Instituto Tecnológico de Costa Rica
Área Académica de Ingeniería en Computadores
Procesamiento y Análisis de Imágenes Digitales
Tarea # 2

Edgar Chaves González - Ki Sung Lim - Christian Alpizar Monge
2017239281 - 2017098352 - 2017146794
edjchg@gmail.com - kisunglim@estudiantec.cr - c99alpizar@estudiantec.cr

Mayo 2021

1. *Fast Median Filter Approximation* (FMFA)

Este método es una optimización del algoritmo del filtro de la mediana, en el cual se toman columnas de un *kernel* o ventana de vecinos. La optimización consiste en que los resultados son acumulativos, de manera tal que cada vez lo único que se debe calcular es el ordenamiento de una columna, puesto que las otras dos columnas, fueron calculadas en la iteración anterior[1].

Para entender mejor este algoritmo se tiene el pseudocódigo de la siguiente sección.

1.1. Pseudocódigo

Algorithm 1: Fast Median Filter Approximation.

```
Result: Imagen con ruido reducido
imagen_con_ruido = I
m = filas_imagen
n = columnas_imagen
imagen_salida = matriz_ceros(m,n)
for 2 ≤ i < m-1 do
    A = ordenar_ascendente(I[i-1,1], I[i,1], I[i+1,1])
    B = ordenar_ascendente(I[i-1,2], I[i,2], I[i+1,2])
    for 3 ≤ j < n-1 do
        píxel_medio_A = A[2]
        píxel_medio_B = B[2]
        C = ordenar_ascendente(I[i-1,j], I[i,j], I[i+1,j])
        píxel_medio_C = C[2]
        vector_medianas = ordenar_ascendente([píxel_medio_A, píxel_medio_B, píxel_medio_C])
        nuevo_píxel = vector_medianas[2]
        imagen_salida[i,j] = nuevo_píxel
        A = B
        B = C
    end
end
```

De este pseudocódigo, se puede observar que por cada fila, se calcula primero las dos primeras columnas, después estos valores van siendo actualizados, pero a lo largo del recorrido de esa fila, el único valor que se recalcula es **C**.

El algoritmo en su primera iteración se puede interpretar de la siguiente forma:

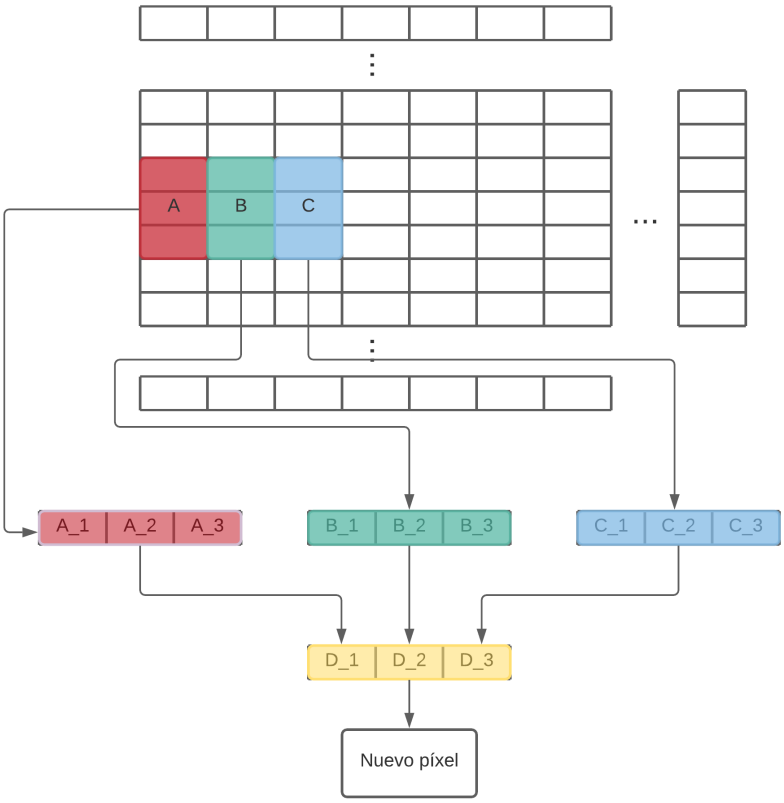

Figura 1: Algoritmo *Fast Median Filter Approximation* en su primera iteración.

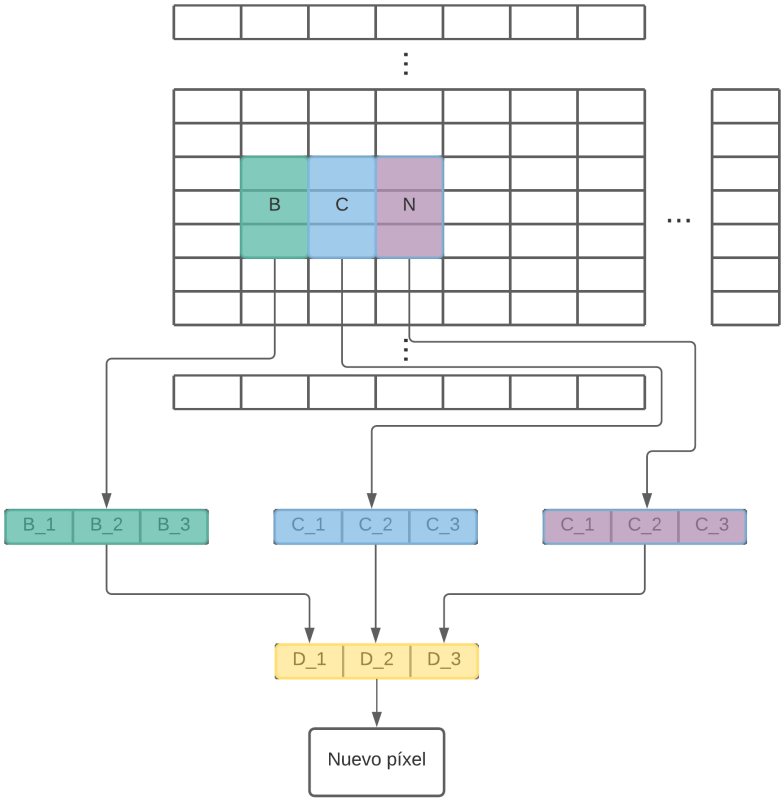

Figura 2: Algoritmo *Fast Median Filter Approximation* en su j-ésima iteración.

Se puede notar que las columnas calculadas en la figura 4 se mantienen y aprovechan en la figura 2, de manera que para esta iteración, basta con calcular una columna únicamente, pues las demás columnas ya se calcularon en la iteración anterior.

2. *Improved approximated median filtering algorithms* (IAMFA-I)

La optimización de este algoritmo consiste en seguir aprovechando el cálculo de columnas, como ocurre en el algoritmo anterior, pero haciendo la selección del nuevo píxel de manera tal que disminuye la probabilidad de elegir un píxel que igualmente este con ruido.

2.1. Pseudocódigo

Algorithm 2: IAMFA-I.

```
Result: Imagen con ruido reducido
imagen_con_ruido = I
m = filas_imagen
n = columnas_imagen
imagen_salida = matriz_ceros(m,n)
for 2 ≤ i < m-1 do
    A = ordenar_ascendente([I[i-1,1], I[i,1], I[i+1,1]])
    A = mid_value_decision_median(A)
    B = ordenar_ascendente([I[i-1,2], I[i,2], I[i+1,2]])
    B = mid_value_decision_median(B)
    for 3 ≤ j < n-1 do
        C = ordenar_ascendente([I[i-1,j], I[i,j], I[i+1,j]])
        C = mid_value_decision_median(C)
        nuevo_píxel = mid_value_decision_median([A, B, C])
        imagen_salida[i,j] = nuevo_píxel
        A = B
        B = C
    end
end
```

Del algoritmo anterior se puede notar la función que permite obtener un píxel con menos probabilidad de que esté corrupto. Esta función es *mid_value_decision_median*, la cual consiste en tomar una columna o arreglo de valores y evaluarlos de la siguiente forma:

$$mid\_value\_decision\_median(P_1,P_2,P_3) = \begin{cases} P_1 & \textbf{si } P_2 = 255 \\ P_2 & \textbf{si } 0 < P_2 < 255 \\ P_3 & \textbf{si } P_2 = 0 \end{cases}$$

Por lo tanto, la decisión del nuevo píxel dependerá enteramente del valor de P_2.

3. Resultados

Figura 3: Imagen en escala de grises de la *Monalisa* con ruido de sal y pimienta.

Figura 4: Imagen de tres canales con ruido de sal y pimienta.

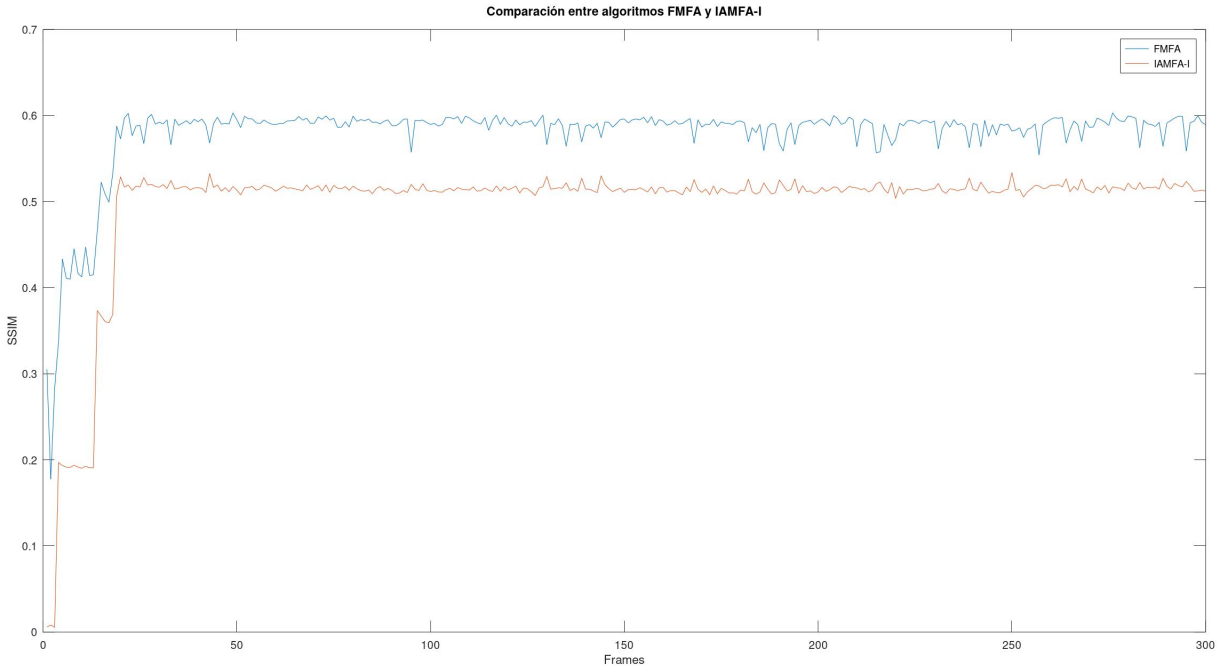

Figura 5: Resultado de la comparación entre el video procesado con FMA y IAM.

El resultado 5 es curioso, porque está diciendo que el algoritmo IAM es peor que el algoritmo FMA. En general esto no sería cierto, sin embargo, cuando hay un fondo completamente negro, o blanco el algoritmo IAM no es tan bueno, y por eso es que se ha obtenido este resultado. El *frame* número 20 fue analizado, encontrando lo siguiente:

Figura 6: Imágenes correspondientes al resultado 5

De la imagen 3 se tiene que realemente el FMA hizo un mejor trabajo que el algoritmo IAMFA-I. Pero si se observa con cuidado la imagen de IAMFA-I los únicos puntos dañados aún son los que se encuentran en la parte del fondo completamente negro, haciendo que el SSIM entre la imagen procesada con FMFA sea más limpia que la imagen analizada con IAMFA.

Siguiendo con otras imágenes, sin fondos totalmente blancos ni negros se obtiene el siguiente resultado:

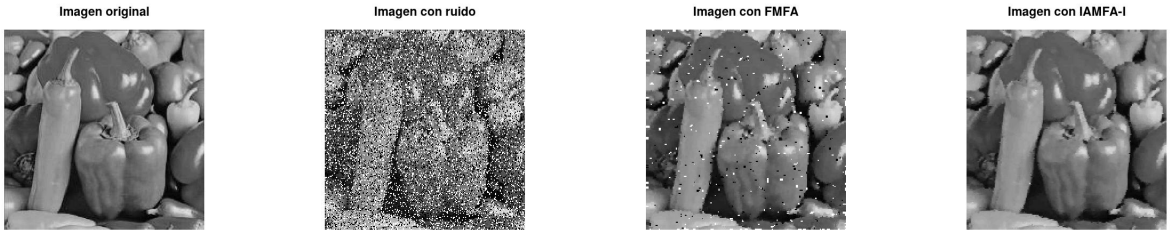

Figura 7: Muestra de los algoritmos sobre la imagen peppers.jpg

Para estas imágenes, los resultados SSIM fueron los siguientes:

- Entre la imagen procesada con FMFA y el original fue de: 0.6797.
- Entre la imagen procesada con IAMFA y el original fue de: 0.9202.

Con los resultados anteriores clarifican que el algoritmo IAMFA es mejor que el FMFA. No solo mediante los resultados anteriores se puede determinar, sino que visualmente es evidente esta mejora.

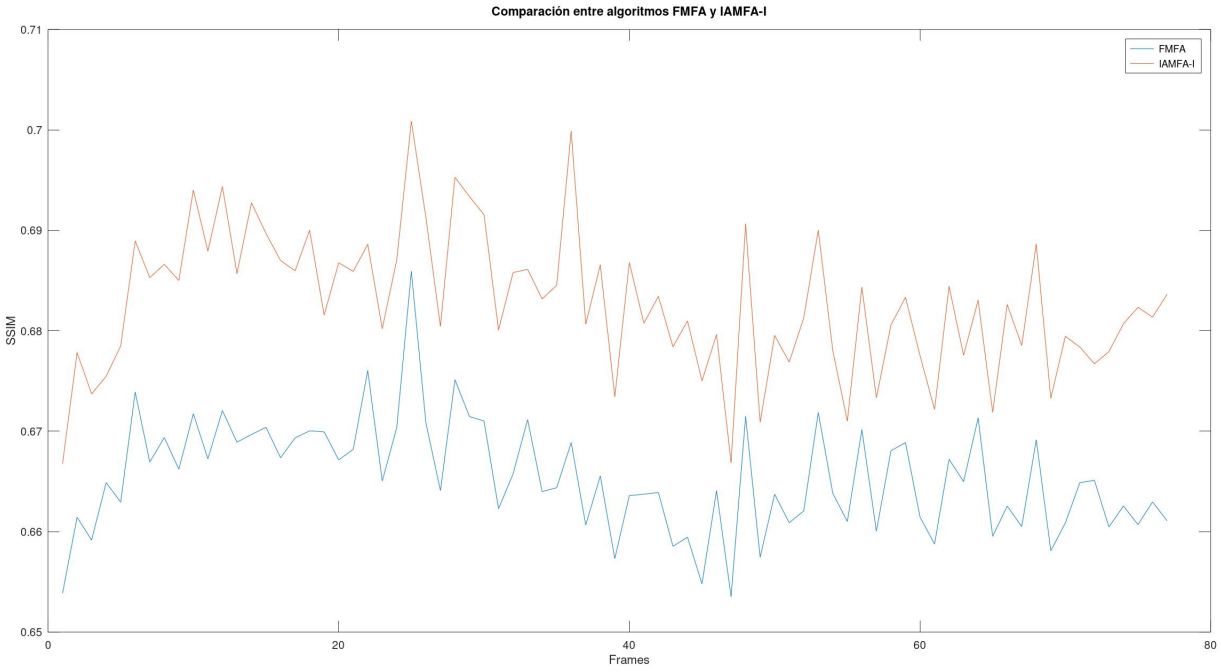

Figura 8: Resultado de un vídeo versión 2 de prueba.

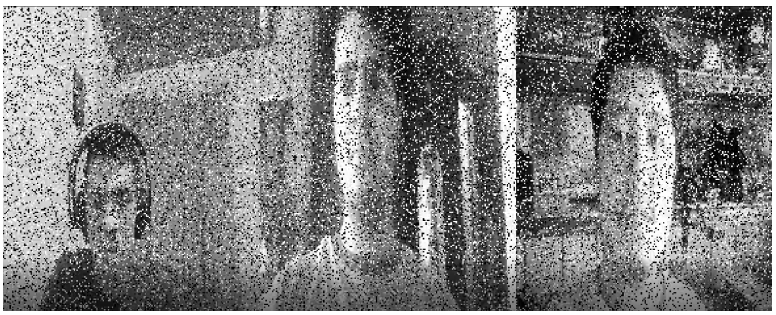

Figura 9: Vídeo versión 2 con ruido.

Figura 10: Vídeo versión 2 procesado con el algoritmo FMFA.

Figura 11: Vídeo versión 2 procesado con el algoritmo IAMFA-I.

De la figura 8 que para el vídeo versión 2 que se tomó el resultado ahora indica que para este conjunto de *frames* el algoritmo IAMFA se comportó mejor en general que el algoritmo FMFA. Como muestra se tienen las figuras 10 y 11.

Referencias

[1] O. Appiah, M. Asante y J. B. Hayfron-Acquah. "Improved approximated median filter algorithm for real-time computer vision applications". En: *Journal of King Saud University-Computer and Information Sciences* (2020).