UNR Universidad Nacional de Rosario

FCEIA FACULTAD DE CIENCIAS EXACTAS, INGENIERIA Y AGRIMENSURA

UNIVERSIDAD NACIONAL DE ROSARIO

Facultad de Ciencias Exactas, Ingeniería y Agrimensura

TECNICATURA UNIVERSITARIA EN INTELIGENCIA ARTIFICIAL

Asignatura:

PROCESAMIENTO DE IMÁGENES

TRABAJO PRÁCTICO 1

TEMA:

Ecualización local de histograma - Corrección de Examen Multiple Choice

PROFESORES:

SAD, Gonzalo

ALVAREZ, Julián

CALLE, Juan Manuel

INTEGRANTES:

ARENAS, Agustín

GIAVENO, Santiago

FECHA:

21.04.25

ÍNDICE:

TEMA 2

1 | ECUALIZACIÓN LOCAL DE HISTOGRAMA 2

1.1 | DESCRIPCIÓN 2

1.2 | PROBLEMAS ENFRENTADOS 2

1.3 | TÉCNICAS APLICADAS 4

1.4 | CONCLUSIÓN 5

2 | CORRECCIÓN DE EXAMEN MULTIPLE CHOICE 5

2.1 | DESCRIPCIÓN 5

2.2 | PROBLEMAS ENFRENTADOS 6

2.3 | TÉCNICAS APLICADAS 6

2.4 | CONCLUSIÓN 8

Ecualización local de histograma - Correción de Examen Multiple Choice

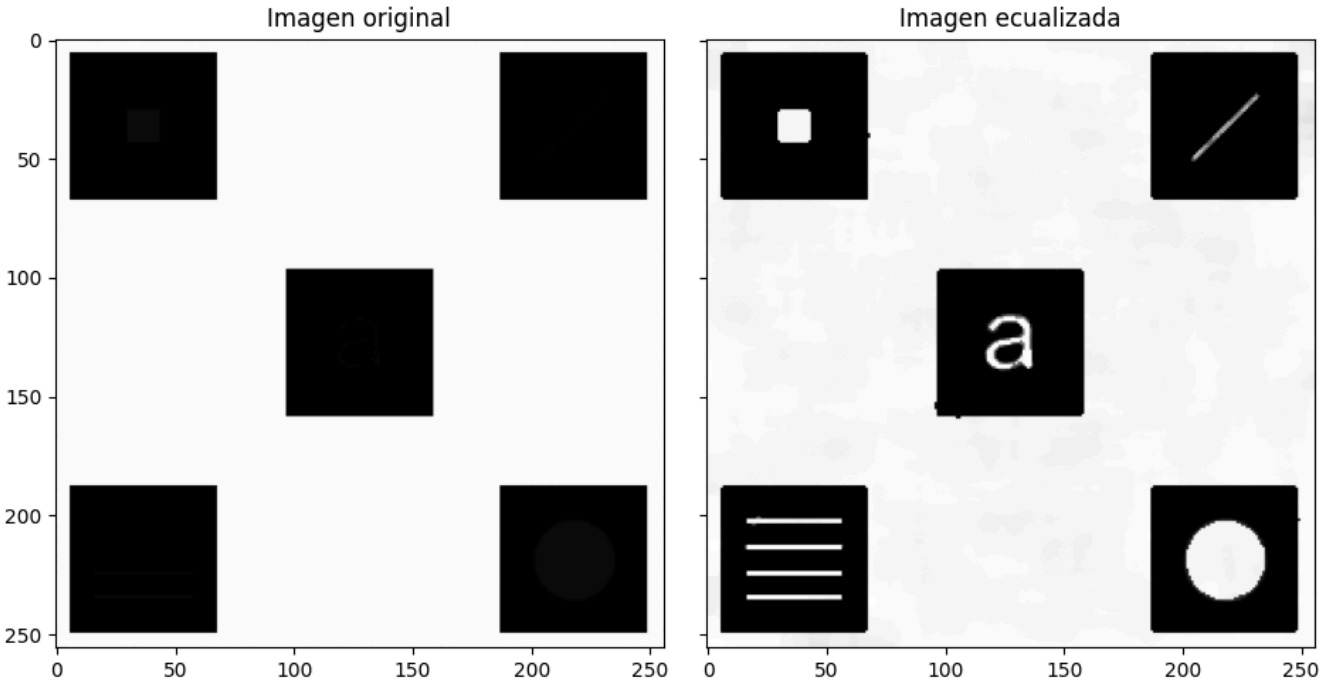
El informe consiste de una descripción del ejercicio, los problemas a los que nos enfrentamos, las diferentes técnicas utilizadas para la resolución de los problemas y una conclusión.

1 | ECUALIZACIÓN LOCAL DE HISTOGRAMA:

1.1 | DESCRIPCIÓN:

En este ejercicio se implementó la ecualización local de histograma sobre una imagen en escala de grises. La ecualización de histograma es una técnica que mejora el contraste de una imagen al redistribuir sus niveles de intensidad de manera más uniforme. En este caso, se aplicó la ecualización de forma local, es decir, se utilizó una ventana deslizante sobre la imagen para realizar la ecualización en bloques pequeños de la imagen, con el fin de mejorar detalles específicos sin afectar globalmente el contenido de la imagen.

Resultado final del ejercicio (ventana 21x17):



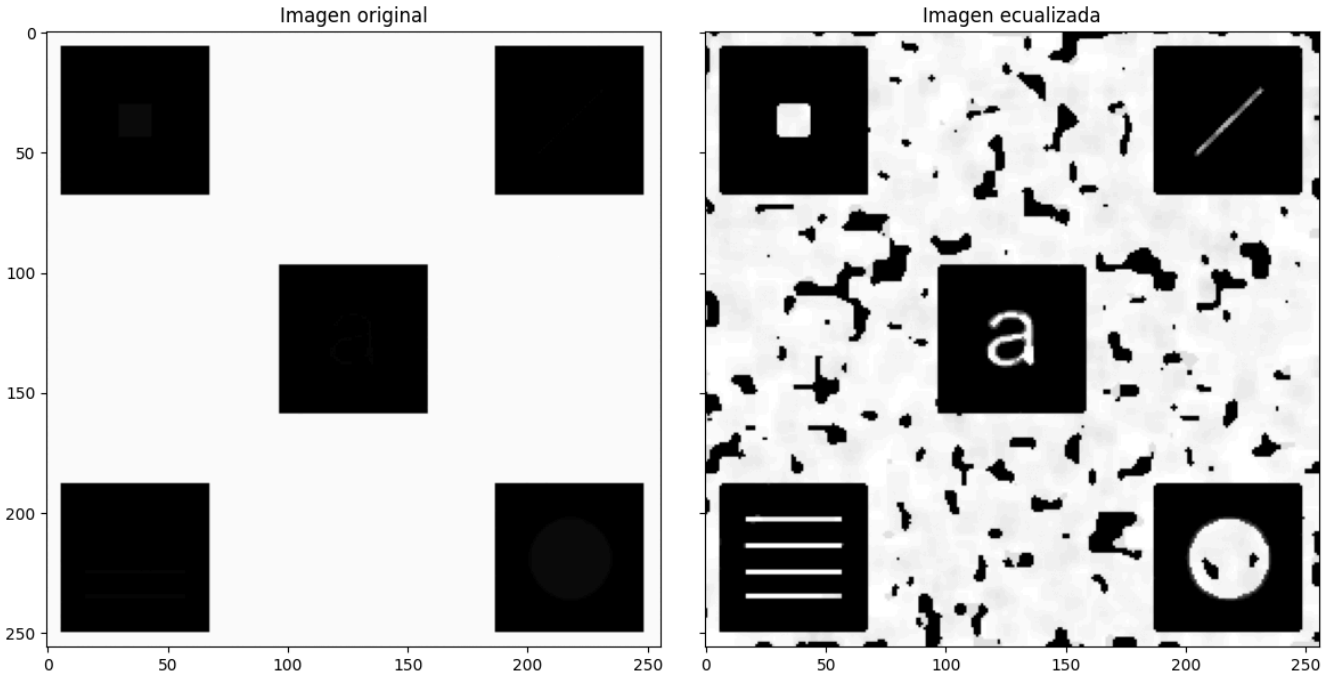
1.2 | PROBLEMAS ENFRENTADOS:

Uno de los principales desafíos en este ejercicio fue manejar el tamaño y el desplazamiento de la ventana de ecualización. A medida que la ventana se mueve sobre la imagen, es necesario gestionar correctamente los bordes de la imagen para evitar distorsiones o artefactos. Para abordar este problema, se utilizó el padding replicado, que extiende los bordes de la imagen de forma que los píxeles adicionales no afecten el proceso de ecualización.

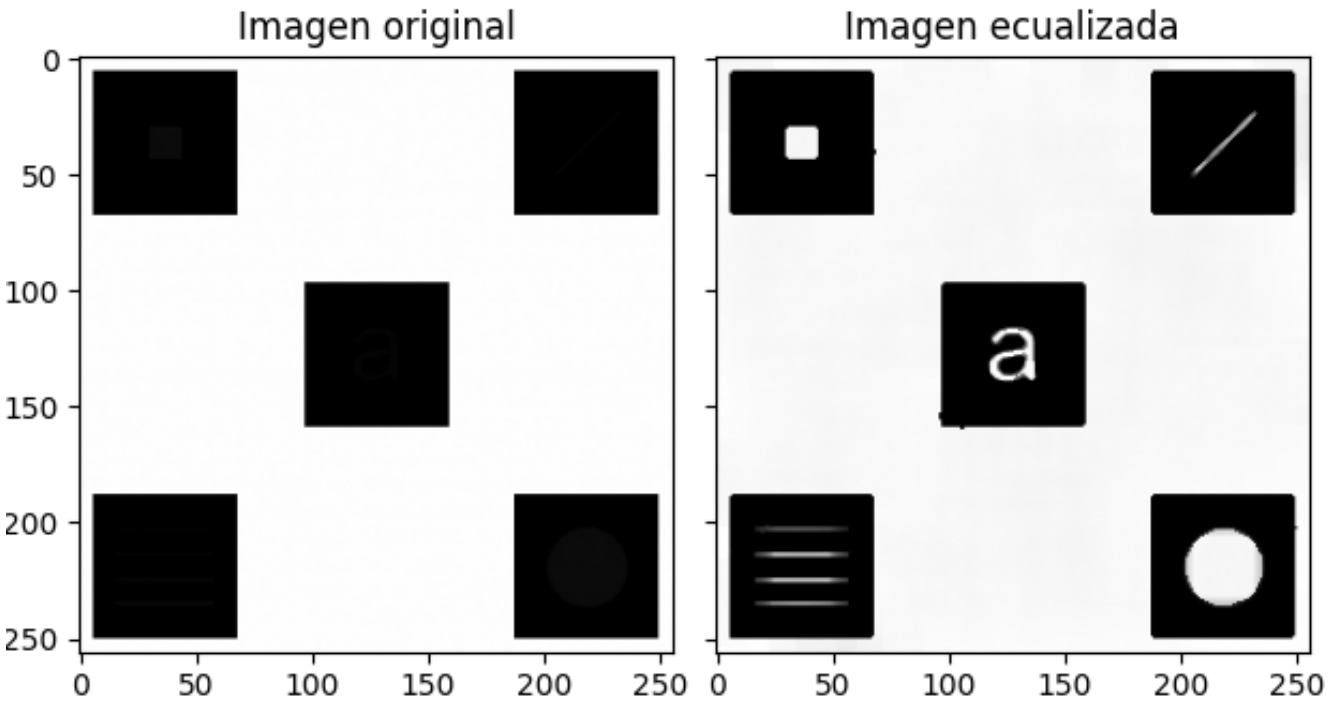
Otro problema fue la elección del tamaño adecuado de la ventana. Si la ventana es demasiado pequeña o es demasiado grande, puede no llegar a los resultados esperados, obteniendo solo alteraciones en la imagen. En este caso, se decidió un tamaño de ventana de 18x18 píxeles, que permitió una ecualización efectiva sin afectar demasiado los detalles finos de la imagen (como fue mostrado en la imagen anterior).

A continuación, se muestran otros resultados de imágenes con diferentes tamaños de ventanas.

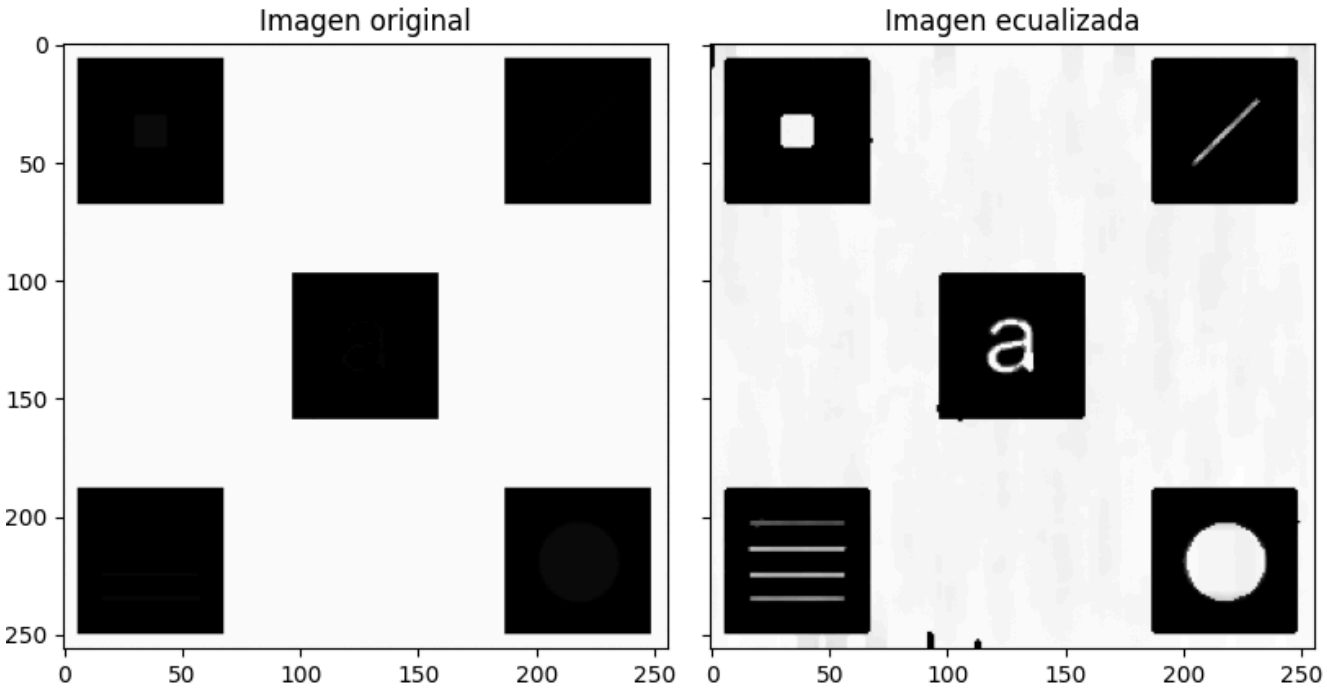
Si es muy pequeña la ventana 9x9, se aleja de la original con distorsiones:



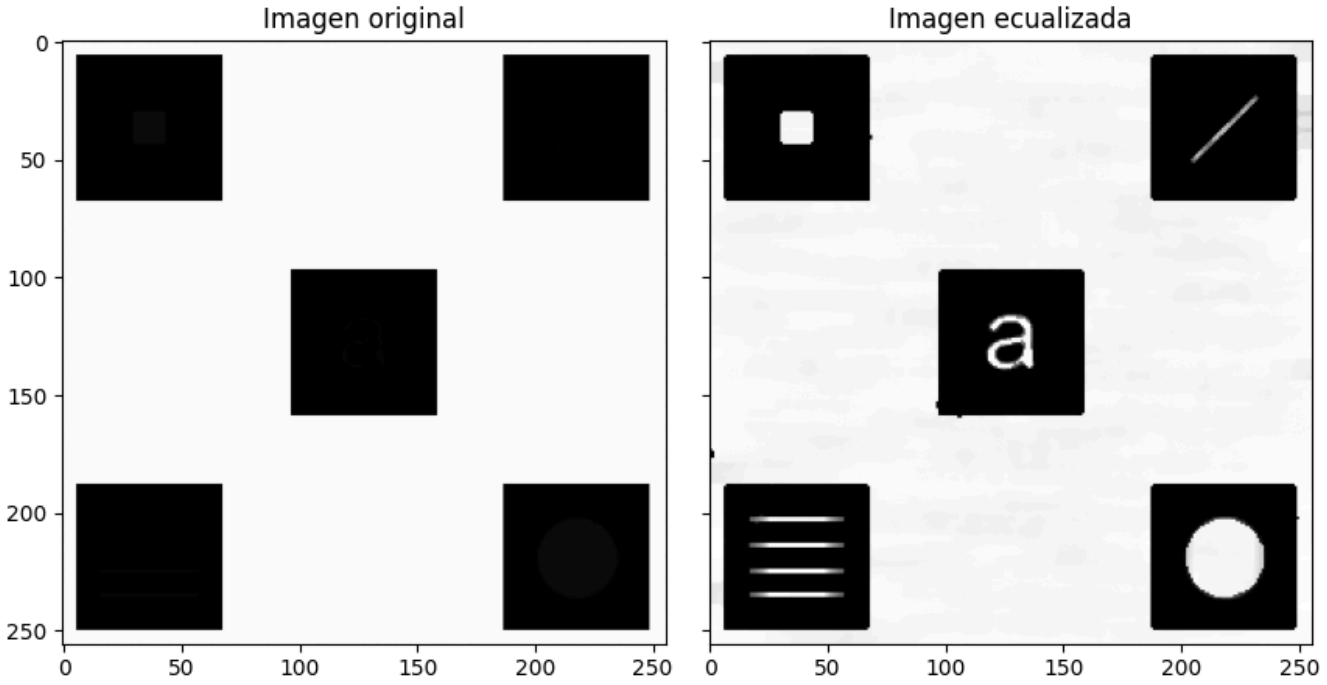
Si es muy grande la ventana 37x37, se producen distorsiones, principalmente en las figuras encontradas:



Ahora con ventanas rectangulares, la ventana 9x37, se producen distorsiones paralelas al lado mayor de la ventana y con algunas presencias de pixeles negros en el fondo blanco:



Mientras que con la ventana rectangular de 37x9, se producen distorsiones paralelas al lado mayor de la ventana otra vez, solo que en esta caso en dirección horizontal y con algunas presencias de pixeles negros en el fondo blanco:



1.3 | TÉCNICAS APLICADAS:

Para resolver los problemas mencionados, se aplicaron las siguientes técnicas:

Padding Replicado: Dado que la ecualización se realiza en bloques, se necesitaba una manera de manejar los bordes de la imagen. Para ello, se aplicó un padding replicado, que agrega bordes duplicados en la imagen, evitando artefactos o distorsiones al aplicar la ecualización.

```
24      # Aplicar padding replicado
25      img_padded = cv2.copyMakeBorder(img, pad_h, pad_h, pad_w, pad_w, borderType=cv2.BORDER_REPLICATE)
```

Desplazamiento de la Ventana: La imagen se recorre en bloques del tamaño de la ventana (21x17), y en cada uno se aplica ecualización de histograma local.

```
30      # Recorrer cada píxel de la imagen original
31      for i in range(height):
32          for j in range(width):
33              # Extraer ventana centrada en (i, j)
34              ventana = img_padded[i:i + window_h, j:j + window_w]
```

Ecualización de Histograma: Utilizada para mejorar el contraste de cada bloque de la imagen. La ecualización de histograma redistribuye los valores de intensidad de los píxeles en un bloque de forma que se optimiza la distribución de los valores.

```
35              # Ecualizar la ventana_
36              ventana_eq = cv2.equalizeHist(ventana)
```

Suavizado con Filtro de Mediana: Después de aplicar la ecualización, se utilizó un filtro de mediana para suavizar la imagen y eliminar posibles artefactos creados durante el proceso de ecualización.

```
54  # Suavizar la imagen para eliminar posibles artefactos
55  eq = cv2.medianBlur(eq_ss, 3)
```

1.4 | CONCLUSIÓN:

La ecualización local de histograma es una técnica eficaz para mejorar el contraste de una imagen, por ejemplo, cuando se trata de detalles específicos o áreas que requieren mayor claridad. El uso de una ventana deslizante y el padding replicado permitieron una correcta aplicación de la técnica sin generar artefactos que cambien mucho a la imagen.

El proceso también permitió experimentar con diferentes tamaños de ventana para encontrar el valor óptimo que balanceara la mejora del contraste. Sin embargo, el desafío principal sigue siendo elegir el tamaño de ventana adecuado para obtener una ecualización efectiva.

La implementación realizada en este ejercicio muestra cómo una técnica de procesamiento de imágenes relativamente simple puede tener un impacto significativo en la visualización de detalles escondidos dentro de una imagen, mejorando sus detalles y contraste general.

2 | CORRECCIÓN DE EXAMEN MULTIPLE CHOICE:

2.1 | DESCRIPCIÓN:

En este ejercicio, que nos pedía evaluar un esquema de una plantilla de respuestas a un examen múltiple choice de 25 preguntas con cinco opciones para cada una de ellas (A, B, C, D y E). Dicha plantilla también cuenta con un encabezado con cuatro campos para completar datos personales (Name, ID, Code y Date). Lo que hicimos fue detectar cada casilla donde están las letras y detectar cual es la que está marcada. Para el encabezado detectamos cada campo donde estaba el nombre, id , código y fecha, para corroborar si la información que proporciona era válida, lo solucionamos a través de componentes conectadas. También por último mostrar una imagen donde se muestre los alumnos aprobados y no aprobados.

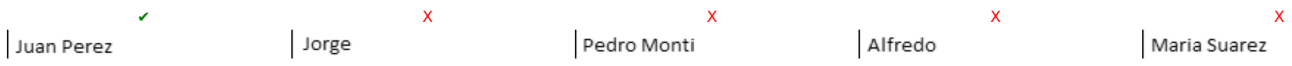
Resultado del ejercicio

```
===== EXAMEN 2 =====

----- RESULTADOS DE LAS RESPUESTAS -----
Pregunta 1: OK
Pregunta 2: OK
Pregunta 3: OK
Pregunta 4: MAL
Pregunta 5: OK
Pregunta 6: MAL
Pregunta 7: MAL
Pregunta 8: OK
Pregunta 9: MAL
Pregunta 10: OK
Pregunta 11: MAL
Pregunta 12: OK
Pregunta 13: OK
Pregunta 14: MAL
Pregunta 15: MAL
Pregunta 16: OK
Pregunta 17: MAL
Pregunta 18: OK
Pregunta 19: MAL
Pregunta 20: MAL
Pregunta 21: MAL
Pregunta 22: MAL
Pregunta 23: MAL
Pregunta 24: MAL
Pregunta 25: MAL

----- RESUMEN -----
Total opciones: 125
Total preguntas: 25
Respuestas detectadas: 25 / 25
Respuestas correctas: 10 / 25
Respuestas sin marcar: 0 / 25
NO APROBADO

----- CONTROL DE DATOS COMPLETADOS -----
Encabezado
name: MAL
id: MAL
code: OK
date: OK
```



2.2 | PROBLEMAS ENFRENTADOS:

Uno de los desafíos que nos enfrentamos al hacer el ejercicio fue encontrar los parámetros de la función ‘houghcircles’, para encontrar todos los círculos donde están las letras, ya que si definimos un radio bajo o un radio alto nos detectaba círculos que no eran.

También para encontrar los campos del encabezado nos surgieron bastantes problemas, como encontrar el umbral correcto para detectar los bordes de los campos, porque nos tomaba las letras también, entonces con prueba y error lo pudimos solucionar. Con el tema de detectar los objetos, utilizamos las componentes conectadas, lo difícil fue hacer que detecte solo los caracteres como objeto y no por ejemplo un borde de la imagen o algún punto o acento, etc. Lo pudimos solucionar filtrando por componentes chicos con las ‘stats’ de los componentes conectados.

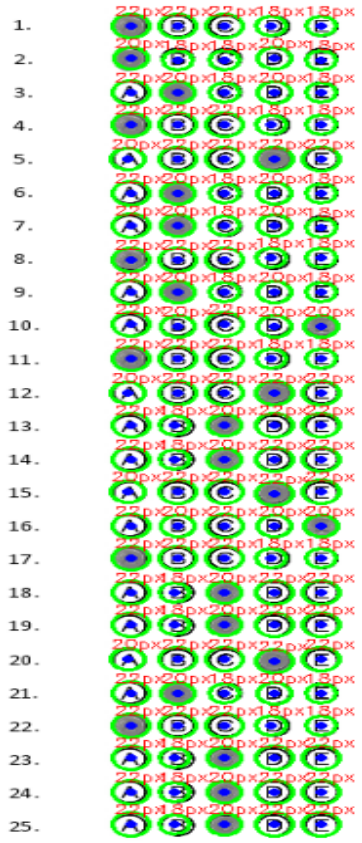
Por último y no el desafío menos importante fue armar todo el código, armando diferentes funciones para la reutilización de código, de manera que sea representable y para poder mostrar un resultado final con una sola función.

2.3 | TÉCNICAS APLICADAS:

Para resolver los problemas mencionados, se aplicaron las siguientes técnicas:

Detección de círculos usando la transformada de Hough: El ejercicio nos pedía detectar en cada pregunta cual era la opción marcada, o si no había marcada. Por lo tanto, nosotros decidimos usar la función HoughCircles(), para obtener todas las opciones y así decidir si estaba marcada o no.

```
circles = cv2.HoughCircles(
    img,
    method=cv2.HOUGH_GRADIENT,
    dp=1,
    minDist=10,
    param1=50,
    param2=15,
    minRadius=9,
    maxRadius=14
)
```



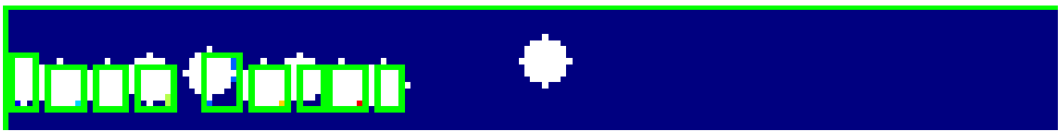
Detección de renglones y separaciones del encabezado: Nos pedía obtener los campos del encabezado, lo que utilizamos para detectar los renglones y la separación de los campos fue usar la técnica basada en detección de cambios en la intensidad de los píxeles. Primero, convertimos la imagen en una matriz booleana que marca con true los píxeles negros . Luego, analizamos fila por fila si hay presencia de renglones. Esto genera una señal del tipo "pulso", donde los valores true indican la presencia de un renglón.

A partir de esta señal, detectamos los cambios de estado (inicio y fin de cada renglón), ya que a nosotros nos interesa la línea más ‘alta’ (encabezado), la recortamos de imagen y guardamos. Finalmente, para detectar las separaciones de los campos, elegimos las columnas del encabezado que tienen mayor cantidad de negro y agrupamos las mas cercanas(ya que una línea puede ocupar más de una columna). Después tomamos una columna representativa por grupo (usando la mediana) y guardamos el resultado que es una lista de coordenadas donde se encuentran las líneas separadoras.

Componentes conectadas: Para detectar los caracteres de cada campo utilizamos la función connectedComponentsWithStats. De stats utilizamos el área para filtrar por los componentes más chicos, el centroide lo usamos para detectar si había más de una palabra, calculando la distancia entre centroides, si había más de una cierta distancia lo tomábamos como espacio.

```
# Detecta componentes conectados
num_labels, labels, stats, centroids = cv2.connectedComponentsWithStats(recorte_bin, 8, cv2.CV_32S)
```

Componentes conectados - name



2.4 | CONCLUSIÓN:

Este trabajo nos permitió conocer y adentrarnos en distintas técnicas de procesamiento de imágenes para resolver un trabajo de automatización de corrección de exámenes. Pudimos obtener un buen resultado en la detección de opciones marcadas, también de los campos del encabezado, pudiendo detectar los caracteres de cada una para poder validar los datos, mediante diferentes técnicas utilizadas a lo largo del trabajo como la transformadas de Hough, detección de líneas y componentes conectadas. A pesar de los desafíos enfrentados llegamos a obtener un buen resultado final, que no solo automatiza la corrección, sino que también te permite ver a través de una imagen los nombres de las personas aprobadas y no aprobadas, como también validar los datos del encabezado,.