
Filtrado de Imágenes y ecualización de Histograma

Profesor:

Javier Ruiz del Solar

Estudiante:

Giovanni Pais L.

Auxiliar:

Patricio Loncomilla

Ayudantes:

Nicolás Cruz

Francisco Leiva

Fecha:

05 de Octubre de 2018

Índice

1. Introducción	1
2. Marco Teórico	2
2.1. Convolución	2
2.2. FFT 1D y 2D	2
2.3. Histograma de la imagen	2
2.4. Ecualización de histograma	3
3. Desarrollo	4
3.1. Filtros	4
3.1.1. Filtros Pasa Bajos	6
3.1.2. Filtros Pasa Altos	11
3.2. Ecualización de Histogramas	15
3.2.1. Implementación	15
3.2.2. Resultados	17
4. Conclusiones	21

Índice de figuras

1.	Imagen original y fft asociado	7
2.	Imagen filtrada con filtro recto 2D	8
3.	Imagen filtrada con filtro recto 1D fila + columna	8
4.	Imagen filtrada con filtro Gaussiano 2D	9
5.	Imagen filtrada con filtro Gaussiano 1D fila + columna	9
6.	Imagen original de figuras.png	10
7.	Imagen filtrada con filtro Recto 2D	10
8.	Imagen filtrada con filtro Recto 1D fila + columna	10
9.	Imagen filtrada con filtro Gausiano 2D	11
10.	Imagen filtrada con filtro Gaussiano 1D fila + columna	11
11.	Imagen filtrada con filtro Prewitt vertical	12
12.	Imagen filtrada con filtro Prewitt horizontal	12
13.	Imagen filtrada con filtro Laplaciano	13
14.	Imagen filtrada con filtro LoG	13
15.	Imagen filtrada con filtro Recto 2D	14
16.	Imagen filtrada con filtro Recto 1D fila + columna	14
17.	Imagen filtrada con filtro Laplaciano	15
18.	Imagen filtrada con filtro LoG	15
19.	Ecualización se imagen agua.png	17
20.	Ecualización se imagen casa.png	18
21.	Ecualización se imagen constr.png	18
22.	Ecualización se imagen corteza.png	19
23.	Ecualización se imagen mala _{ilum} .jpg	19
24.	Ecualización se imagen termal.png	20

1. Introducción

El objetivo de la tarea es realizar filtrado de imágenes a través de convoluciones de máscaras de distintos tipos, se usan distintos filtros pasa bajos y pasa altos en 2 imágenes para comparar los resultados. Además con fin de comparar las variaciones entre cada filtro se aplica una fft a las imágenes y se presenta junto a cada imagen como resultado. En la segunda parte de la tarea se busca aplicar ecualización de la intensidad de luz en 5 imágenes distintas a través de histogramas, para así usar las probabilidades entre las intensidades de luz para lograr una mejor distribución y un mejor contraste.

Para el desarrollo de la tarea se utiliza la librería OpenCV en el lenguaje C++, principalmente para cargar las imágenes, convertirlas y mostrarlas, dado que el objetivo de la tarea es implementar los métodos de filtrado y ecualización, no usar las herramientas ya proporcionadas por OpenCV. Para esto se tiene 2 packages, src_conv y src_histeq, con las imágenes asociadas a cada parte de la tarea y el código principal donde se implementa la tarea. La compilación se realiza a través de cmake usando la configuración dada por el cuerpo docente.

A continuación en este informe se encuentran las siguientes secciones:

1. Marco Teórico: Se explican brevemente los conceptos teóricos utilizados en la realización de la tarea, como son explicar la operación de convolución, la función FFT, el cálculo de histograma y la ecualización de histogramas.

2. Desarrollo: Se explica la implementación de la operación de convolución y la función de filtrado para la primera parte de la tarea, además se explican las máscaras usadas para cada filtro. Se muestran las imágenes resultantes de la aplicación de cada filtro y el resultado de usar la función fft, en torno a estos resultados se realiza un análisis comparando los filtros. Para la segunda parte de la tarea se explica la implementación de la ecualización de histograma y se muestran los resultados, comparando la imagen original y la imagen ecualizada.

3. Conclusiones: Finalmente se concluye a partir del análisis general de los resultados, el efecto de los filtros y la ecualización con histogramas sobre las imágenes, como varían los resultados para distintas imágenes y distintos parámetros. También se concluyen sobre los conocimientos adquiridos y dificultades.

2. Marco Teórico

2.1. *Convolución*

Una convolución matemático que toma 2 funciones y realiza una operación de superposición por barrido entre ellas para generar una nueva función. En el procesamiento de señales es muy utilizada debido a la representación que tiene este operador en el espacio de frecuencia correspondiendo a una multiplicación. En tiempo discreto se define de la siguiente forma:

Dada la definición de la función de convolución en procesamiento de imágenes se utiliza para aplicar distintos filtros a una imagen, donde se tiene la imagen como una de las funciones y una mascara como la otra función de la convolución. Donde para cada pixel se hace un barrido en torno a una vecindad del tamaño de la mascara, graficamente se observa en la siguiente imagen.

2.2. *FFT 1D y 2D*

En el análisis de señales es sumamente útil tener conocimiento del espacio de frecuencia, dado que puede presentar información importante que en el espacio del tiempo no se puede observar. Esto también se puede aplicar para nueva información sobre una imagen en procesamiento de imágenes. La transformada de Fourier permite pasar del espacio del tiempo al espacio de frecuencia, dado que las imágenes son continuas se tomará la definición discreta de transformada de Fourier.

Para aplicar la transformada de Fourier es altamente utilizada la función FFT (Fast Fourier Transform) dado que está implementada inteligentemente para ser eficiente en su ejecución, esta tiene una versión para 1D y para 2D.

2.3. *Histograma de la imagen*

Una forma de analizar las características de una imagen es construyendo un histograma de intensidades, donde se recorre la imagen pixel a pixel y en una arreglo de 256 se van contando cuantos pixeles tienen cierta intensidad de luz. Este histograma permite describir una imagen y ver la distribución de intensidad que posee, viendo si tiene problemas de contraste. Además puede usarse para distintas aplicaciones como es ecualización o una versión sencilla de un descriptor de

objetos.

2.4. Ecualización de histograma

La función de ecualizar usando histogramas, aprovecha el histograma como probabilidad de intensidad de un pixel y realizando la suma acumulada de intensidades puede verificar si la distribución de intensidades es uniforme, una vez detectado esto puede usarse la probabilidad de una intensidad para ponderar cada pixel y lograr la distribución uniforme de la imagen.

3. Desarrollo

3.1. Filtros

Para la implementación de filtros se crearon las funciones **convolucion()** y **Filtrar()**, además cada filtro tiene una función asociada donde se construye la mascara a aplicar, para luego ser llamada en el main del código.

La función **convolucion()** recibe 3 variables **Mat** la entrada, la mascara y la salida donde se almacena el resultado, para la implementación se utilizaron 4 ciclos for, dado que como se menciona en el marco teórico se aplica una mascara o kernel sobre cada pixel. Para la aplicación de la mascara en un pixel son necesarios 2 ciclos for para así recorrer la vecindad en torno al pixel analizado, en este caso corresponden a los ciclos con indices i,j. Se recorre la vecindad del pixel y se multiplica el valor del pixel de la imagen con el valor de la máscara, el resultado de esta operación se acumula en la variable out y luego se guarda en la imagen de salida output. Para recorrer el resto de los pixeles de la imagen se utilizan los ciclos con indices r y c, es importante notar que las iteraciones no consideran los primeros y últimos pixeles de la imagen par así solucionar el problema de convolucionar los bordes, los pixeles excluidos estan dados por el tamaño de la máscara. En el siguiente código se puede observar esta implementación de la función de convolución.

```
1 void convolucion(Mat input, Mat mask, Mat output)
2 {
3     for (int r=mask.rows/2; r<input.rows-mask.rows/2; r++) {
4         for (int c=mask.cols/2 ; c<input.cols-mask.cols/2; c++) {
5             float out=0.0;
6             for (int i=0; i<mask.rows;i++) {
7                 for (int j=0; j<mask.cols;j++) {
8                     out+=input.at<float>(r-mask.rows/2+i,c-mask.cols/2+j)*mask.
9                         at<float>(i,j);
10            }
11        }
12        output.at<float>(r,c) = out;
```

```
12     }
13 }
14 }
```

Código 1: Implementación de función de convolución

Para realizar el filtrado de las imágenes se implementa la función **Filtrar()** que recibe la imagen de entrada en escala de grises y la mascara a utilizar, esta función simplemente crea la variable de salida y llama a la función **convolucion()** para luego aplicar la función **fft()** sobre la salida. Tomando los resultados procede a mostrarlos a través de imágenes a través de la función **imshow()** de OpenCV, además guardando la imagen usando la función **imwrite()** y usa la función **cvWaitKey(0)** para mostrar la imagen y pausar el programa. A continuación se observa la implementación de la función **Filtrar()**

```
1 void Filtrar(Mat input,Mat mask){
2     Mat output = input.clone();
3     convolucion(input, mask, output);
4
5     Mat esp_out;
6     esp_out = fft(output);
7     imshow("spectrum_out", esp_out);
8     imwrite("spectrum_out.jpg", esp_out); // Grabar imagen
9
10    output = abs(output);
11    Mat last;
12    output.convertTo(last, CV_8UC1);
13
14    imshow("filtered", last); // Mostrar imagen
15    imwrite("filtered.jpg", last); // Grabar imagen
16    cvWaitKey(0); // Pausa, permite procesamiento interno de OpenCV
17 }
```

Código 2: Implementación de función de Filtrar

3.1.1. Filtros Pasa Bajos

A continuación se muestra el desarrollo de la aplicación de filtros pasa bajos, para esto se utilizaron los siguientes filtros:

1. Pasa bajos recto 2D (3x3)
2. pasa bajos recto filas + columnas (1x3 ->3x1)
3. pasa bajos Gaussiano 2D con $\sigma = 1$ (5x5)
4. pasa bajos Gaussiano filas + columnas con $\sigma = 1$ (1x5 ->5x1)

Cada máscara se definió en una función para cada filtro, donde se implementaron usando ciclos for, la implementación se puede revisar en el código. A continuación a modo de ejemplo se muestra el caso del filtro Gaussiano de 2D:

```
1 void Gauss_2D(Mat input){  
2  
3     Mat mask = Mat(5, 5, CV_32FC1);  
4     float sig=1.0;  
5  
6     for (int i=0; i<5; i++){  
7         for (int j=0; j<5; j++){  
8             mask.at<float>(i,j) = exp(-(1/(2*sig*sig))*((i-mask.rows/2)*(i-  
mask.rows/2)+(j-mask.cols/2)*(j-mask.cols/2)))/(2*M_PI*sig*sig);  
9         }  
10    }  
11    Filtrar(input,mask);  
12}  
13}
```

Código 3: Implementación de máscara Gaussiano 2D

Corteza

A continuación se muestra el resultado de implementar los filtros pasa bajo en la imagen corteza.png:

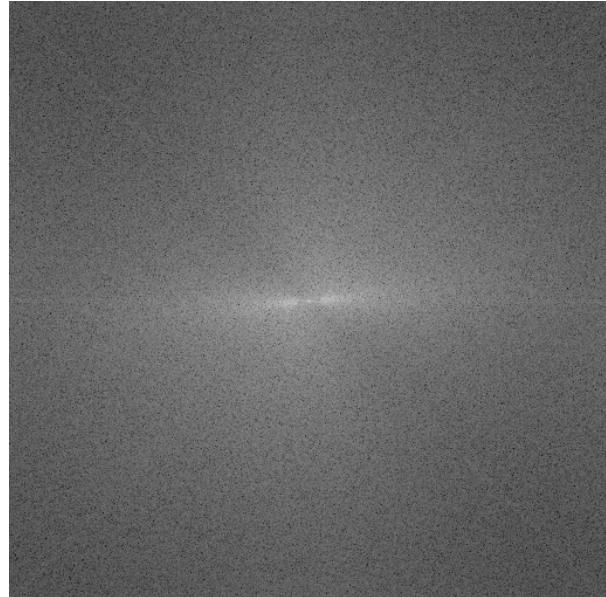


Figura 1: Imagen original y fft asociado

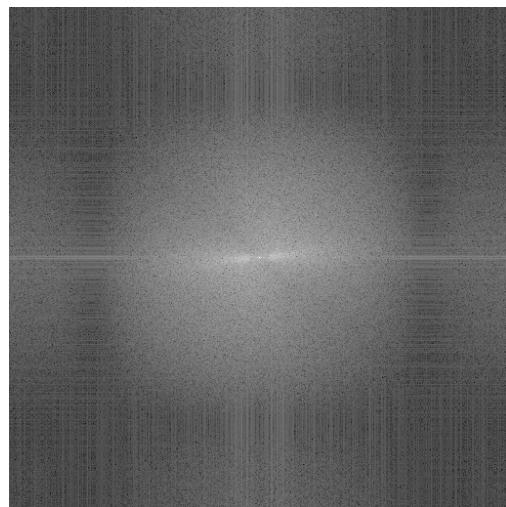


Figura 2: Imagen filtrada con filtro recto 2D

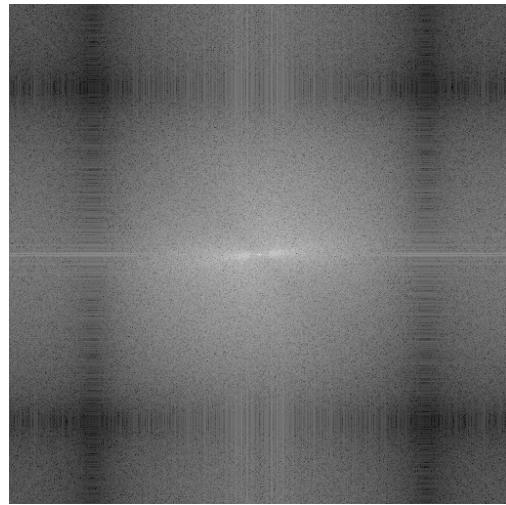


Figura 3: Imagen filtrada con filtro recto 1D fila + columna

Se observa que las imágenes resultantes son más borrosas que la original dado que este filtro elimina bajas frecuencias suavizando los bordes. En el espectro se observa que aparece una zona oscura con forma de cuadrado asociado al filtro recto, se observa que el segundo presenta este cuadro más oscuro. El hecho que ambos resultados sean similares tiene que ver como esta definida la operación de convolución y la mascara usada, donde realizar las suma en 2D o realizar las sumas en cada eje por separado es similar, sin embargo se observa en el espectro que el segundo filtro afecta más la imagen.

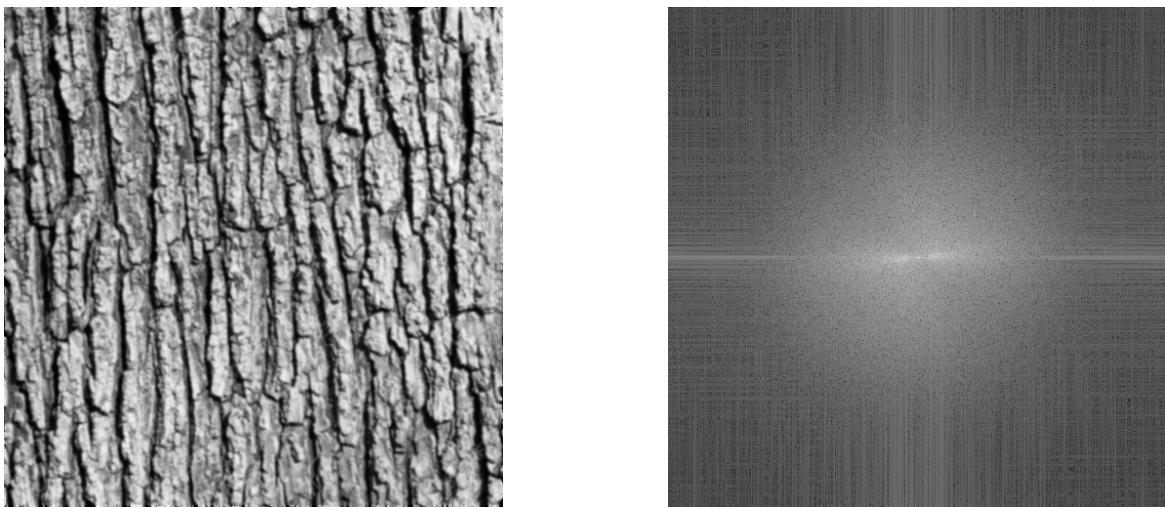


Figura 4: Imagen filtrada con filtro Gaussiano 2D

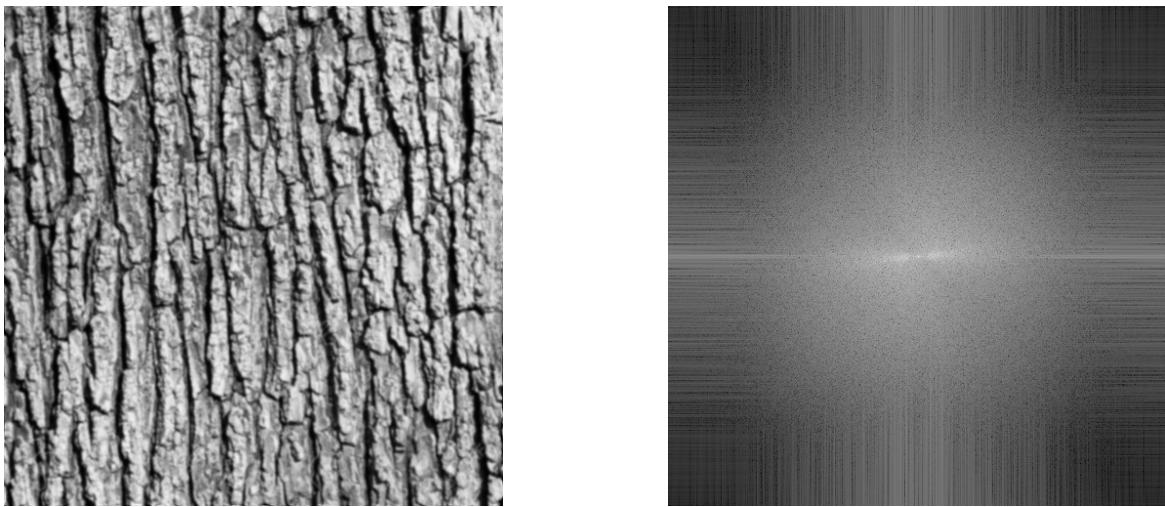


Figura 5: Imagen filtrada con filtro Gaussiano 1D fila + columna

Las imágenes resultantes son muy similares al caso del filtro recto pero en espectro se puede observar que ahora no es una forma cuadrada la que aparece sino más circular para el filtro 2D asociado a una función gaussiana 2 y circular con esquinas ennegrecidas el segundo, esto dado que se suman 2 gaussianas 1D por lo que la intersección de ellas no es circular perfecta.

Figuras geométricas

A continuación se muestra el resultado de implementar los filtros pasa bajo en la imagen `figuras.png`.

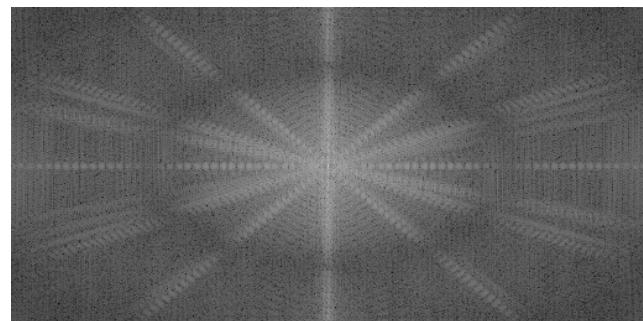
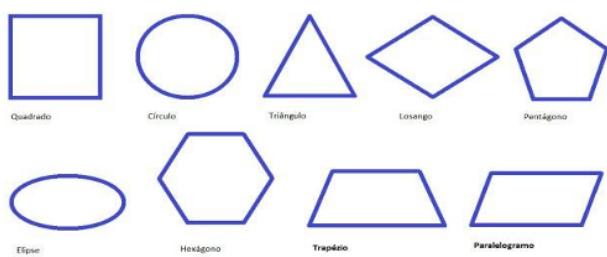


Figura 6: Imagen original de figuras.png

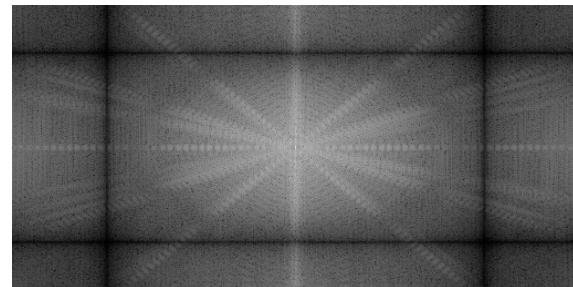
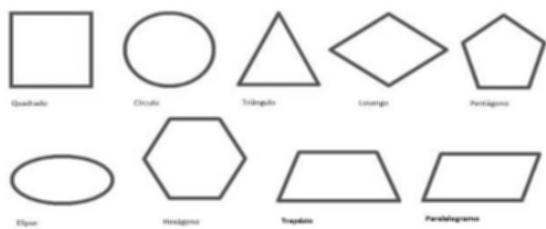


Figura 7: Imagen filtrada con filtro Recto 2D

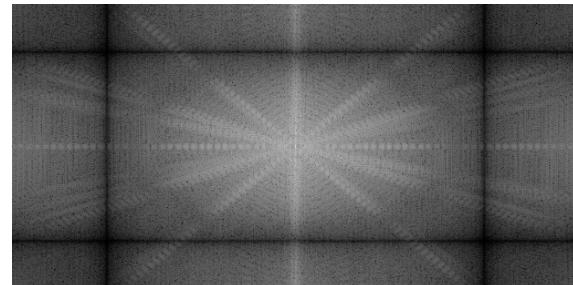
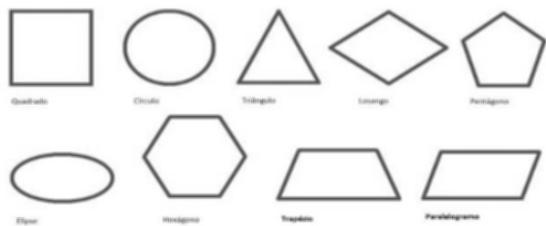


Figura 8: Imagen filtrada con filtro Recto 1D fila + columna

Para esta imagen también se observa que se suavizan los bordes de las figuras geométricas. Se observa muy marcado en el espectro la forma del filtro recto en esta imagen. Por otro lado para esta imagen muestra que aplicar el filtro 2D o aplicar filtro 1D en secuencia tiene el mismo resultado, mostrando mínimas diferencias en la imagen y el espectro.

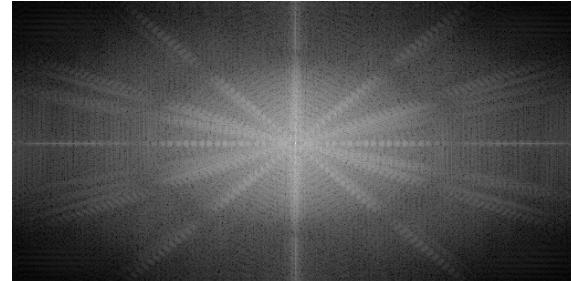
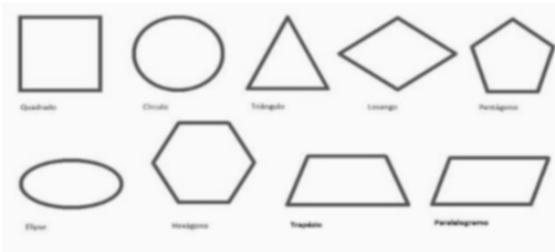


Figura 9: Imagen filtrada con filtro Gausiano 2D

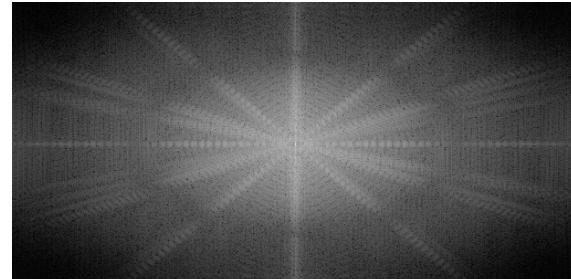
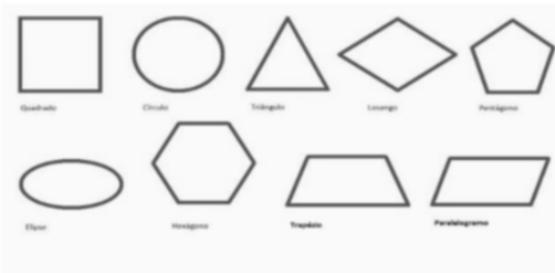


Figura 10: Imagen filtrada con filtro Gaussiano 1D fila + columna

Aplicando el filtro gaussiano se ve la suavización de bordes y más notoria la forma del filtro en el espectro, presentando una forma de una gaussiana de 2 dimensiones. Al igual que el caso anterior, realizar la operación 2D o 1D en secuencia son equivalentes dado la definición de la operación de convolución.

3.1.2. Filtros Pasa Altos

A continuación se muestra el desarrollo de la aplicación de filtros pasa altos, para esto se utilizaron los siguientes filtros:

1. Prewitt vertical (3x3)
2. Prewitt vertical (3x3)
3. Laplaciano (5x5)
4. LoG (5x5)

Corteza

A continuación se muestra el resultado de implementar los filtros pasa alto en la imagen corteza.png:

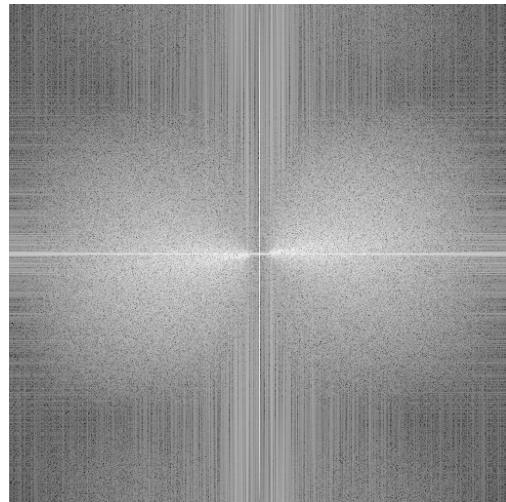


Figura 11: Imagen filtrada con filtro Prewitt vertical

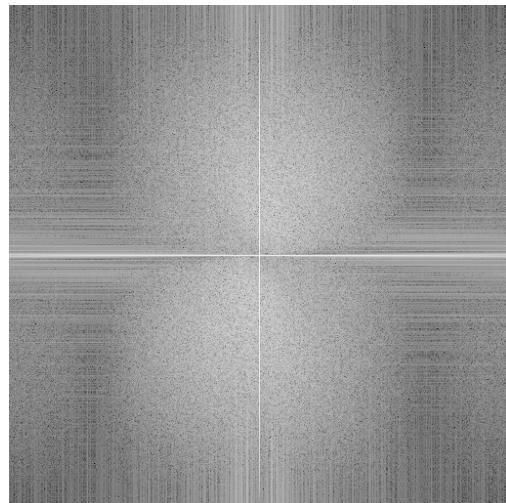


Figura 12: Imagen filtrada con filtro Prewitt horizontal

Se observa que estos filtros oscurecen la imagen y solo dejan con mayor intensidad los bordes, para esta imagen es difícil notarlo debido a la gran cantidad de detalles. Entre estas dos imágenes se puede observar que son considerablemente distintas dado que filtros direccionalizados, donde el primero deja pasar los bordes verticales y el segundo deja pasar los bordes horizontales. En el

espectro se puede notar que las direcciones son perpendiculares dado que el comportamiento está rotado en 90 grados.

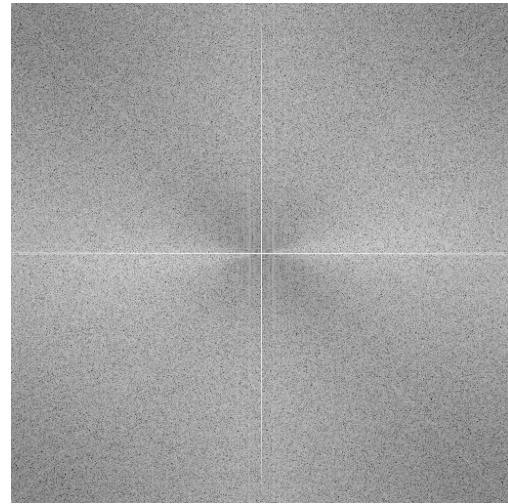
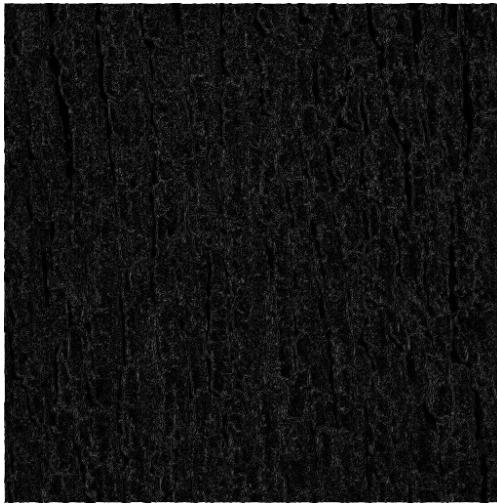


Figura 13: Imagen filtrada con filtro Laplaciano

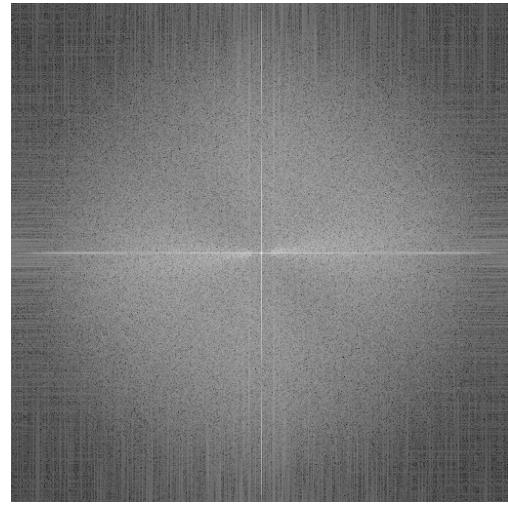
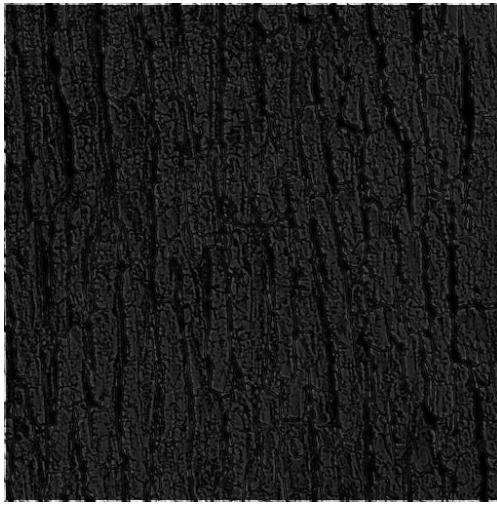


Figura 14: Imagen filtrada con filtro LoG

Se observa que el filtro Laplaciano detecta bordes en todas las direcciones debido a la forma del filtro, el cuál se puede observar en el espectro que es simétrico respecto al origen, el problema que se presenta aquí es que la imagen corteza tiene mucho ruido por lo que no se distingue mucho la imagen original, dado que el ruido es de alta frecuencia y pasa.

Para el caso del filtro LoG se observa que detecta mejor los bordes dado que al igual que el

laplaciano es simétrico y detecta en toda dirección, pero el hecho de aplicar un filtro Gaussiano al inicio (pasa bajo) elimina el ruido que afectaba al filtro laplaciano. Por lo anterior se observa que el resultado remarca los bordes y se distinguen las formas de la figura original.

Figuras geométricas

A continuación se muestra el resultado de implementar los filtros pasa alto en la imagen `figuras.png`.

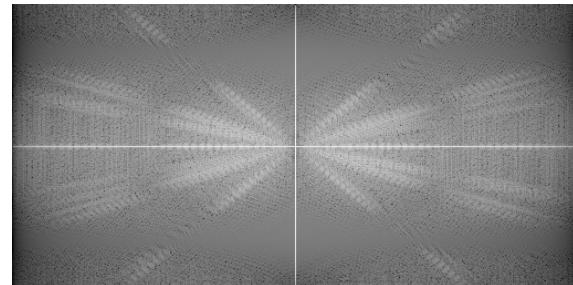


Figura 15: Imagen filtrada con filtro Recto 2D

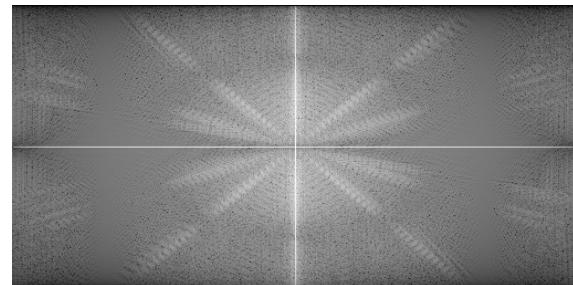
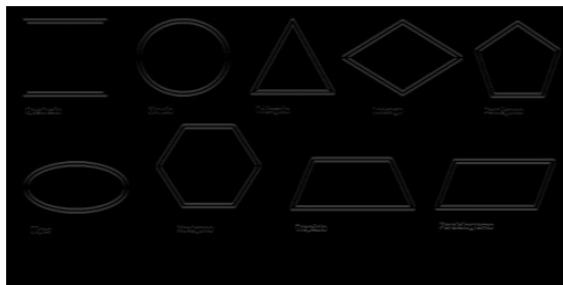


Figura 16: Imagen filtrada con filtro Recto 1D fila + columna

En esta imagen es más notoria la diferencia entre los filtros Prewitt debido a que son rectas y curvas sin mucho detalle, podría considerarse menos ruidosa. Se observa claramente que el primer filtro detecta los bordes verticales y el segundo detecta bordes horizontales, por ejemplo en el cuadrado para una imagen se ven solo los lados verticales y la segunda solo los horizontales.

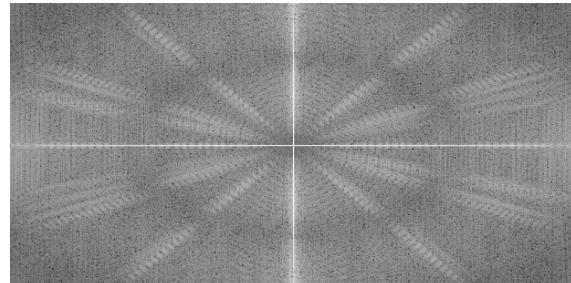
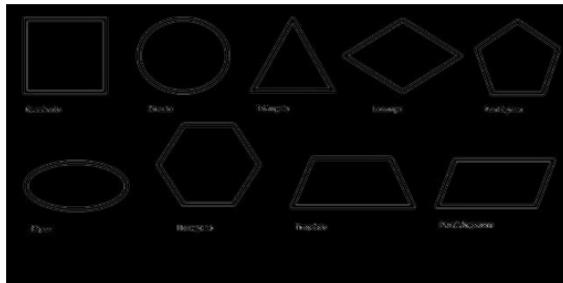


Figura 17: Imagen filtrada con filtro Laplaciano

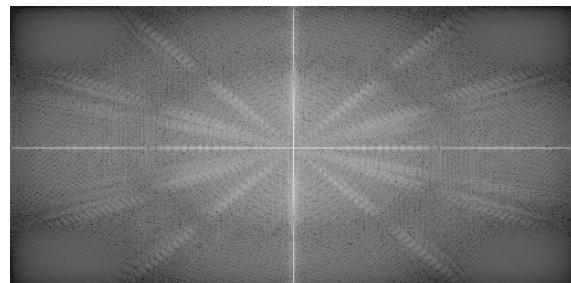
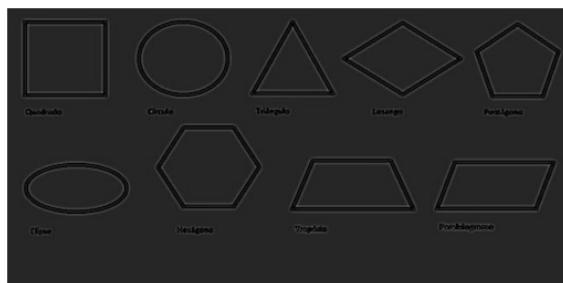


Figura 18: Imagen filtrada con filtro LoG

Aquí se confirma el hecho de que el filtro Laplaciano detecta bordes en todas direcciones, dado que la imagen no tiene mucho ruido se observan claramente los bordes de las figuras. A diferencia de los filtros Prewitt, el filtro Laplaciano rescata los contornos de las figuras.

En este caso debido a la baja presencia de ruido se observa que el efecto del filtro LoG es casi identico al filtro Laplaciano simple, detectando todos los bordes de las figuras.

3.2. Ecualización de Histogramas

3.2.1. Implementación

Para la implementación de histograma se implementó la función **ecualizar()** que recibe la imagen de entrada y salida, esta crea el arreglo *hist* en donde guardará la información del histograma por intensidad de brillo. Se recorre la imagen pixel a pixel utilizando 2 ciclos for y se van contando las veces que una intensidad de gris es detectada. Luego para crear la LUP se crear el arreglo *cumhist* donde se guardará el histograma acumulado y posterior a eso se procede a normalizar *cumhist* por la cantidad total de pixeles Finalmente se redefine la imagen recorriendo la imagen

pixel a pixel y usando su intensidad de gris como llave de la LUP, el valor detectado en la LUP se guarda en la posición de pixel correspondiente en la imagen de salida. A continuación se muestra el código donde se implementó la ecualización.

```
1 void ecualizar(Mat input, Mat output){  
2     float hist[256];  
3     for (int i=0; i<256; i++)  
4         hist[i] = 0;  
5     for (int r=0; r<input.rows; r++){  
6         for (int c=0; c<input.cols; c++){  
7             int ind = input.at<unsigned char>(r,c);  
8             hist[ind] = hist[ind]+1;  
9         }  
10    }  
11    float cumhist[256];  
12    cumhist[0]=hist[0];  
13    for (int i=1; i<256; i++)  
14        cumhist[i] = cumhist[i-1]+hist[i];  
15  
16    for (int i = 0; i < 256; i++)  
17        cumhist[i] = floor(cumhist[i]/double(input.rows * input.cols)  
* 255);  
18  
19    for (int r=0; r<input.rows; r++){  
20        for (int c=0; c<input.cols; c++)  
21            output.at<unsigned char>(r,c)=cumhist[input.at<unsigned char>(r,  
c)];  
22    }  
23    return;  
24 }
```

Código 4: Implementación de función de ecualizar

3.2.2. Resultados

A continuación se ven los resultados de aplicar la ecualización a distintas imágenes, comparando el original con el ecualizado.



(a) Original



(b) Ecualizada

Figura 19: Ecualización se imagen agua.png

Se observa que la ecualización ayuda considerablemente a la distinción de detalles en la imagen dada la mejor distribución de las intensidades de grises, sin embargo en torno al sol se pierden detalles dado que la imagen original era oscura y la ecualización desplazó el histograma hacia la derecha.



(a) Original



(b) Ecualizada

Figura 20: Ecualización se imagen casa.png

Se observa que aumenta el brillo de la imagen sin embargo el aumentar la dispersión del histograma provoca una gran diferencia entre los valores de algunos píxeles generando que la imagen se desigure marcando manchas en las partes mayormente oscura, marcando mucho la diferencia de intensidad.



(a) Original



(b) Ecualizada

Figura 21: Ecualización se imagen constr.png

Para el caso de la construcción se observa que debido a que la imagen tiene grandes concentraciones de intensidades bajas se aumente la intensidad promedio eliminando detalles den las partes

que están las luces de la construcción. Es tiene que ver con la característica contextual de la imagen dado que es de noche y se concentra la alta intensidad en los focos.



(a) Original



(b) Ecualizada

Figura 22: Ecualización se imagen corteza.png

Dado que la imagen de corteza presenta gran cantidad de detalle donde se aprovecha bien la distribución de intensidades no se observa un gran cambio.



(a) Original



(b) Ecualizada

Figura 23: Ecualización se imagen mala_ilum.jpg

Se observa que la ecualización logra aumentar la cantidad de detalles de la imagen dado que esta pareciera presentar un velo, este velo se traduce como un desplazamiento del histograma a la derecha dado que pareciera que todas las intensidades tienen sumado un valor fijo.



(a) Original



(b) Ecualizada

Figura 24: Ecualización se imagen termal.png

Dado que es una imagen termal se observa una gran concentración de intensidad de gris alrededor de la persona y zonas muy oscuras atrás de ella, luego de la ecualización en general la imagen se aclara debido a la gran concentración de negro en la imagen original, para este caso no se muestra gran aporte a los detalles.

4. Conclusiones

Se logró implementar exitosamente la función de convolución, la cuál fue utilizada para aplicar distintos filtros a las imágenes. Se puede concluir que los filtros pasa bajos del tipo rectangular y gaussianos tienen un efecto similar en la imagen, el cuál suaviza los bordes de la imagen. Por otro lado se confirma la propiedad que tiene la convolución de separación en el caso 2D, donde aplicar un filtro 2D o aplicar en secuencia un filtro 1D tienen un efecto bastante similar en la imagen resultante.

Respecto a los filtros pasa altos se concluye que estos logran detectar los bordes de la imagen, para el caso de los filtros Prewitt en una sola dirección y en el caso de los filtros Laplacianos en todas direcciones. Se observa que el ruido en la imagen puede generar falsos positivos, afectando la detección de bordes como es el caso de la corteza. Sin embargo el usar un filtro Gaussiano antes de Laplaciano puede eliminar el ruido y la detección de bordes mejora.

De la ecualización de histogramas se concluye que es un método efectivo en algunos casos para eliminar problemas de contraste, mostrando un mejor desempeño en imágenes que concentran el histograma de intensidad de gris en algún sector. Para caso en que el histograma tenga más de un sector de concentración se observa que se genera ruido o se pierden detalles en algunos sectores.