Resumen del Código Ecualización Local

El código implementa un algoritmo de **ecualización de histograma local** para mejorar el contraste de una imagen en escala de grises. El objetivo principal es revelar detalles ocultos en áreas de la imagen que tienen un contraste muy bajo (en este caso, figuras dentro de los cuadrados negros). Para lograrlo, en lugar de analizar la imagen de forma global, el programa recorre cada píxel, analiza una pequeña región o "bloque" (con un tamaño que definimos al iniciar la función a su alrededor, ecualiza el histograma de esa región y asigna el nuevo valor al píxel central en una nueva imagen de salida.

Técnicas Utilizadas

1. **Ecualización de Histograma Local:** A diferencia de la ecualización de histograma global, que utiliza una sola transformación para toda la imagen, la versión local calcula una transformación de contraste distinta para cada píxel basándose en la distribución de brillo de su vecindad.
   Esto se aplica recorriendo la imagen original, aplicando la transformación sobre una ventana del tamaño que elegimos, y pegando la nueva sub-imagen transformada en una imagen de salida.

2. **Procesamiento por Bloques (Ventana):** El código utiliza un enfoque de "ventana deslizante". Se define un bloque de un tamaño específico (m x n) que se centra en cada píxel de la imagen original para analizarlo.

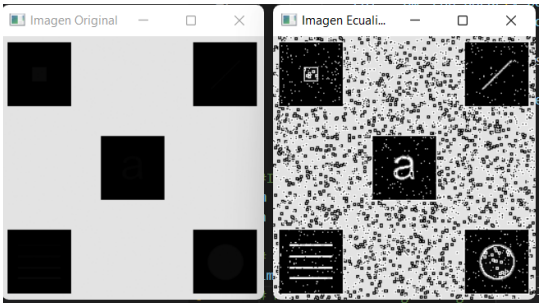
3. **Manejo de Bordes (cv2.BORDER_REPLICATE):** Para poder procesar los píxeles que se encuentran en los bordes de la imagen (donde el bloque se saldría de los límites), se añade un borde artificial a la imagen. La técnica BORDER_REPLICATE simplemente copia los píxeles del borde, lo cual es un método eficaz para evitar artefactos no deseados en los límites de la imagen procesada.

4. **Uso de la función cv2.equalizeHist:** Aunque esta función está diseñada para una ecualización global, el código la aplica sobre cada pequeño bloque (ROI) extraído de la imagen, logrando así un efecto local.

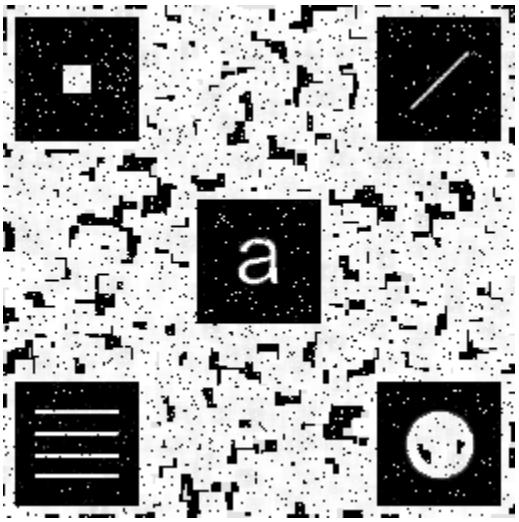
Problemas Afrontados

En primer lugar, unos de los principales problemas era encontrar el “Tamaño justo” que debía tener la ventana, para que esta tenga la “mejor” calidad posible.
Definimos mejor entre comillas, ya que la misma es subjetiva según cada usuario.

Empezamos probando con una ventana de **3x3**, pero veíamos que la imagen seguía cargada de mucho Ruido, e inclusive algunos objetos no se distinguían de la mejor manera

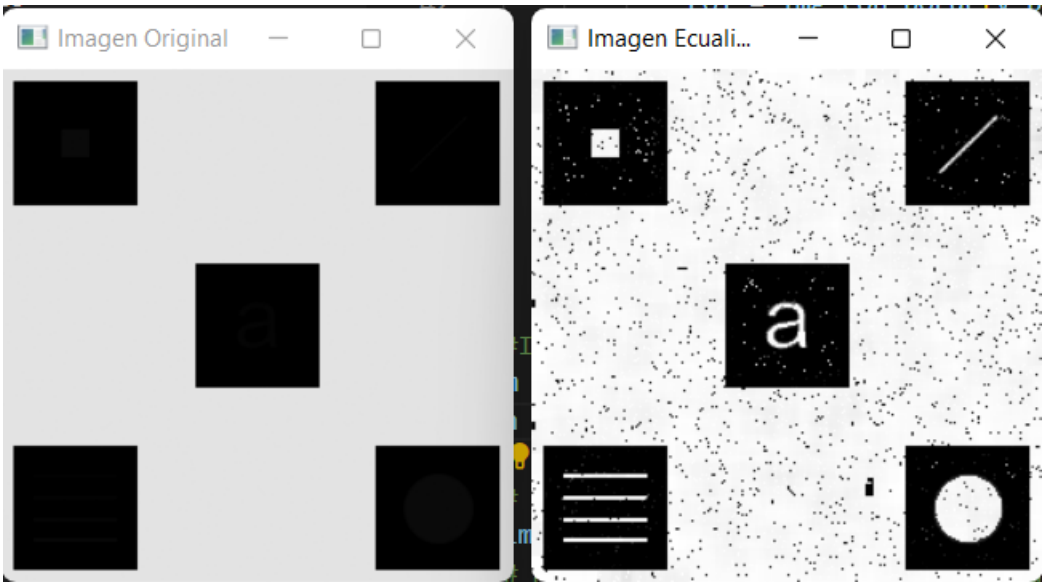


Luego pasamos a una ventana de **9x9**, en la cual si bien el ruido disminuía, aún teníamos posibilidad de mejora
Acá encontramos un patrón, entendimos que a medida que íbamos aumentando el tamaño de la ventana, se podía ir mejorando la distinción de la misma

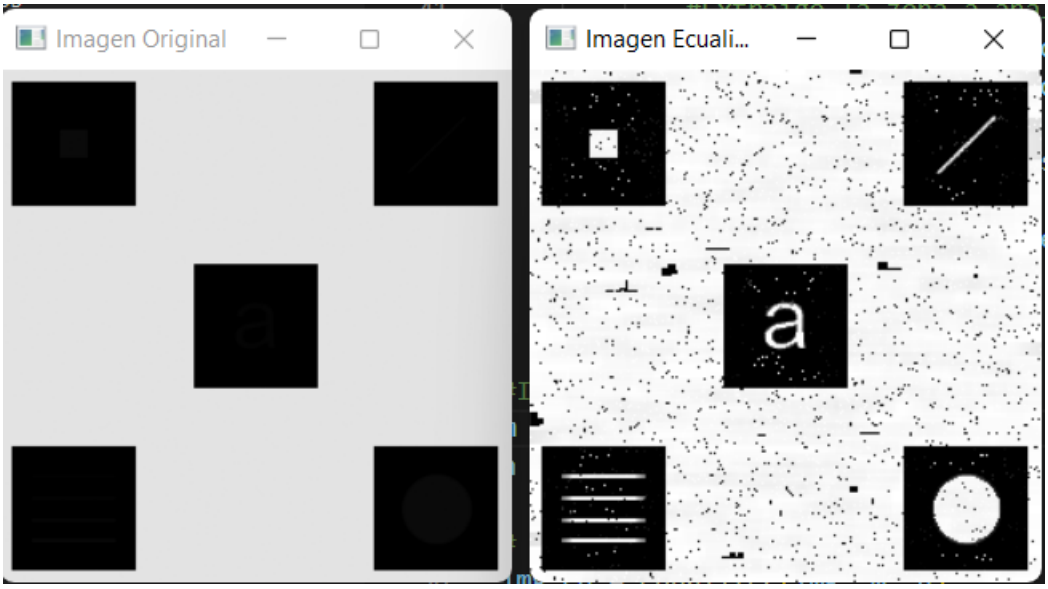


Pasamos a una ventana de **16x16**

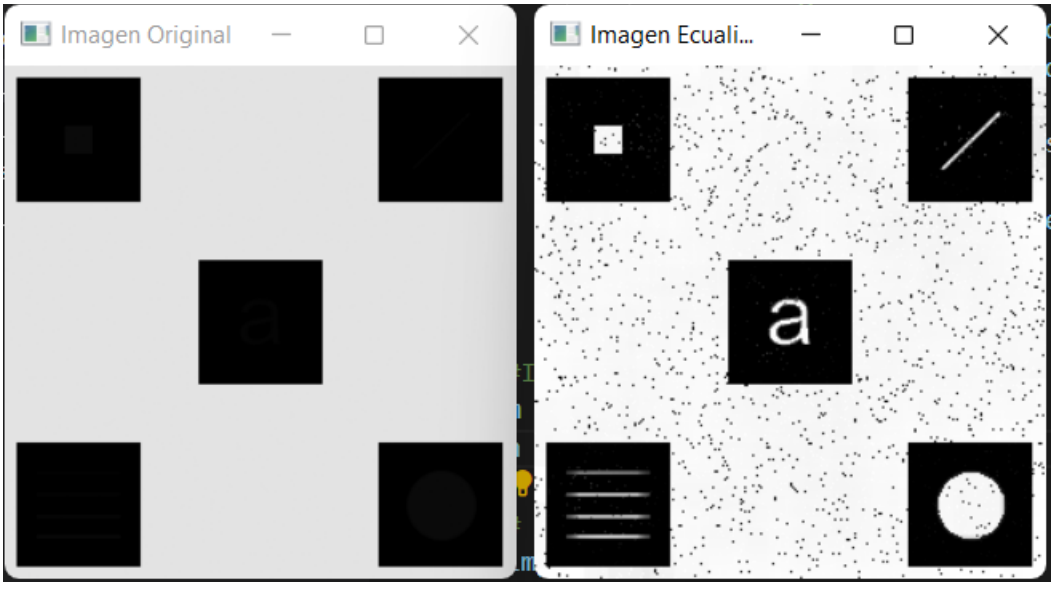
En la cual vimos que podía ser la definitiva, dado que las imágenes dentro de los cuadros negros se resaltaban correctamente gracias al contraste aplicado. Y el “ruido” del fondo, no era tan pesado como con ventanas anteriores



Intentamos con ventanas Rectangulares, ejemplo **7 X 30**
Con esto validamos que el código soportara distintos tamaños y no únicamente formas cuadradas, y además validamos que la solución presentada era bastante buena.

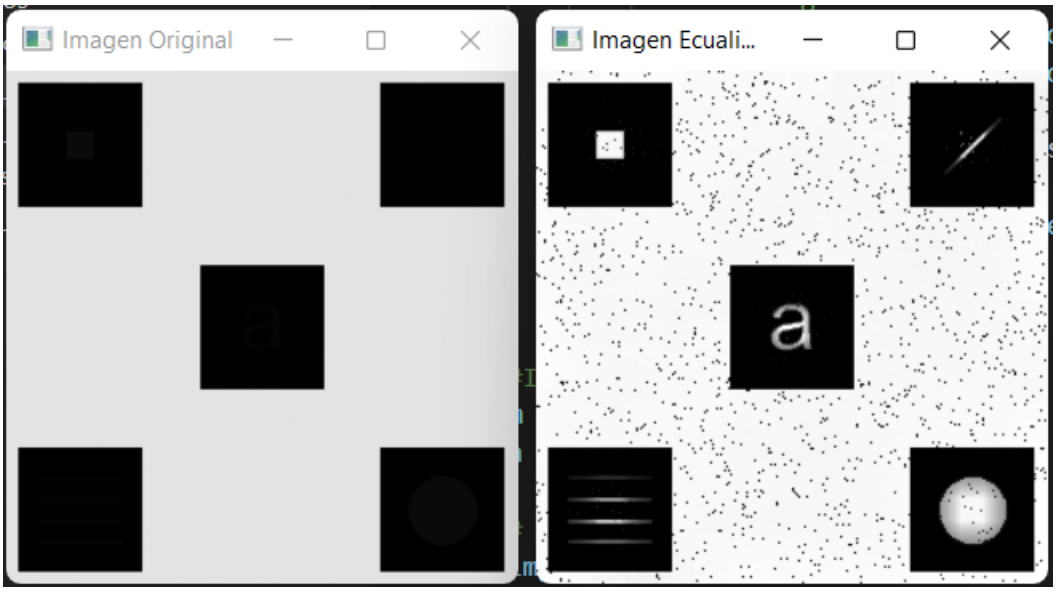


32X32
Esta solución vemos que es la mas cercana a lo que buscamos, diferenciando bien cada objeto, a pesar de que algunos de ellos puedan tener una pequeña difuminación en sus terminaciones



50X50

A partir del tamaño 50x50 de Ventana en adelante, comenzamos a distinguir que los objetos perdían calidad, pareciendo difuminados en lugar de contrastados al 100%, ya que esto suaviazaba los saltos entre cada cambio de valores.



EN RESUMEN

El ajuste de ventana se puede ir mejorando poco a poco, cambiando su tamaño, pero este tiene un techo global en el cual luego, entre mas grande sea el tamaño de ventana, mas calidad va a perder la misma, difuminando su imagen y suavizando los saltos. Es decir, si lo viéramos en forma de Curva, esta tendría forma de campana, en donde al aumentar el tamaño de ventana, mejoramos la definición hasta llegar a su techo, en el cual luego la empeoramos poco a poco.

Esto ocurre, dado que al aumentar el tamaño del bloque, el resultado de la Equalización Local comienza a parecerse al resultado de la Ecualización Global, perdiendo asi calidad en el mismo

En síntesis, nuestra ventana Optima consideramos que es 32x32, entregando una imagen con distinción clara entre los objetos.