Tecnológico de Monterrey, Campus Guadalajara

Procesamiento y Análisis de Imágenes Médicas

BI3008

Nombre completo: Abel Aguilar Chávez
Matricula: A01112847
No de práctica: 3
Fecha de entrega: 20/02/2018

Resumen de la práctica:

Teniendo una imagen patrón sobre la cual se aplica distintos tipos de ruidos; gaussiano, sal y pimienta, uniforme, se espera recuperar lo mejor posible la imagen original con una distintos filtros pasa bajos; media, media ponderada, mediana, suavizado conservador, al igual que observar sus diferencias con los filtros pasa altos; Laplaciano y sustracción de media. Y por último hacer una introducción a filtros direccionales; direccionales, Sobel, Canny, Prewitt, y ver la capacidad que tiene para captar los bordes de sus direcciones respectivas.

Introducción:

La forma en que se ilustra la forma de frecuencias en un mundo bidimensional en razón a su transformada de fourier. Una representación de frecuencias bajas es la varianza lenta de intensidades en una imagen, mientras que en el caso de un cambio transitorio marcados como el borde de una columna respecto al fondo, o bien la presencia de ruido.

Tomando estas definiciones en cuenta el caso donde se realice un filtrado de bajas se esperaría que se generé un efecto de blurring en la imagen resultante, mientras que en un filtro pasa altas se espera que los cambios rápidos de intensidad se vean realzados. Distinto de un filtro en una dimensión, se necesita realizar junto con un parchado, padding, para que el resultado final por ejemplo en un filtrado pasa bajos sea uniforme.

La razón común por lo que se realiza un filtrado es por la aparición de ruido, donde los más comunes son los ruidos probabilisticos de densidad; entre ellos se encuentra el ruido impulsivo, o de sal y pimienta, ruido uniforme, y gaussiano. (Rafael, 2008)

Objetivos:

- *General*:
    - Realizar filtrado a distintas imágenes y observar los resultados.
- *Particulares:*
    - Agregar ruido a una imagen y observar lo que genera.

- Realizar filtros pasa bajos a imágenes con ruido, comparar sus resultados.
- Utilizar filtros pasa altos en una imagen sin ruido, compara sus resultados.
- Comparar los resultados de diferentes tipos de filtros direccionales.
- Realizar un filtro de suavizado conservador sobre una imagen con ruido de sal y pimienta, y observar los resultados.

Metodología – Resultados- Análisis

1. Ruido en una imagen

Para la primer parte de la práctica se realizaron diferentes tipos de ruido sobre una imagen para poder observar el tipo de deformación que generan sobre esta. Para esto se realizaron 3 tipos de ruidos; Gaussiano, figura 1, sal y pimienta, figura 2, y uniforme multiplicativo, figura 3, sobre imágenes en espacio RGB al igual que sobre la imagen en escala de grises. Para realizar la adición se utilizó el comando imnoise de Matlab. Donde la ecuación que lo expresa matemáticamente es:

eq. 1. $$f_{ruido}(x,\ y) = f(x,\ y) + r(x,\ y)$$

Donde r(x, y) representa algún tipo de ruido, y f(x, y) es la imagen original.

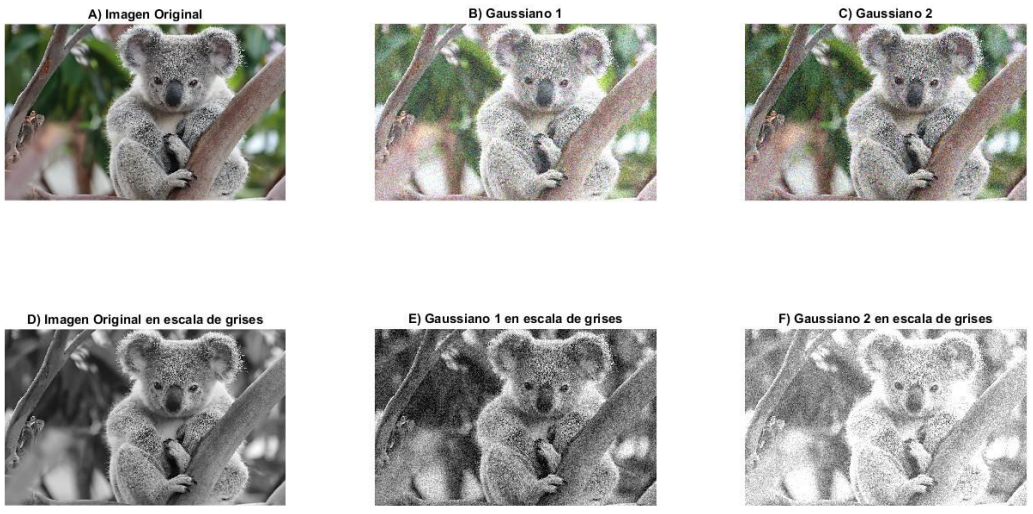

Fig. 1. A) Imagen original, B) Imagen original tras una adición de ruido gaussiano, C) Imagen original tras una adición de ruido gaussiano, D) Imagen original en escala de grises, E) Imagen en escala de grises con ruido gaussiano, F) Imagen en escala de grises con ruido gaussiano.

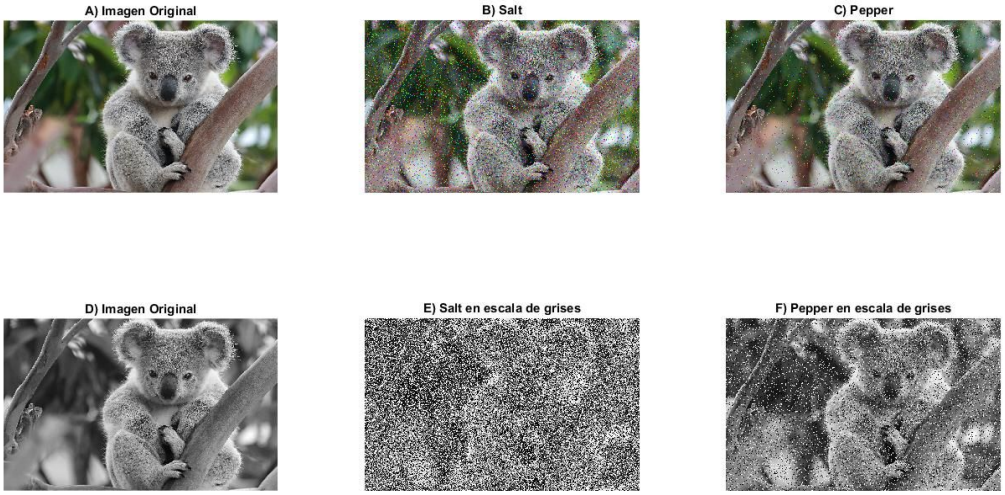

Fig. 2. A) Imagen Original, B) Imagen con ruido de sal, C) Imagen con ruido de pimienta, D) Imagen original en escala de grises, E) Imagen en escala de grises con ruido de sal, F) Imagen en escala de grises con ruido de pimienta.

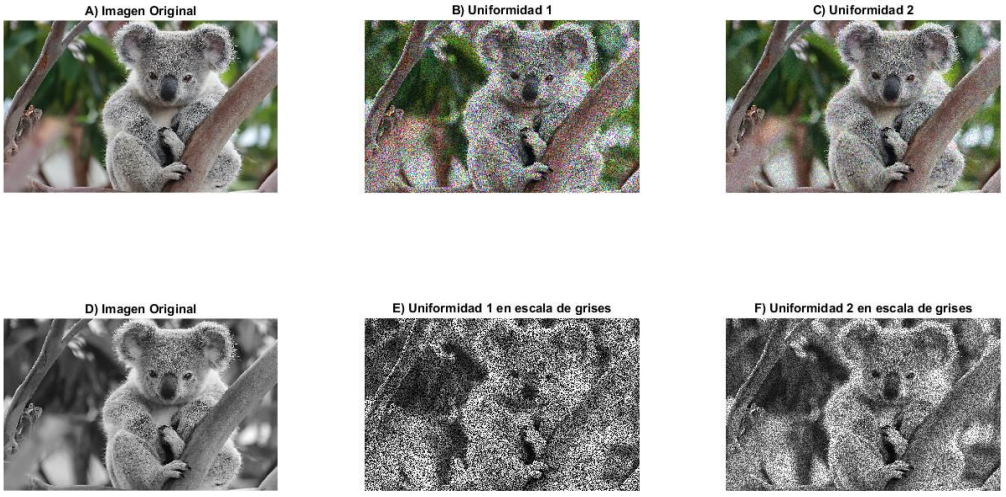

Fig. 3. A) Imagen Original, B) Imagen con ruido uniforme multiplicativo, C) Imagen con ruido uniforme multiplicativo, D) Imagen original en escala de grises, E) Imagen en escala de grises con ruido uniforme multiplicativo, F) Imagen en escala de grises con ruido uniforme multiplicativo.

Algoritmo: Adición de ruido

Escribir 'Leer imagen'
I = Leer Imagen como RGB
IG = Leer Imagen como escala de grises

Escribir 'Ruido Gaussiano'
I_Gaussian1 = I + Gauss con 0.2 de varianza
I_Gaussian2 = I + Gauss con 0.09 de varianza

IG_Gaussian1 = IG + Gauss con 0.01 de varianza
IG_Gaussian2 = IG + Gauss con 0.32 de varianza

Escribir 'Ruido Sal y Pimienta'
I_Salt = I + Salt con densidad de 0.08
I_Pepper = I + Pepper con densidad de 0.02

IG_Salt = IG + Salt con densidad de 0.08
IG_Pepper = GI + Pepper con densidad de 0.02

Escribir 'Ruido uniforme multiplicativo'
I_Uniforme1= I + Speckle con 0.18 de varianza
I_Uniforme2= I + Speckle con 0.02 de varianza

IG_Uniforme1= IG + Speckle con 0.58 de varianza
IG_Uniforme2= IG + Speckle con 0.14 de varianza

Fin del algoritmo

Análisis de la imagen:
La aparición del ruido se puede apreciar como intensidad de color en ciertos pixeles donde no se esperaría ver cierto, o bien, en el caso de la escala de grises con una tonalidad muy blanca, que pueden llevar a la imagen a ser altamente deformada e irreconocible como se puede observar en la figura 2-E. Pero cada uno los tipos de ruidos presentados tienen ciertas características.

Empezando por la figura 1, donde fue aplicado un ruido gaussiano, lo que se puede observar es que en distintas varianzas la imagen a parte de empezar a tener ciertos puntos blancos, también hay un incremento en la intensidad de todos los pixeles que conforman la imagen original; más apreciable en la figura 2-B y 2-F, esto se debe a que el ancho sobre el cual trabaja este tipo de ruido es todo el espectro de pixeles que

forman parte de la imagen donde cada uno recibe un incremento proporcional distinto dependiendo de la varianza, o la media, de la curva de Gauss.

Distinto a lo sucedido en la figura 2, donde fue agregado ruido sal y pimienta. El resultado fue la aparición de puntos de color no esperados en la imagen, debido algún tipo de saturación en alguno de los canales de color de la imagen, más apreciable en la figura 2-B, pero al igual que puede haber saturación a blanco también se generan saturaciones complementarias donde se aproxima mayormente a un negro, figura 2-C.

Y por último el ruido uniforme aunque su histograma es similar al de una curva de Gauss, o ruido gaussiano, no son lo mismo ya que el ancho de de trabajo de este tipo de ruido se encuentra acotado por un factor aleatorio de media 0. Por lo que sí se generan cambios de color visibles pero no existe una adición de iluminación en la imagen final.

2. Filtros pasa bajas

Posteriormente a las imágenes con ruido se le realizaron una serie de filtros; filtro de media, media ponderada; para los anteriormente mencionados se utilizó el comando imfilter de Matlab, y el filtro de mediana, para este se utilizó el comando medfilt2 de Matlab, y observar si se recuperaba la imagen original. Primero se filtró la imagen con ruido gaussiano, figura 4, luego con ruido sal y pimienta, figuras 5 y 6, y por último la imagen con ruido uniforme multiplicativo, figura 7. Donde su representación matemática se presenta a continuación:

$$f_{filter}(x,\ y) = \sum_{s=-b}^{b} \sum_{t=-a}^{a} k(s,\ t) f(x+s,\ y+t)$$

eq. 2.

Donde k(s, t) es la máscara del filtro, y f(x, y) es la imagen original

Fig. 4. A) Imagen original con ruido gaussiano, B) Imagen filtro de media, C) Imagen filtrada con filtro de media ponderada, D) Imagen con filtro de mediana.

Fig. 5. A) Imagen con ruido de sal, B) Filtro de media, C) Filtro de media ponderada, D) Filtro de mediana.

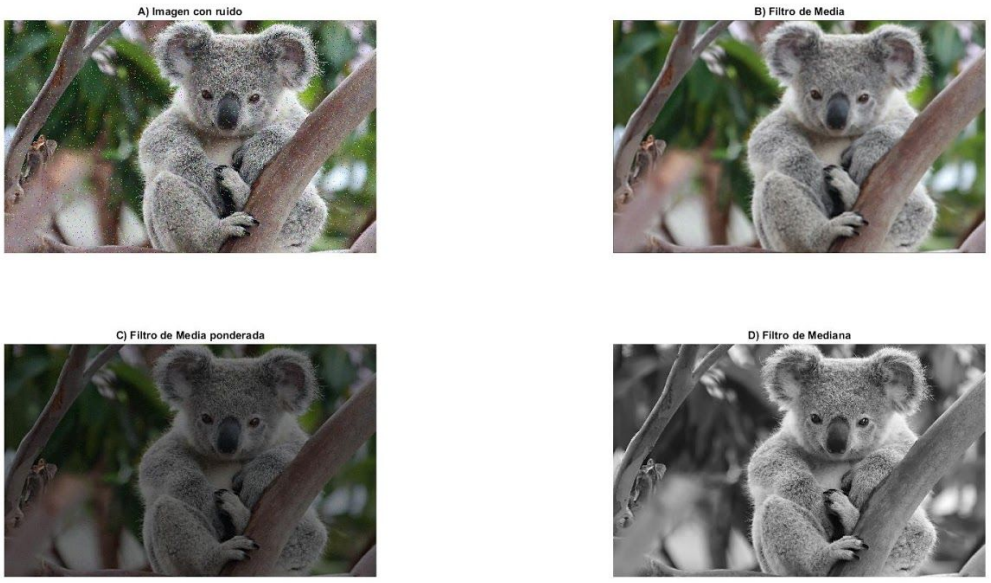

Fig. 6. A) Imagen con ruido de pimienta, B) Filtro de media, C) Filtro de media ponderada, D) Filtro de mediana.

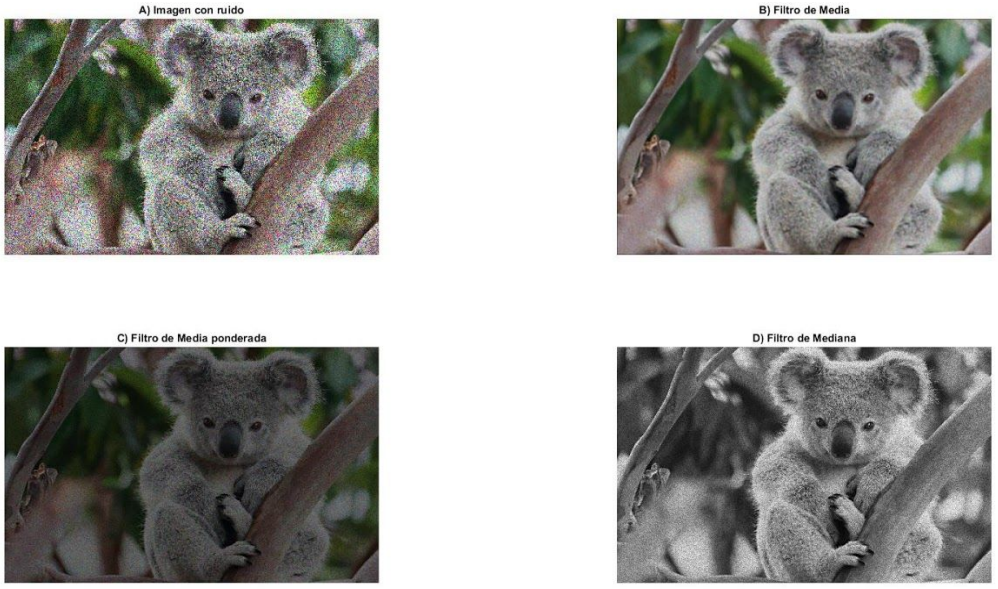

Fig. 7. A) Imagen con ruido uniforme multiplicativo, B) Filtrado de media, C) Filtrado de media ponderada, D) Filtrado de mediana.

Algoritmo: Filtrado pasa bajas

```
Escribir 'Leer imagen'
I = Leer Imagen como RGB

Escribir 'Filtrado de ruido gaussiano'
I_Gaussian1 = I + Gauss con 0.2 de varianza
IG_Gaussian1 = I_Gaussian1 en escala de grises

kmed = (1/81).*(matriz de unos de 9x9)
kPond = (1/32).*[1 2 1; 2 4 2; 1 2 1]
[m, n] = dimensiones de I_Gaussian1

Filt_media = Realizar correlación respecto a kmed sobre I_Gaussian1
Filt_Pond= Realizar correlación respecto a kPond sobre I_Gaussian1

G_Filt_media = Realizar correlación respecto a kmed sobre IG_Gaussian1
G_Filt_Pond= Realizar correlación respecto a kPond sobre IG_Gaussian1

Para z = 0 hasta 2
    Para i = 1 hasta m-1
        Para j = 1 hasta n-1
            Filt_mediana = La mediana de la vecindad de la I_Gaussian1(i, j)

        end
    end
end
Para i = 1 hasta m-1
    Para j = 1 hasta n-1
        G_ Filt_mediana = La mediana de la vecindad de la IG_Gaussian1(i, j)

    end
end


Escribir 'Filtrado de ruido sal'

I_Salt = I + Salt con densidad de 0.08
IG_Salt = I_Salt en escala de grises

Filt_media_S = Realizar correlación respecto a kmed sobre I_Salt
Filt_Pond_S = Realizar correlación respecto a kPond sobre I_Salt

G_Filt_media_S = Realizar correlación respecto a kmed sobre IG_Salt
```

```
G_Filt_Pond_S = Realizar correlación respecto a kPond sobre IG_Salt

Para z = 0 hasta 3
    Para i = 1 hasta m-1
        Para j = 1 hasta n-1
            Filt_mediana_S = La mediana de la vecindad de la I_Salt (i, j)

        end
    end
end
Para i = 1 hasta m-1
    Para j = 1 hasta n-1
        G_Filt_mediana_S = La mediana de la vecindad de la IG_Salt (i, j)

    end
end

Escribir 'Filtrado de ruido pimienta

I_pepper =  I + Pepper con densidad de 0.08
IG_Pepper = I_pepper en escala de grises

Filt_media_P= Realizar correlación respecto a kmed sobre I_pepper
Filt_Pond_P = Realizar correlación respecto a kPond sobre I_pepper

G_Filt_media_P = Realizar correlación respecto a kmed sobre IG_Pepper
G_Filt_Pond_P = Realizar correlación respecto a kPond sobre IG_Pepper

Para z = 0 hasta 2
    Para i = 1 hasta m-1
        Para j = 1 hasta n-1
            Filt_mediana_P = La mediana de la vecindad de la I_pepper (i, j)

        end
    end
end
Para i = 1 hasta m-1
    Para j = 1 hasta n-1
        G_Filt_mediana_P = La mediana de la vecindad de la IG_Pepper (i, j)

    end
end
```

Escribir 'Filtrado de ruido uniforme'

I_Uniforme1= I + Speckle con 0.18 de varianza
IG_Uniforme1= I_Uniforme1 en escala de grises

Filt_media_U= Realizar correlación respecto a kmed sobre I_Uniforme1
Filt_Pond_U = Realizar correlación respecto a kPond sobre I_Uniforme1

G_Filt_media_P = Realizar correlación respecto a kmed sobre IG_Uniforme1
G_Filt_Pond_P = Realizar correlación respecto a kPond sobre IG_Uniforme1

```
Para z = 0 hasta 2
    Para i = 1 hasta m-1
        Para j = 1 hasta n-1
            Filt_mediana_U = La mediana de la vecindad de I_Uniforme1(i, j)

        end
    end
end
Para i = 1 hasta m-1
    Para j = 1 hasta n-1
        G_Filt_mediana_U = La mediana de la vecindad de IG_Uniforme1(i, j)

    end
end
```

Fin del algoritmo

Análisis de la imagen:
En el caso del ruido gaussiano los filtros respetan los valores de intensidad original de la imagen en el caso del filtro de media, donde se puede apreciar que sí se recuperó la imagen original sin corregir la intensidad añadida por la curva de Gauss. Distinto del caso del filtro de media ponderada donde la imagen recuperada se encuentra con una tonalidad más obscura que la original, mientras que el de mediana es el que más respeto las tonalidades sin suavizar demasiado los bordes.

En el caso del ruido de sal, el filtro de media logró remover el ruido generado pero en las zonas donde se encontraban los distintos puntos generados por el ruido de sal se ven cierta suavización, como motas de pintura esparcida de forma local. En el caso del filtro de media ponderada la imagen todavía contiene cierta cantidad de ruido al igual que se vio oscurecida de forma considerable. El filtro de mediana también realizó un

buen trabajo eliminando el ruido, con algunos cuantos puntos de color blanco pero sin suavizar notoriamente la imagen.

Ruido de pimienta, el resultado es muy parecido al de sal. Tenemos que el filtro de media logró eliminar el ruido en su totalidad se nota un poco suavizado en sus bordes. El de media ponderada se observa oscurecida y con un suavizado muy notorio. Y el filtro de mediana logró eliminar en su mayor parte el ruido, y su suavizado es difícil de notar.

En el caso de ruido uniforme multiplicativo, sólo el filtro de media logró restaurar la imagen de mejor forma. El filtro de mediana no logró su cometido ya que los pixeles de en la vecindad de evaluación puede tener más de un solo pixel de ruido, mientras que el filtro de media ponderada aparte del oscurecimiento, se mantuvo cierta cantidad de ruido, notorio en el pelaje del koala.

3. *Filtros pasa altas*

Para este proceso se requirió generar los kernels (K) de cada uno de los filtros a realizar; filtro de sustracción de media y Laplaciano, ya que el menos Laplaciano es el negativo del Laplaciano. Donde la operación matemática que sigue se presenta a continuación:

$$f_{filter}(x,\ y) = \frac{\sum_{s=-b}^{b}\sum_{t=-a}^{a} k(s,\ t) f(x+s,\ y+t)}{\sum_{s=-b}^{b}\sum_{t=-a}^{a} k(s,\ t)}$$

eq. 3.

Donde K(s, t) es el kernel del filtro, f(x, y) es la imagen original.

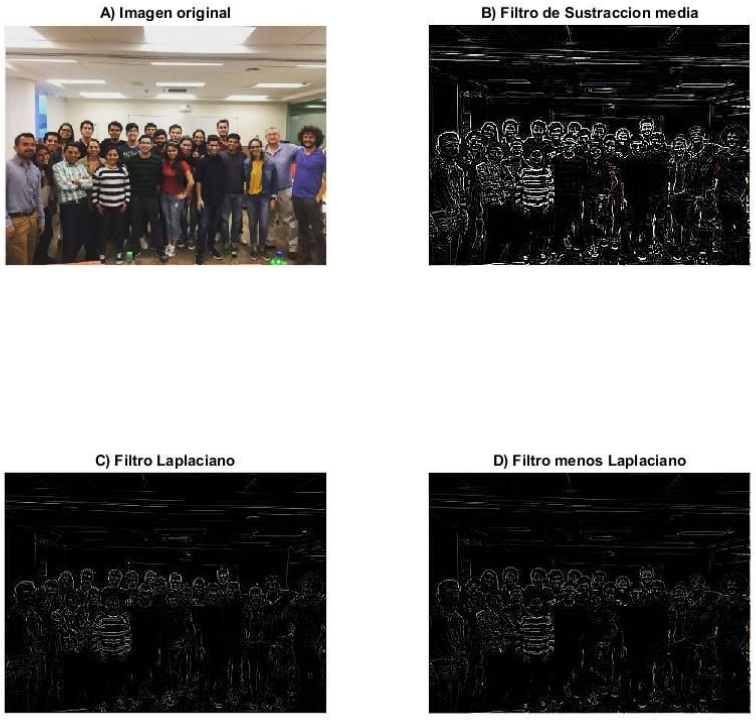

Fig. 8. A) Imagen original, B) Sustracción de media de la imagen original, C) Filtro Laplaciano de la imagen original, D) Filtro menos Laplaciano de la imagen original.

Fig. 9. A) Imagen original en escala de grises, B) Filtro de sustracción de media de la imagen original, C) Filtro laplaciana de la imagen original, D) Filtro menos laplaciano de la imagen original.

Algoritmo: Filtrado pasa altas

```
Escribir 'Leer imagen'
I = Leer Imagen como RGB
IG = I en escala de grises

K_Sust_Media = [-1 -1 -1; -1 9 -1; -1 -1 -1]
K_Laplaciano = [0 -1 0; -1 4 -1; 0 -1 0]
K_min_Lap = -K_Laplaciano

Filt_sust_media = Realizar correlación respecto a K_Sust_Media sobre I
Filt_Laplaciano = Realizar correlación respecto a K_Laplaciano sobre I
Filt_min_Lap = Realizar correlación respecto a K_min_Lap sobre I

G_Filt_sust_media = Realizar correlación respecto a K_Sust_Media sobre IG
G_Filt_Laplaciano = Realizar correlación respecto a K_Laplaciano sobre IG
G_Filt_min_Lap = Realizar correlación respecto a K_min_Lap sobre IG
```

Fin del algoritmo

Análisis de la imagen:
Todos tienen en común que son filtros pasa altas por lo que sólo permiten que las intensidades con cambios abruptos en contraste, por lo que en su mayor parte son detectores de contornos pero cada uno es capaz de extraer diferentes tipos de contornos.

Tomando en cuenta primero que nada el filtro de sustracción de media, figuras 8-B y 9-B, se puede observar que es más común que los colores más próximos a blanco como la camisa a rayas de blanco y negro, es lo que resaltó más ya que su cambio de intensidad es muy alto distinto de las camisas más obscuras como azul marino, o negro, en las que no se captaron cambios aunque sí existe un borde visible entre los torsos. También es bueno para detectar los reflejos de luz en las caras.Por lo que es un detector de intensidades relativamente altas, en valores próximos a un color blanco poniendo como ejemplo en una imagen de 8 bits serían intensidades superiores a 180.

Distinto del filtro de sustracción de media la representación de los bordes de un filtro laplaciano son más definidos. También es capaz de ignorar ciertas texturas visibles en la ropa como por ejemplo la camisa azul en la izquierda, donde sustracción de media

captó los olanes, el laplaciano no los tomo como referencia, esto podría ser debido a que en su vecindad se encuentran ceros, específicamente en las esquinas. pero al igual que el filtro de sustracción de media no es capaz de captar los cambios de contraste entre la ropa más oscura, debido que sólo es filtra cambios radicales. En el caso del filtro menos laplaciano, no es el complemento del laplaciano si no más bien son los valores de la vecindad del cambio más radical de intensidad, por lo que en estos casos se puede a llegar captar las texturas.

4. Filtros direccionales

Los filtros direccionales, normalmente utilizados para detección de bordes, como su nombre implica son capaces de seguir una dirección y resaltar los cambios de intensidad que sigan el mismo camino. Existen distintos tipos en este caso se revisaron el filtro direccional, Sobel, Canny, y Prewitt, donde cada uno tiene un template específico sobre el cual genera su evaluación.

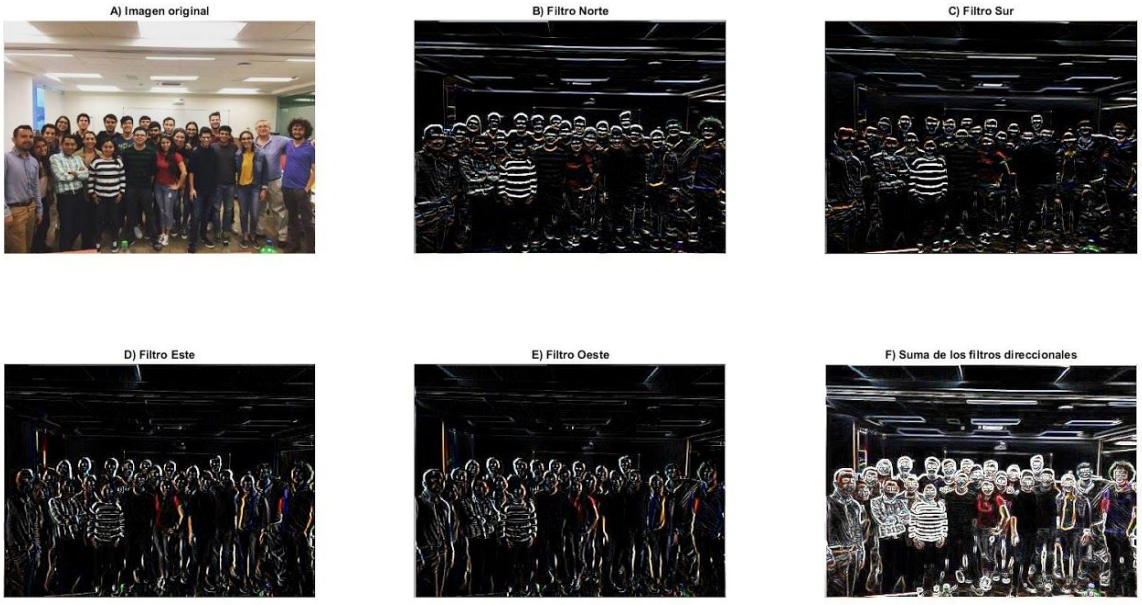

Fig. 10. A) Imagen original, B) Filtro direccional norte, C) Filtro direccional sur, D) Filtro direccional este, E) Filtro direccional oeste, F) Suma de todos los filtros direccionales

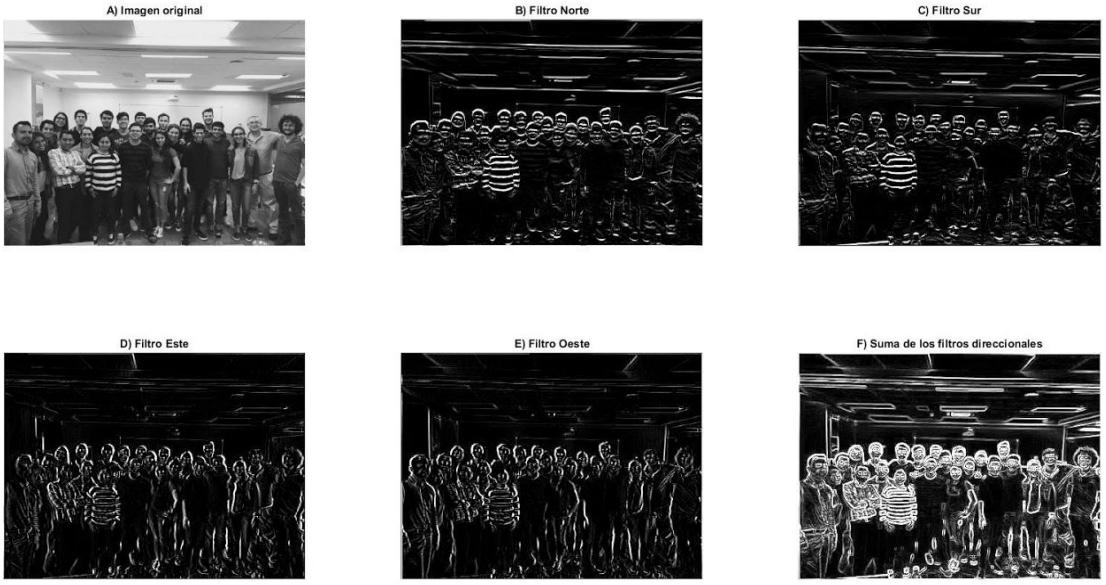

Fig. 11. A) Imagen original en escala de grises, B) Filtro norte de la imagen original, C) Filtro sur de la imagen original, D) Filtro este de la imagen original, E) filtro de la imagen oeste, F) Suma de los filtros.

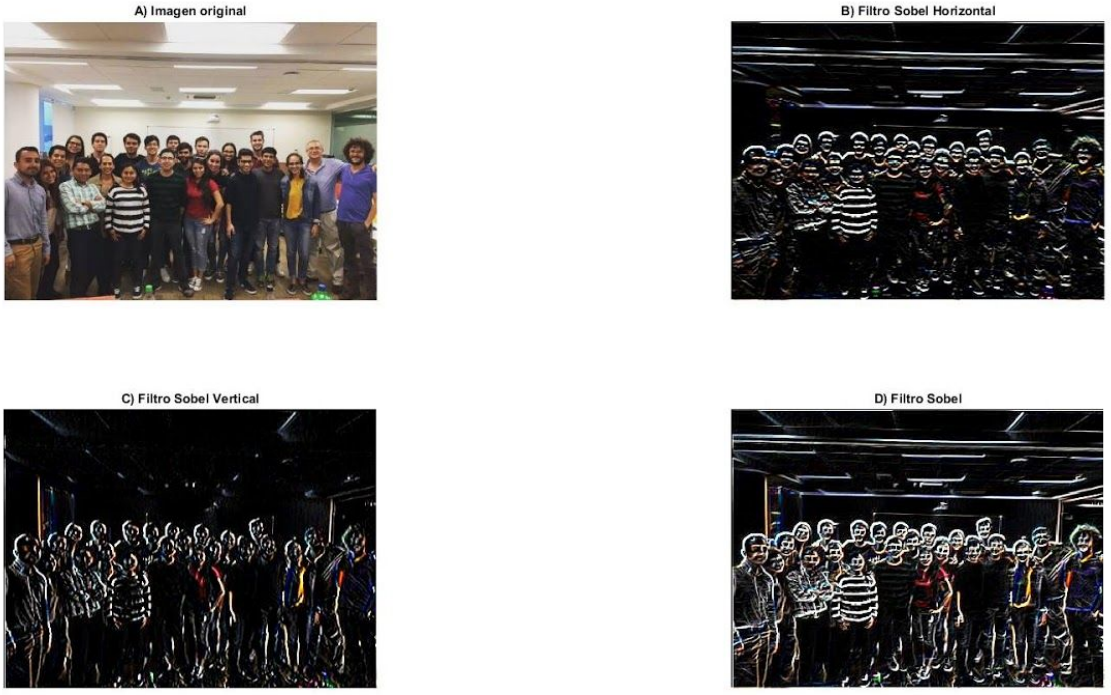

Fig. 12. A) Imagen original, B) Filtro Sobel horizontal, o Sobel C, C) Filtro Sobel vertical, o F, D) Suma de los filtros.

Fig. 13. A) Imagen original en escala de grises, B) Filtro Sobel horizontal, o Sobel C, C) Filtro Sobel vertical, o F, D) Suma de los filtros.

Fig. 14. A) Imagen en escala de grises, B) Filtro canny de la imagen original

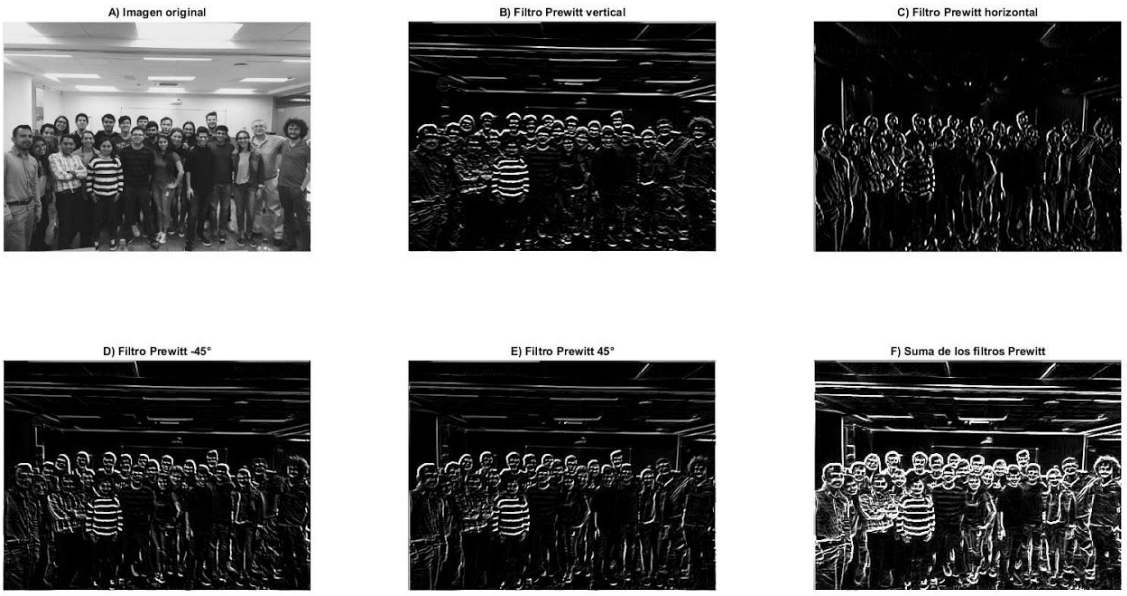

Fig. 15. A) Imagen original en escala de grises, B) imagen filtrada con Prewitt vertical, C) Filtro Prewitt horizontal, D) Filtro Prewitt de -45°, E) Filtro Prewitt de 45°, F) Suma de los filtros.

Algoritmo: Filtrado direccional

Escribir 'Leer imagen'
I = Leer Imagen como RGB
IG = I en escala de grises

Escribir 'Direccionales'

K_norte = [1 1 1; 1 -2 1 ; -1 -1 -1];
K_este = [-1 1 1; -1 -2 1 ;-1 1 1] ;

Filt_norte = Realizar correlación respecto a K_norte sobre I
Filt_sur = Realizar convolución respecto a K_norte sobre I
Filt_este = Realizar correlación respecto a K_este sobre I
Filt_oeste = Realizar convolución respecto a K_este sobre I

F_tot = Filt_norte + Filt_sur + Filt_este + Filt_oeste

G_Filt_norte = Realizar correlación respecto a K_norte sobre IG
G_Filt_sur = Realizar convolución respecto a K_norte sobre IG
G_Filt_este = Realizar correlación respecto a K_este sobre IG
G_Filt_oeste = Realizar convolución respecto a K_este sobre IG

G_F_tot_D = G_Filt_norte + G_Filt_sur + G_Filt_este + G_Filt_oeste

Escribir 'Filtrado Sobel'
K_Sobel_C = [-1 0 1; -2 0 2; -1 0 1]
K_Sobel_F = K_Sobel_C transpuesto

Filt_C = Realizar correlación respecto a K_Sobel_C sobre I
Filt_F = Realizar correlación respecto a K_Sobel_F sobre I

SF_tot_S = Filt_C + Filt_F

G_Filt_C = Realizar correlación respecto a K_Sobel_C sobre IG
G_Filt_F = Realizar correlación respecto a K_Sobel_F sobre IG

GF_tot_S= G_Filt_C + G_Filt_F

Escribir 'Filtrado Canny'
K_Canny = [2 4 5 4 2; 4 9 12 9 4; 5 12 25 12 5; 4 9 12 9 4; 2 4 5 4 2] *(1/159)

Filt_Canny = Realizar correlación respecto a K_Canny sobre IG

Escribir 'Filtrado Prewitt'
K_Prewit_hor = [-1 -1 -1; 0 0 0; 1 1 1]
K_Prewit_verr = K_Prewit_hor transpuesto
K_Prewit_45 = K_Prewit_hor rotado 45
K_Prewit_min45 = K_Prewit_hor rotado -45

G_Filt_hor = Realizar correlación respecto a K_Prewit_hor sobre IG
G_Filt_ver = Realizar convolución respecto a K_Prewit_verr sobre IG
G_Filt_45 = Realizar correlación respecto a K_Prewit_45 sobre IG
G_Filt_min45 = Realizar convolución respecto a K_Prewit_min45 sobre IG

G_F_tot_P = G_Filt_hor + G_Filt_ver + G_Filt_45 + G_Filt_min45

Fin del algoritmo

Análisis de la imagen:
Primero evaluando los resultados de las figuras 10 y 11 donde se utilizaron filtros direccionales norte, sur, este, y oeste, como su función es resaltar el contraste que llevé la dirección del nombre del filtro se puede observar que las zonas realzadas aparentan ser los bordes donde incide un haz de luz generando un contraste mayor. Esto se puede notar en el reflejo de la luz en las cabezas, al igual que en los colores de ropa más intensos, figura 10. Aunque la reconstrucción final de la imagen tiene zonas de saturación, posiblemente porque tanto el filtro norte como el sur captaron bordes similares. Lo mismo sucedió en su análisis en escala de grises pero en este caso es más fácil notar que el filtro también pasó los olanes que se forman en la ropa.

En el caso del filtro sobel direccional, captó de mejor forma los bordes pero a cambió parece que cambió mucho el color de fondo probablemente para contrastar de mejor forma la imagen resultante. Y también es notorio que tuvo problemas con mantener el color original de la imagen como se puede observar en las figuras 12-C y 12-D, donde marco de color azul partes de la camisa amarilla, confundiendo que la chamarra tomaba una mayor área de la imagen. Al igual que la reconstrucción que logró generar parace que tiene montado ruido ya que se observa sal y, o, pimienta. El ruido es más notorio en el filtrado de la imagen en escala de grises, al igual que se nota que no logró definir bien los bordes ya que en ciertas zonas no hay continuidad, o bien tienen saltos, o bultos, en los bordes que sí logró definir. Y distinto del filtrado a color, no logró delimitar los focos en la parte superior.

Filtrado Canny fue el siguiente en ser evaluado distinto de los anteriores sólo se realizó el filtrado de la imagen en escala de grises. En casos anteriores se identificaban los bordes como aparecían en la imagen sin ser alterados, en el caso del filtro Canny esto no es de esta forma ya que primero difumina los bordes para hacer más aparentes los cambios de intensidad, o bien para ignorar los cambios de intensidad que no forman parte los bordes de la figura como serían los olanes de la ropa (no tan marcados), al igual que este proceso de difuminación se puede observar en objetos que originalmente son rectos que después del filtrado tienen cierta concavidad. Y distinto del filtro Sobel, el filtro Canny no aparenta tener ruido de sal, o pimienta, en su imagen final al igual que no necesito de la suma de otros filtros direccionales para formar una imagen. Cabe mencionar que como en casos anteriores, es complicado para el filtro discernir en los cambios de intensidad de la ropa ya que tienen una paleta de color similar.

Por último filtrado a partir de las máscaras de Prewitt, logró reconstruir la mayor parte de la imagen original a partir de sus bordes, captó la texturas de la mayoría de la ropa al igual que los bordes donde la ropa cambia de color por uso, el problema que se observa es que en algunos lugares (como el techo) captó ciertos puntos que no son bordes, u objetos, sino más bien como ruido gaussiano por su similitud a neblina. A diferencia de los filtrados anteriores en este caso se puede distinguir entre los torsos de varios de los individuos en la foto.

5. Filtro suavizado conservador

Aunque la forma en que trabaja el filtrado de suavizado conservador es similar al filtro de mediana, sólo que en lugar de agregar el valor medio en un arreglo de valores ordenado de forma ascendente, o descendente, busca el valor mínimo, y máximo, del arreglo de vecinos y compara su valor con el pixel central del template. En el caso donde el máximo del arreglo es menor que el máximo del pixel central, se cambia el valor del pixel central por el máximo del arreglo, y lo mismo sucede para el mínimo; si el mínimo del arreglo es mayor que el mínimo del pixel central entonces se hace el cambio. El resultado es una imagen teóricamente sin ruido de sal y pimienta, figuras 16 y 17.

Fig. 16. A) Imagen original, B) Imagen filtrada

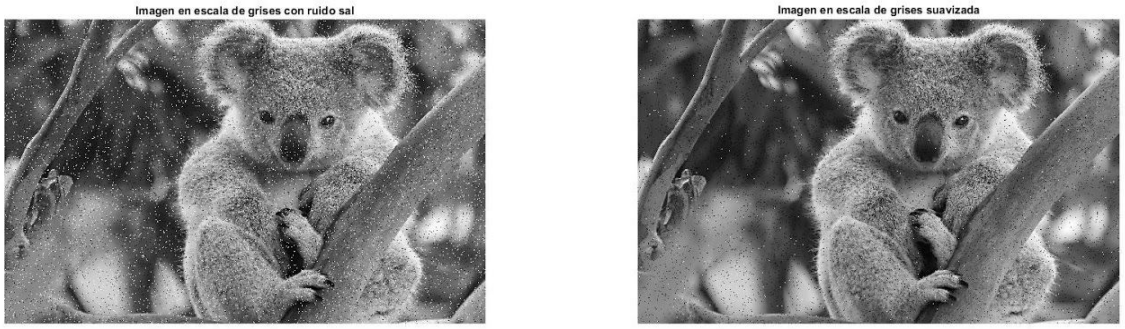

Fig. 17. A) Imagen original en escala de grises, B) imagen filtrada

Algoritmo: Filtrado pasa altas

```
Escribir 'Leer imagen'
I = Leer Imagen como RGB
IG = I en escala de grises

I_Salt = I + Salt con densidad de 0.05
IGS = I_Salt en escala de grises;

[m, n] = dimension de IG

arr = arreglo de ceros con dimensiones (1,8)
Canvas1 es una matriz de ceros con dimensiones (m-2, n-2, 3)
Canvas2 es una matriz de ceros con dimensiones (m-2, n-2)

a = 1;
b = 1;
p = 1;

Escribir 'Filtro suavizado conservador'

for z = 1:3
   for i = 2:m-1
      for j = 2:n-1
         for s = -a:a
            for t = -b:b
              if(s ~= 0 && t ~= 0)
                 arr(p) = IS(i+s,j+t,z)
                 p = p + 1;

              end
            end
         end
         p = 1;

         si (IS(i,j,z) > max(arr))
            Canvas1(i-1,j-1,z) = max(arr);

         o si(IS(i,j,z) < min(arr))
            Canvas1(i-1,j-1,z) = min(arr);

         de otra forma
            Canvas1(i-1,j-1,z) = IS(i,j,z);
```

```
            end

        end
    end
end

a = 1;
b = 1;
p = 1;

for i = 2:m-1
    for j = 2:n-1
        for s = -a:a
            for t =-b:b
                if(t ~= 0 && s ~= 0)
                  arr(p) = IGS(i+s,j+t);
                  p = p + 1;

                end
            end
        end
        p = 1;

        Si(IGS(i,j) > max(arr))
            Canvas2(i-1,j-1) = max(arr);

        O si(IGS(i,j) < min(arr))
            Canvas2(i-1,j-1) = min(arr);

        de otra forma
          Canvas2(i-1,j-1) = IGS(i, j);

        end

    end
end
```

Fin del algoritmo

Análisis de la imagen:
Aunque logró eliminar la mayor parte del ruido de sal y pimienta, no logró hacerlo en su totalidad. La razón de esto es porque la imagen original tiene un ruido con densidad alta, lo que significa que en más de una ocasión los valores de las vecindad tendrá ruido de sal, o pimienta, lo que cuando se hace la evaluación de máximos, o mínimos, resulta que el valor máximo del arreglo, o mínimo, es igual al pixel central por lo que el filtro admite que el ruido permanezca en el pixel central de la imagen resultante. Apreciable en ambos resultados, la imagen en escala de grises y a color, figuras 16-B y 17-B.

Conclusión:

Tener la capacidad de restaurar información de una imagen aun después de haber adquirido algún tipo de ruido, ya sea por el medio de captura o de procesamiento, facilita por ejemplo la segmentación de información (aislamiento de datos) ya sea a partir de la ubicación de un objeto por la definición de sus bordes. O bien facilitando el procesamiento a partir eliminar valores como ruido de sal donde los valores máximos serán muchos más común, o cuando se tiene una desviación normal montada sobre la imagen que evita que se puede generar una ecualización correcta de la imagen.

Referencias:

- Rafael Gonzalez. (2008). Digital image processing. Estados unidos: Pearson.