI.
- `show(bool)`: Si es verdadero, se visualizará la ROI obtenida.

`headerDetector` recibe la imagen de un examen y las coordenadas de sus rectas horizontales. Define la región del encabezado, que comienza en la esquina superior izquierda de la imagen del examen y finaliza en el extremo derecho de la primera línea horizontal detectada en la misma imagen. Se recorta el área delimitada y se retorna como un `np.ndarray`.

3.2.7 Función letterBoxDetector

Función para identificar la región que comprende a un caracter, retornando una lista con las regiones obtenidas. La función se parametriza con:

- `img(np.ndarray)`: Imagen a procesar.
- `show(bool)`: Si es verdadero, se visualizará la región obtenida.
- `header(bool)`: Si es verdadero, se analizará especifícamente una sección del encabezado de un examen.

`letterBoxDetector` recibe el recorte de imagen de un examen que comprende a una pregunta y su resolución, lo convierte a escala de grises y le aplica un umbralado. Luego, mediante la función `connectedComponents` de `OpenCV` obtiene las componentes comentadas de la imagen. Se iteran todas las todas las etiquetas obtenidas, se las convierte a formato `UINT8` y se les aplica una máscara antes de obtener un recuadro que comprenda a cada componente usando la función `boundingRect` de `OpenCV`, consiguiendo así sus coordenadas y dimensiones. Se busca la componente conectada de la recta sobre la cual el estudiante completa con respuestas o datos personales, que al ser tanto más larga que ancha su relación de aspecto será mucho más alto que el del resto de componentes de la imagen. Finalmente se recorta un área de la

imagen sobre la recta, de un tamaño específico de acuerdo a si se trata de un encabezado o no.

Finalmente, se retorna el recorte de la imagen como un `np.ndarray`.

3.2.8 Función headerValidator

Función para analizar la forma de diferentes sectores del encabezado y retorna una salida booleana que indica si el sector analizado está correcto. Recibe como parámetros:

- `img(np.ndarray)`: Imagen del encabezado a procesar.
- `field(str)`: Sector a analizar (nombre, fecha, clase).

`headerValidator` recibe la imagen del encabezado a analizar, la transforma a escala de grises, le aplica un umbralado y posteriormente busca componentes conectadas haciendo uso de la función `connectedComponentsWithStats` de `OpenCV`. Luego, dependiendo del campo (`field`) a analizar realiza distintas evaluaciones:

Si se desea analizar el "nombre" se verifica que el número de componentes conectadas sea al menos 3 (2 correspondientes al nombre y apellido y una al fondo de la imagen) y además la existencia de un espacio entre "nombre" y "apellido". En caso de cumplirse esas condiciones la salida será "True". De lo contrario, el retorno será "False".

Si el sector a analizar es el de "fecha" se verifica que el número de componentes conectadas sea 9 (8 correspondientes a la fecha y una por el fondo de la imagen). Si esto se cumple, la función arrojará un "True" como salida.

En caso de que el sector a analizar sea "clase" se verifica que el número de componentes conectadas sea 2 (1 correspondiente a la clase y una por el fondo de la imagen). Si esa condición se satisface la salida será "True".

3.2.9 Main

El script del código principal (main) estructura la ejecución del análisis de la imagen haciendo uso de las funciones descriptas anteriormente. Comienza cargando al entorno de trabajo las imágenes que usará hacia el final para generar el reporte de los resultados.

Figura 4: Respuestas o datos correctos.

Figura 5: Respuestas o datos incorrectos.

Figura 6: Examen aprobado.

Figura 7: Examen desaprobado.

Figura 8: Planilla de reporte de exámenes.

Se itera entre los cinco exámenes a evaluar para analizarlos de a uno y adjuntar los resultados en la planilla mostrada en la *Figura 8*. Se carga un examen al entorno de trabajo mediante la función `imread` de `OpenCV` y se utiliza la función `lineDetector` para encontrar las líneas rectas en la imagen, lo que ayudará más adelante a encontrar las secciones que contienen las preguntas con las respuestas de los estudiantes.

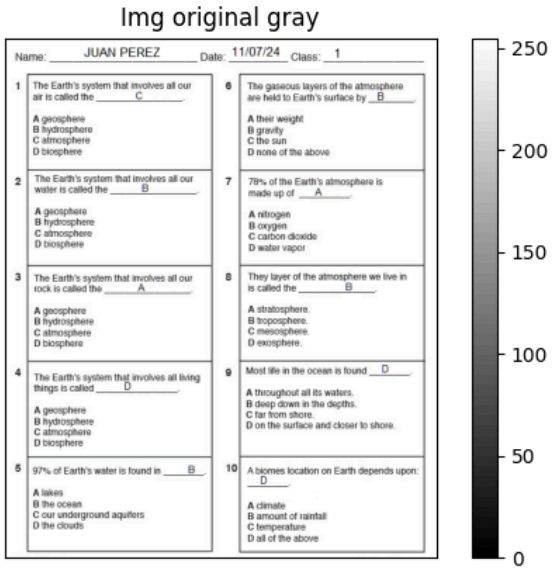


Figura 9: Examen original.

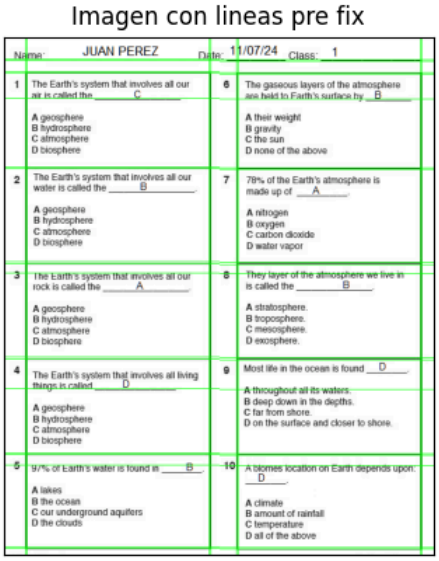


Figura 10: Examen con líneas detectadas por HoughLines.

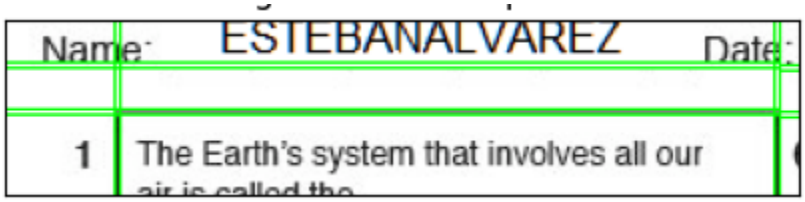


Figura 10.1: Recorte del examen con líneas detectadas por HoughLines.

Luego, graficamos las líneas encontradas en el examen, resultando en algo como lo que se ve en la *Figura 11 y Figura 11.1*.

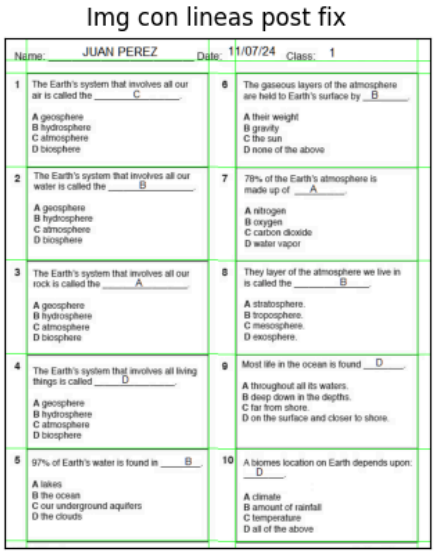


Figura 11: Resultado final luego de eliminar líneas indeseadas.

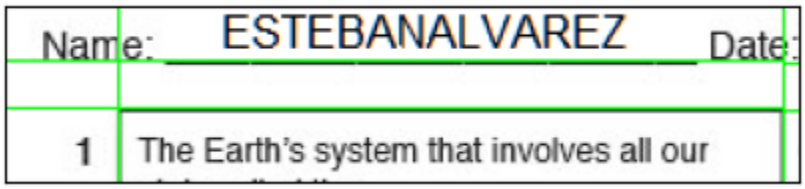


Figura 11.1: Recorte del resultado final luego de eliminar líneas indeseadas.

A continuación, se clasifican las líneas encontradas entre horizontales o verticales mediante la función `lineOrientation` para luego usarlas como parámetro dentro de la función `questionROIDetector`, retornando las regiones con las preguntas.

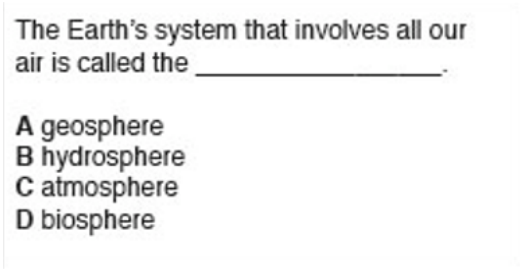


Figura 12: Ejemplo de ROI.

Habiendo generado una lista con los recortes de las preguntas, se la itera para encontrar las regiones donde se encuentran las respuestas del estudiante usando la función `letterBoxDetector`. En el proceso, los recortes con las preguntas se alteran para facilitar encontrar las ROIs.

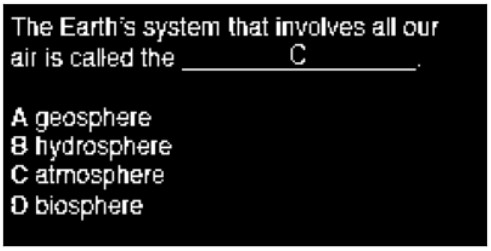


Figura 13: Pregunta luego de ser umbralada e invertida.

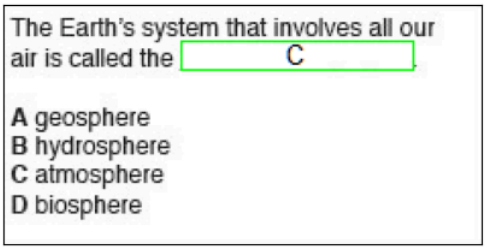


Figura 14: ROI reconocida.

Extrayendo finalmente la ROI que contiene una respuesta del estudiante.

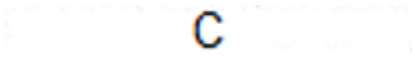


Figura 15: ROI recortada.

Se la inserta, junto a las otras respuestas, dentro de una lista. Las respuestas son enviadas a la función `letterAnswer` que encuentra las respuestas del estudiante en las ROIs y las guarda en una lista y luego las muestra por consola.

Figura 16: Ejemplo de detección de bordes para una respuesta.

A continuación se crea una lista con los resultados correctos esperados para cada pregunta y se inicializa un contador de respuestas correctas por parte del estudiante. Se iteran las respuestas del estudiante y se las compara con los resultados correctos esperados. Si coinciden, se copia la imagen de la *Figura 4* sobre la posición correcta en la planilla de reporte, caso contrario, se copia la imagen de la *Figura 5* sobre esa posición y se avanza a

la siguiente pregunta y posición. Los resultados de las respuestas también son mostradas por consola.

Habiendo analizado las respuestas del estudiante, se comienza a analizar el encabezado que es aislado usando la función `headerDetector`.

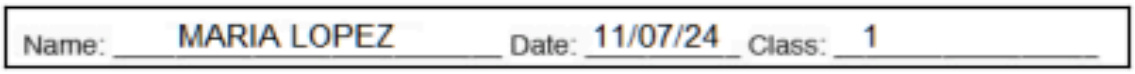


Figura 17: Ejemplo de encabezado aislado

Luego, se extraen los datos deseados del encabezado: el nombre del alumno, la fecha del examen y la clase. Esto se realiza con la función `letterBoxDetector`. La imagen que contiene el nombre extraído es redimensionada para luego poder insertarla en nuestro reporte final.

Figura 18: Ejemplo de nombre aislado.

Figura 19: Ejemplo de fecha aislada.

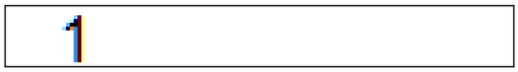


Figura 20: Ejemplo de clase aislada.

Los datos extraídos se validan con la función `headerValidator`. El resultado de esta validación también es incluída en el reporte final. El resultado de esta validación también es mostrado por consola.

Por último, se analiza la cantidad de respuestas correctas obtenidas por el alumno. Si este tiene 6 o más respuestas correctas, se agrega la imagen de la *Figura 6* sobre la posición correcta en la planilla del reporte, de lo contrario, se agrega la imagen de la *Figura 7.*

Finalmente, se visualiza el reporte final con la función `imshow`.

3.2.10 Análisis del desarrollo

Para la resolución del presente ejercicio (y también para el anterior) se optó, como criterio de diseño la creación de una librería de funciones propias (`customlib.py`) con el objeto de obtener un código principal más descomprimido.

Nos enfrentamos a varios problemas. El primero de ellos fue que al momento de detectar las líneas horizontales y verticales de la imagen la cantidad obtenida era el doble de la deseada. Esto es debido a que el proceso utilizado a través de la detección de bordes considera como tal a toda transición “blanco - negro”, “negro - blanco” y por lo tanto a cada línea le asocia dos bordes. Para resolver este inconveniente realizamos una promediación de líneas cercanas.

Otro problema con el que nos encontramos fue que, a la hora de detectar la cantidad de contornos para determinar las letras de las respuestas, no obteníamos una definición clara de la letra C. Para resolver esto, realizamos un ajuste del valor de “threshold” en la instancia de umbralado de forma experimental hasta obtener una respuesta acorde.

Para mostrar los resultados devueltos por el script decidimos elaborar una imagen acorde de modo tal de lograr una respuesta con visualización amigable al usuario (*Figura 8*).

3.3 Conclusiones

Por medio de las diversas técnicas de procesamiento de imágenes utilizadas, como el umbralado, detección de bordes (`Canny`), detección de líneas (`HoughLines`), detección de componentes conectadas, entre otras, fue posible afrontar el desafío de extraer toda la información deseada de las imágenes de los exámenes. De esta forma, logramos desarrollar un sistema que valida y corrige los exámenes con precisión. El reporte final muestra las correcciones de cada examen, donde se detalla si la pregunta fue respondida correctamente, o no. Además, contiene el nombre del alumno, como así también la validación de los datos de encabezado (nombre, fecha y clase). Por último, se establece si el alumno ha aprobado o desaprobado el examen. Este reporte se diseñó con el objetivo de que estuviera en un formato amigable para todos los usuarios.

Figura 21: Reporte final de las correcciones de los exámenes. Se observa que la mayoría de los alumnos desaprobó.

Este sistema podría adaptarse para funcionar con mayor cantidad de exámenes, lo cual sería una valiosa herramienta para reducir el tiempo y esfuerzo humano.