UNIVERSIDAD MILITAR NUEVA GRANADA

PROCESAMIENTO DE IMÁGENES EN EL ESPACIO

PAOLA JIMÉNEZ	DANIEL SALAZAR	DANIELA SILVA
(6000280)	(6000291)	(6000253)

1. INTRODUCCIÓN

En el siguiente informe se justificará y se profundizará de manera escrita la teoría de los diferentes temas mencionados en la materia ´Procesamiento de Imágenes basados en el trabajo práctico desarrollado durante el primer y segundo corte. Se indagará en los diferentes filtros desarrollados, y las especificaciones del código implementado en la parte práctica.

2. DISEÑO ORIENTADO A OBJETOS DE LA APLICACIÓN

Imagen 1. Diagrama de clases del programa.

PROCESAMIENTO DE IMÁGENES EN EL ESPACIO

Descripción del diagrama de clases:

A continuación se pretende destacar los aspectos más importantes de la superclase imagen con todos sus atributos y sus métodos:

2.1 ATRIBUTOS

- Variables de tipo **XYSeries:** Este es un tipo de variable de la librería open source JFreeChart la cual sirve para hacer gráficas de gran calidad en java, y este tipo de variable almacena una data de puntos xy para graficar. Este tipo de dato se usa para graficar el histograma y el histograma logarítmico.
- La variable de tipo **FILE** trae todos los atributos de un archivo o fichero y lo permite hacer operaciones con él.
- El tipo de dato **Buffer Image** es el tipo de dato que maneja java para operar con imágenes en un espacio de memoria lo más rápido posible, la variable original, es la imagen sin editar y la variable editada es la resultante después de aplicarle alguna funcionalidad del programa.
- **Ancho, Alto:** Guardan el tamaño de tipo entero de una imagen.
- Las variables de tipo color guardan los colores en rgb de la imagen a analizar y trabajar, el que es una matriz guarda todos los colores por pixel de la imagen en un arreglo y las variables de color normales son para capturar uno a uno los puntos de la matriz.
- **R,G,B:** Son enteros que nos ayudan a tomar los valores del color rgb por aparte.
- **rn,gn,bn,** son enteros que ayudan a calcular los colores del negativo de la imagen.
- **Promedio Intensidad:** esta variable guarda la intensidad o nivel de gris promedio de la imagen.
- **GrisMin,grimax:** sacan el gris mínimo y el gris máximo de la imagen.

2.2 Métodos

- **Los métodos setter y getter** son para el traspaso de información es decir para enviar y hacer recibir datos de la clase Imagen.
- **EscalaGrices():** Convierte la imagen a escala de grises.
- **NegativoIMG():** Convierte la imagen a su negativo.
- **filtroPasaAltos():** Le pone a la imagen un filtro Pasa altos.
- **filtroPasaBajos():** Le pone a la imagen un filtro Pasa bajo.
- **filtroHBPasaAltos():** Le pone a la imagen un filtro high boost a la imagen partiendo del filtro pasaAltos.
- **filtroHBPasaBajos():** Le pone a la imagen un filtro high boost a la imagen partiendo del filtro pasaBajos.
- **filtroPascal():** Trabaja con el triángulo de pascal y dependiendo del nivel del triángulo que se desee el programa realiza una matriz para aplicarla a la imagen.

- **TrianguloPascal():** Se le envía como parámetro el número del nivel del triángulo de pascal que se desee y calcula un arreglo lleno con los datos de los números de este piso de pascal en específico.
- **CalcularMatriz():** Recibe el arreglo de números con el nivel del triángulo de pascal y hace una matriz con esos números.
- **filtroMaximo():** Le coloca un filtro máximo la imagen.
- **filtroMinimo():** Le coloca un filtro mínimo la imagen.
- **OrdenarMM():** Ordena la matriz que se le envíe, y al ordenarla saca tres valores de posible retorno, el mínimo del arreglo, el máximo del arreglo y el intermedio del arreglo que sería la mediana.
- **Histograma():** Grafica el histograma de la imagen.
- **HistogramaLogaritmico()**: Grafica el histograma en logaritmo base 10 de la imagen.
- **grisMaxyMin():** Calcula el nivel de gris mínimo y máximo de la imagen.
- **PromedioIntesidad():** retorna el nivel de gris promedio de la imagen.
- **ContrasteAuto():** hace el contraste automático de la imagen.
- **Ecualizacion():** Hace la ecualización del histograma para mejorar el contraste de la imagen.
- **filtroPersonalizado():** Toma valores de una arreglo que está en la interfaz gráfica y partiendo de esos campos le aplica esa matriz a la imagen, la matriz admite solo hasta 3x3 por el momento.
- **FiltroMediana():** Obtiene el valor de la mediana de la matriz con 3 vecinos cercanos gracias al método OrdenarMM() y retorno la mediana de esa matriz.
- **FiltroModa():** Le aplica un filtro Moda a la imagen. hace uso del método ObtenerModa() para sacar la moda de una matriz 3x3 que se le envíe.
- **ObtenerModa():** Mediante algunos cálculos, sacando la frecuencia de cada número dentro de la matriz que se le envía a este método(que tiene que ser 3x3) obtiene la moda de esa matriz.
- **CrearImagenGrises():** Se le pasa como parámetro un arreglo de enteros con los niveles de gris de la imagen por pixel y crea una imagen en escala de grises.
- **CrearImagenColor():** recibe como parámetro un arreglo de colores de la imagen por pixel y crea una imagen a color.

3. MARCO TEÓRICO

3.1 APERTURA DE ARCHIVOS Y GRAFICACIÓN DE INFORMACIÓN DE IMÁGENES

3.1.1 ABRIR IMAGEN

Debido a que en el programa se usó el lenguaje de programación java , para abrir el archivo hay distintas maneras de hacerlo y diferentes objetos para hacerlo, en esta ocasión se quiere hablar de la opción usada en el programa que fue una más gráfica, y en donde el usuario pudiera elegir el archivo que queria abrir. Se usó el Objeto de java llamado JFileChooser para cargar la ruta donde se encuentra el archivo a abrir, en este caso solo sirve para archivos jpg y png, la funcionalidad de este objeto es guardar la ruta seleccionada para poder pasarla a una archivo de tipo File que también es de java, el cual si tiene la posibilidad de abrir el archivo elegido en nuestro aplicativo.

Diseño Lógico del algoritmo

```
 //Se crea el objeto JFileChooser llamado selecion.

       seleccion.setDialogTitle("Seleccione una imagen");
//se crea un File para almacenar la imagen
       File imagen;
       String ruta;
//si el archivo abre
       si el archivo abre entonces{
          imagen = a la ruta de la imagen
//lector de imagen de java
             original = ImageIO.read(imagen);
// se crea el objeto imagen img y se le manda el archivo y el buffer
             img = new Imagen(imagen,original);
             this.lblImagen.setIcon(new ImageIcon(img.getOriginal()));
       }
```

Resultados

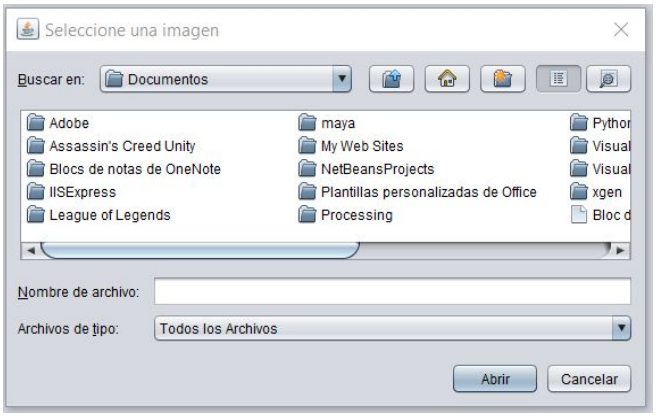


Imagen 1. Abrir imagen

Para seleccionar la imagen que se desea, da la opción de buscar entre las diferentes carpetas, y dar “click” al boton de abrir.

3.1.2 GUARDAR IMAGEN

Para guardar la imagen se hace de una manera muy similar a la de abrirla, se hacen los objetos necesarios como lo son el File y el BufferImage, simplemente ponemos el escritor de imágenes de java y pasamos como parámetro la ruta File de destino y un buffer image para mandarle la info a esa imagen.

```
 //Se crea el objeto JFileChooser llamado guardar.

       guardar.setDialogTitle("");
//se crea un File para almacenar la imagen
       File imagen;
       String ruta;
//si el archivo abre
       si el archivo abre entonces{
          imagen = a la ruta de destino
//escritor de imagen de java
             Editado = ImageIO.write(imagen);
       }
```

Resultados

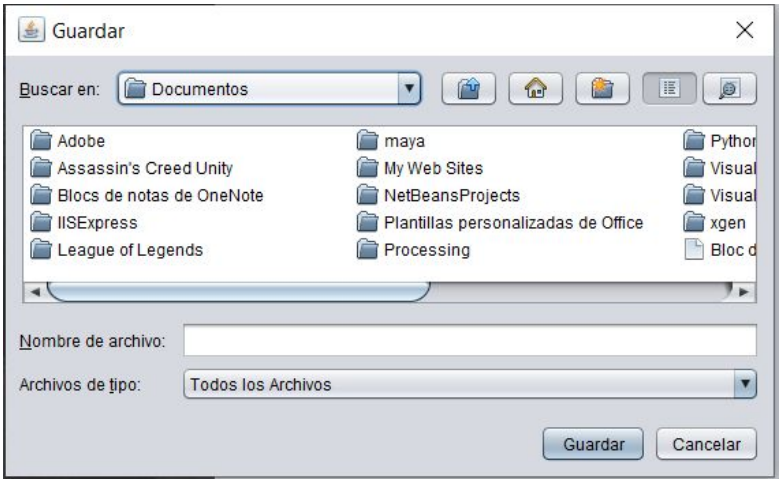


Imagen 2. Guardar imagen

Se da la opción de guardar la imagen después de usar el filtro que se desea, en la cual se puede nombrar sin problemas.

3.1.3 GRAFICAR IMAGEN

Para graficar la imagen se hace mediante un objeto de java llamado BufferImage y un JLabel o etiqueta. El BufferImage es un espacio de memoria intermedio entre la aplicaion de nosotros, y la ruta o ubicación del archivo, este tipo de dato hace que las operaciones con imágenes se lleven a cabo a altísimas velocidades, mejorando la eficiencia del programa. los archivos JLabel tiene una propiedad en su interior llamada icono, este nos permite guardar imágenes en etiquetas, para cargar la imagen y visualizarla en pantalla se hace a través de esta propiedad y el traspaso de un BufferImage el cual se va a transformar en la nueva imagen de la etiqueta.

Diseño Lógico del algoritmo

```
Editada = new BufferedImage(ancho, alto,original.getType());
//Se hace un nuevo bufferImage y se le asigna un ancho, un alto , y un tipo para saber el tipo de color
      hacer i=0 hasta ancho
         hacer i=0 hasta alto
            newrgb = arrayImagen[x][y] << 16 | arrayImagen[x][y] << 8 | arrayImagen[x][y];
//se pasa el color a un número de bytes
//luego le pasamos ese color hecho byte como color de la imagen
            Editada.setRGB(x, y, new Color(newrgb).getRGB());
         }
      }
```

Resultados

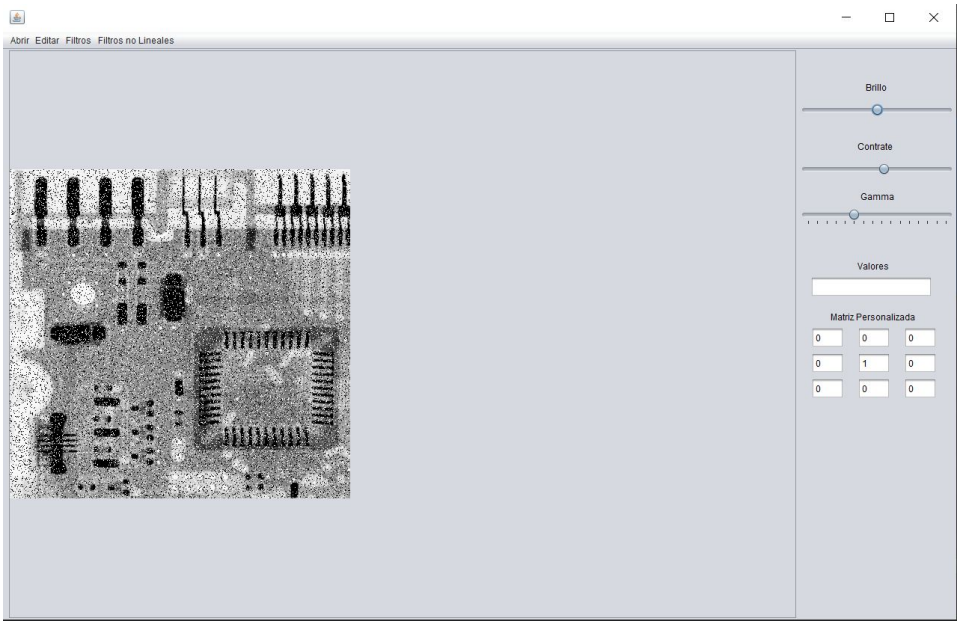


Imagen 3. Graficar Imagen

Al graficar la imagen se puede ver la interfaz del editor de imágenes con sus diferentes opciones, brillo, contraste, gamma etc. El tamaño que se puede ver al graficar la imagen depende de su dimensión.

3.1.4 HISTOGRAMA

El objetivo del histograma es enseñar una gráfica donde se ve el número de píxeles, nk, de cada nivel de gris, rk, que se muestra en la imagen.

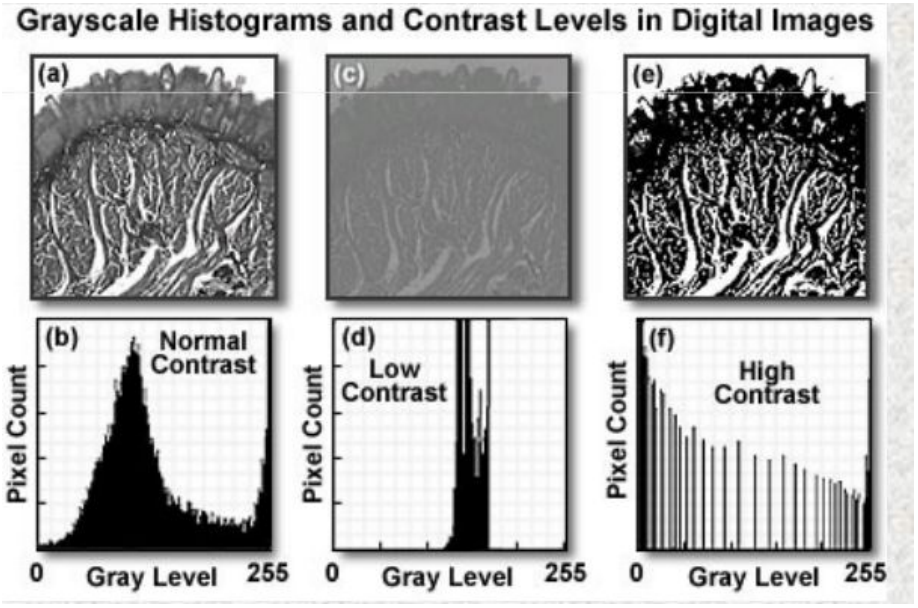


Imagen 4. Imágenes con sus respectivos histogramas

El histograma es útil ya que con esta se puede comparar el contraste y la intensidad entre imágenes; con el histograma al alterarlo, puede producir cambios en la imagen.

Cuando en el histograma se puede ver que el rango de niveles de gris tomada de la imagen está concentrada en una región del intervalo, quiere decir que dicha imagen posee poco contraste. Para poder aumentar el contraste de una imagen, lo que se puede hacer es expandir el histograma o en su defecto, ecualización del mismo.

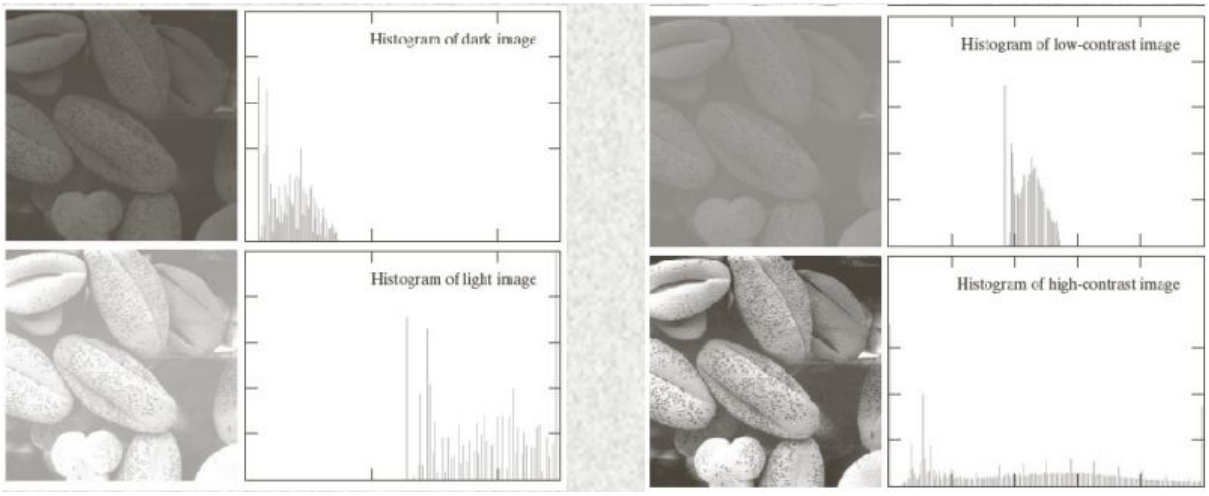


Imagen 5. El contraste reflejado en el histograma

Diseño Lógico del Algoritmo:

Se pasa la imagen a escala de grises

```
        frecuencia=0;
        aux=0;
        nivelGrises = números del 0 al 255
// arreglo que va a almacenar los valores de las frecuencias de cada nivel de gris
        frecuenciaGrices = int[256]

       hacer i=0 hasta tamaño del arreglo de nivelGrises
           hacer a=0 hasta ancho
              hacer b=0 hasta alto
                 si valorGrisOriginal[a][b] es igual a nivelGrises[i]
                     frecuencia++
                 }
              }
           }
           frecuenciaGrices[n]=frecuencia
           frecuencia=0
           punticos.add(nivelGrices[n],frecuenciaGrices[n])
        }

//para el resto se uso una libreria llamada JFreeChart para graficar los valores anteriores

     Para mas info:http://www.jfree.org/jfreechart/
```

Resultados

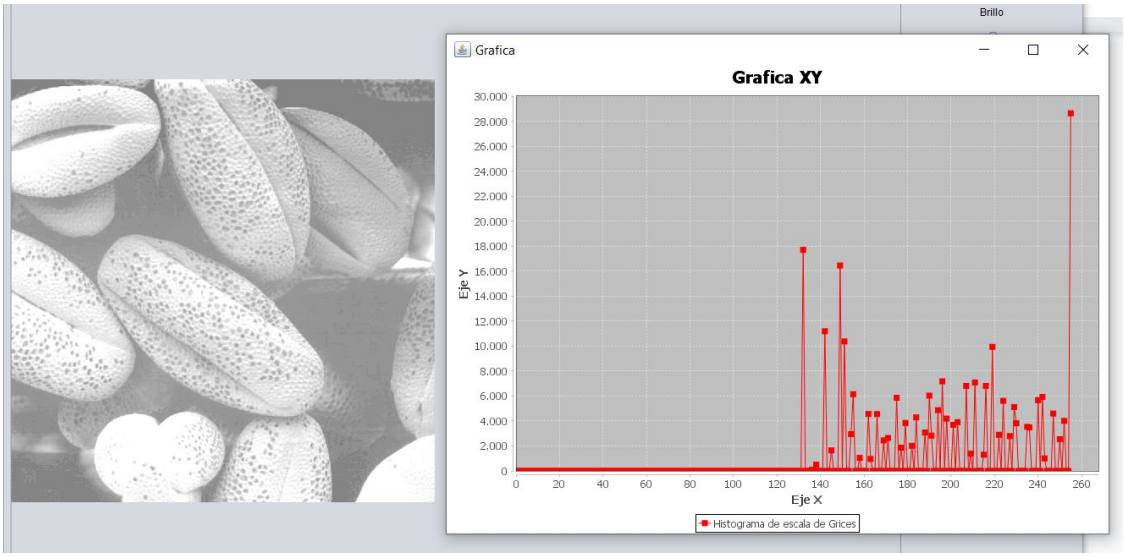


Imagen 6. Histograma

Se puede evidenciar el histograma normalizado basado en la imagen, en el que se puede notar una concentración de intensidad de grises al lado derecho, lo que se puede concluir que predominan los niveles de grises inclinados hacia el blanco y carece del negro. También se puede ver que el nivel de gris que más se repite es el último nivel.

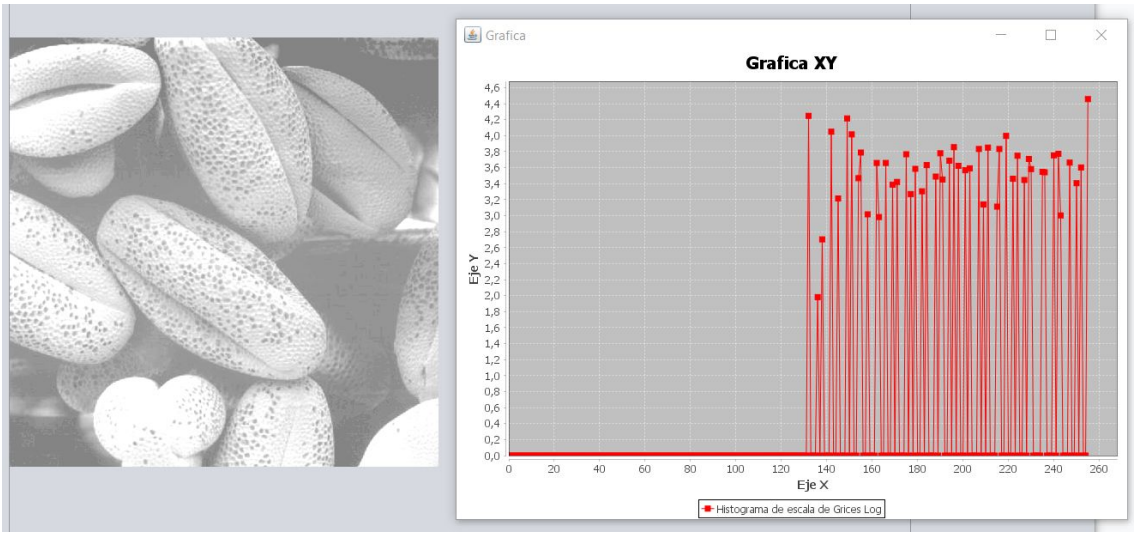


Imagen 7. Histograma visualización semilogarítmica

Se puede observar el histograma en base 10, la cual es el reflejo de la imagen que está a su lado izquierdo, sin embargo, su visualización pasa de ser normal a ser semi logarítmica lo cual se puede observar mejor dicho histograma.

3.2 MEJORA DE IMÁGENES

3.2.1 BRILLO

El brillo es la intensidad de la luz en cada uno de los pixeles, al observar una imagen con variaciones en el brillo, esta se verá más clara o más oscura de acuerdo a qué tanto brillo tiene.

Para ajustar el brillo, se puede modificar el histograma, sumando o restando una constante a todos los valores de la imagen dependiendo al resultado que se desea.

$$R(x,y)=[(x,y)+k]$$

La imagen resultante = Imagen Original + Constante
K puede ser positivo o negativo, según sea el caso.

Diseño Lógico del algoritmo

```
rgbOriginal = matriz de colores de la imagen
grises = matriz de grises de la imagen

hacer i=0 hasta ancho

    hacer j=0 hasta alto
//obtener el rgb pixel a pixel
        pixel = original.getRGB(x, y);
//crearlo como color nuevo
        c = new Color(pixel);
        rgbOriginal[x][y] = c;
//guardar los valores r g y b para operarlos
        r = c.getRed();
        g = c.getGreen();
        b = c.getBlue();
//el gris va a ser igual al 30% del rojo, 59% del verde y 11% del azul
        valorGrisOriginal[x][y] = (int) ((r * 0.3) + (g * 0.59) + (b * 0.11))+ brillo que le quieras poner;
        si valorGrisOriginal[x][y]  es menor que 0
           valorGrisOriginal =0
        }
        si valorGrisOriginal[x][y] es mayor que 255
           valorGrisOriginal =255
        }
    }
}

Se llama a la función CrearImagenGris(valorGrisOriginal).
```

Resultados

Imagen 8. Original

Imagen 9. Con brillo

Al comparar la imagen original con el resultante de una imagen aplicando brillo, se puede ver como se ve más clara, usando brillo a 120.

3.2.2 CONTRASTE

Lo que se hace es alterar el nivel de gris que corresponde a la transformación lineal por medio de rectas, el proceso lo que hace es cambiar el intervalo de los valores de intensidad del píxel.El proceso ocasiona en la imagen el estiramiento o contracción del histograma . A este proceso también se le llama normalización o como rango dinámico. Lo que sucede es que se obtiene el nivel de gris máximo y el nivel de gris mínimo, para obtener el nuevo rango. Y se evalúa cada pixel para obtener el nuevo gris del pixel.

$$I_N = (I - \text{Min})\frac{\text{newMax} - \text{newMin}}{\text{Max} - \text{Min}} + \text{newMin}$$

Imagen. Ecuación de contraste

PROCESAMIENTO DE IMÁGENES EN EL ESPACIO

Diseño Logico del algoritmo:

```
 rgbOriginal = matriz de colores de la imagen
       grises = matriz de grises de la imagen

       hacer i=0 hasta ancho
          hacer j=0 hasta alto
//obtener el rgb pixel a pixel
             pixel = original.getRGB(x, y);
//crearlo como color nuevo
             c = new Color(pixel);
             rgbOriginal[x][y] = c;
//guardar los valores r g y b para operarlos
             r = c.getRed();
             g = c.getGreen();
             b = c.getBlue();
//el gris va a ser igual al 30% del rojo, 59% del verde y 11% del azul
             valorGrisOriginal[x][y] = (int) ((r * 0.3) + (g * 0.59) + (b * 0.11))*valor de contraste floatr;
             si valorGrisOriginal[x][y]  es menor que 0
                valorGrisOriginal =0
             }
             si valorGrisOriginal[x][y] es mayor que 255
                valorGrisOriginal =255
             }
          }
       }

       Se llama a la función CrearImagenGris(valorGrisOriginal).
```

Resultados

Imagen 8. Original

Imagen 10. Con contraste

Se puede evidenciar como en la imagen resultante al usar un contraste de 2, se intensifica ciertos aspectos de la imagen como los cráteres de la imagen pasando a la oscuridad

3.2.3 CONTRASTE AUTOMÁTICO

El contraste automático es una técnica que opera pixel a pixel una fórmula la cual tiene como objetivo mejorar la calidez visual de la imagen, mejorando significativamente el contraste de la imagen, reduciendo un poco el ruido y mejorando lo que se conoce en imágenes como la reflectancia y la iluminancia, esta técnica nos ayuda en gran manera ya que al funcionar de manera automática sirve en ocasiones para ajustes rápidos de contraste de una imagen con un resultado bastante bueno como punto de partida en la edición de una imagen.

la ecuación que opera pixel a pixel la imagen es la siguiente:

$$R(x,y) = \frac{(I(x,y) - I_{min})}{I_{max} - I_{min}} * (L-1)$$

Imagen 11. Ecuación de contraste automático

Para esta ecuación es necesario tener los niveles de gris minimo y maximo de la imagen, y aplicando esta función pixel a pixel se logra una mejora significativa del contraste.

Diseño Lógico del algoritmo:

```
rgbOriginal = matriz de colores de la imagen
      grises = matriz de grises de la imagen
      grisMin = nivel de  gris mínimo de la imagen
      grisMax = nivel de  gris máximo de la imagen

      hacer i=0 hasta ancho
         hacer j=0 hasta alto
//obtener el rgb pixel a pixel
            pixel = original.getRGB(x, y);
//crearlo como color nuevo
            c = new Color(pixel);
            rgbOriginal[x][y] = c

//guardar los valores r g y b para operarlos
            r = c.getRed()
            g = c.getGreen()
            b = c.getBlue()

//el gris va a ser igual al 30% del rojo, 59% del verde y 11% del azul
            valorGrisOriginal[x][y] = (int) ((r * 0.3) + (g * 0.59) + (b * 0.11))

//aplicar ecuacion de contraste auto

            valorGrisOriginal[x][y]= (valorGrisOriginal[x][y]-grisMin)/(grisMax-grisMin) *255

            si valorGrisOriginal[x][y]  es menor que 0
               valorGrisOriginal =0
            }
            si valorGrisOriginal[x][y] es mayor que 255
```

```
            valorGrisOriginal =255
        }
    }
}
```

Se llama a la función CrearImagenGris(valorGrisOriginal).

Resultados

Imagen 12 Original

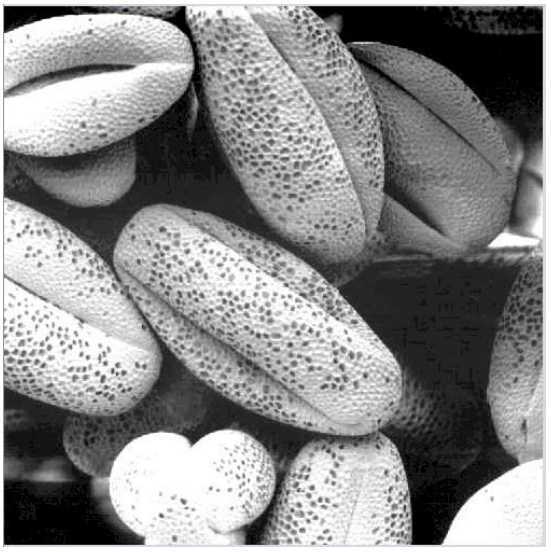


Imagen 13. Contraste Automático

Se puede evidenciar como la imagen original pasa de tener un alto brillo, y bajo contraste a una imagen en la que se puede diferenciar los diferentes componentes de esta.

3.2.4 ECUALIZACIÓN

El objetivo de la ecualización es mejorar el contraste, podrá poner los valores del histograma de manera uniforme

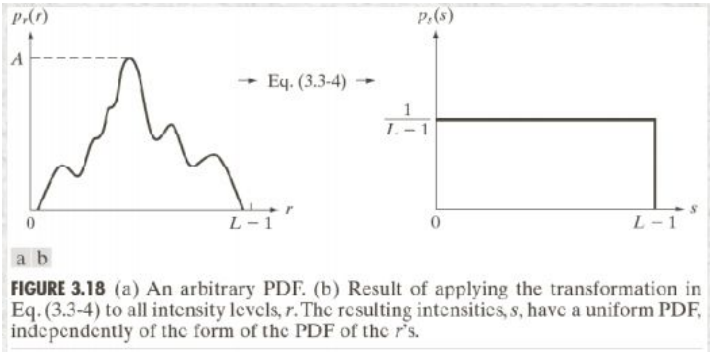


FIGURE 3.18 (a) An arbitrary PDF. (b) Result of applying the transformation in Eq. (3.3-4) to all intensity levels, r. The resulting intensities, s, have a uniform PDF, independently of the form of the PDF of the r's.

Imagen 14. Ejemplo ecualización.

Una imagen con dimensiones MxN, con nk pixeles para cada nivel rk, se podrá usar una transformación que realiza dicha actualización sobre los niveles de intensidad de grises:

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) = \frac{L-1}{MN}\sum_{j=0}^{k} n_j$$

Imagen 15. Transformación de Ecualización

Da como resultado una dispersión del histograma con un intervalo entre [0, L-1].

```
Se pasa la imagen a escala de grises

      frecuencia=0;
      aux=0;
      nivelGrises = números del 0 al 255
// arreglo que va a almacenar los valores de las frecuencias de cada nivel de gris
      frecuenciaGrices = int[256]
// hacer un arreglo que guarde la frecuencias acumuladas
      probabilidadesAcum = int[256]
// hacer un arreglo para guardar los grises ecualizados
      grisesEcualizados = int[256]
     hacer i=0 hasta tamaño del arreglo de nivelGrises
        hacer a=0 hasta ancho
           hacer b=0 hasta alto
             si valorGrisOriginal[a][b] es igual a nivelGrises[i]
                 frecuencia++
              }
           }
        }
        frecuenciaGrices[i]=frecuencia; //llenamos el arreglo de frecuencias
        probabilidades[i]=frecuenciaGrices[i]/(ancho*alto)
        aux+=probabilidades[i];
        probabilidadesAcum[i] = aux;
        float nuevoGris = (float)probabilidadesAcum[i]*255;
        grisesEcualizados[i] =(int)nuevoGris;
        frecuencia=0;
        grises[a][b] = gricesEcualizados[i];

      }

Se llama a la función CrearImagenGris(valorGrisOriginal).
```

Resultados

Imagen 12. Original

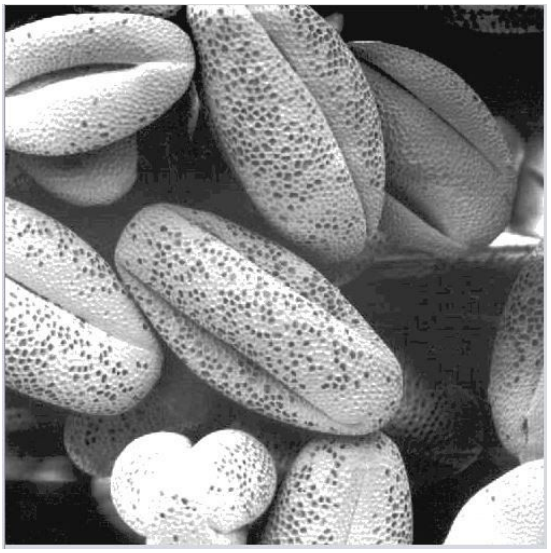


Imagen 16. Ecualización

Se visualiza la mejora de contraste de la imagen resultante a partir de la imagen original, se puede pensar que si se ve su histograma, sus niveles de grises están mejor distribuidos.

3.2.5 GAMMA: La corrección gamma es la operación no lineal que se utiliza para codificar o decodificar luminancia. La codificación gama es necesaria para compensar ciertas propiedades de la visión humana, y de cómo nosotros percibimos el color y la luz. La corrección se obtiene mediante la siguiente ecuación:

$$V_{\text{out}} = AV_{\text{in}}{}^{\gamma}$$

Imagen 17. ***Ecuación de corrección gamma***

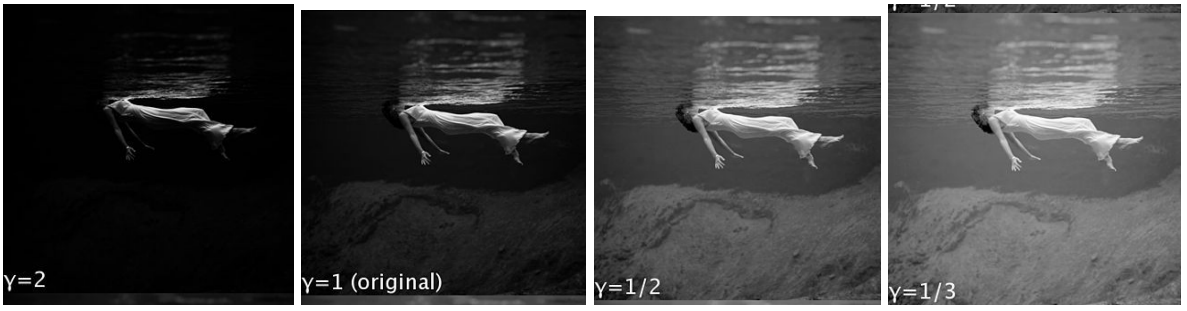


Imagen 18. Corrección gamma con diferentes valores.

Diseño Logico del algoritmo:

```
 rgbOriginal = matriz de colores de la imagen
       grises = matriz de grises de la imagen

       hacer i=0 hasta ancho
          hacer j=0 hasta alto
//obtener el rgb pixel a pixel
             pixel = original.getRGB(x, y);
//crearlo como color nuevo
             c = new Color(pixel);
             rgbOriginal[x][y] = c;
//guardar los valores r g y b para operarlos
             r = c.getRed();
             g = c.getGreen();
             b = c.getBlue();
//el gris va a ser igual al 30% del rojo, 59% del verde y 11% del azul
             valorGrisOriginal[x][y] = (int) ((r * 0.3) + (g * 0.59) + (b * 0.11)) ^ valor de Gamma float;
             si valorGrisOriginal[x][y]  es menor que 0
                valorGrisOriginal =0
             }
             si valorGrisOriginal[x][y] es mayor que 255
                valorGrisOriginal =255
             }
          }
       }

       Se llama a la función CrearImagenGris(valorGrisOriginal).
```

Resultados

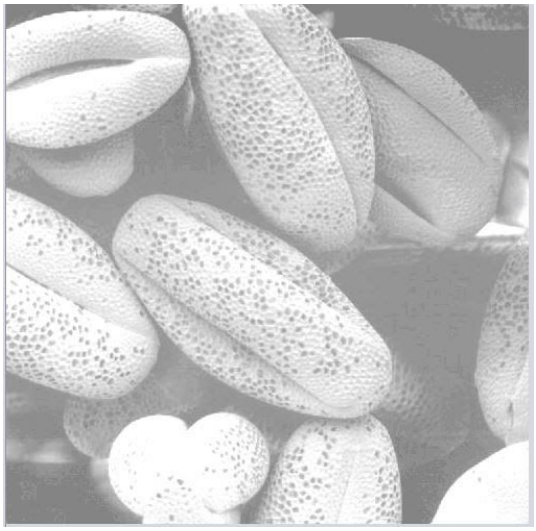


Imagen 12. Original

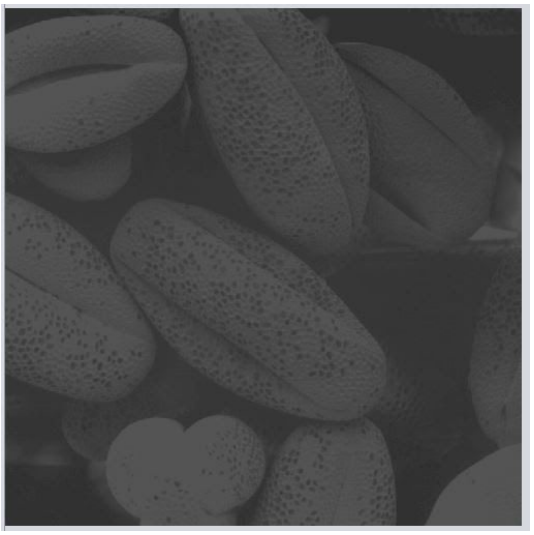


Imagen 19. Contraste Automático

Se evidencia el uso de gamma de la imagen resultante con y=0.4, en la cual dicha imagen se pone más opaca, visualizando mejor esta.

3.1 FILTROS LINEALES

Los filtros lineales son filtros basados en máscaras de convolución, estas se caracterizan por su resultado, cada pixel que se obtiene es por la combinación lineal de sus vecinos.

3.1.1 PASABAJOS PROMEDIO 3X3

El filtro de pasa-bajos consiste en buscar atenuar las frecuencias altas sin modificar las frecuencias bajas. De acuerdo a la función del presente filtro, se puede asimilar a un filtro de suavizado ya que sus altas frecuencias al ser filtradas corresponden cambios bruscos con respecto a la intensidad.

Sus principales características es que son útiles para reducir el ruido de alta frecuencia, o eliminar todo lo que no sean variaciones suaves de nivel de gris.

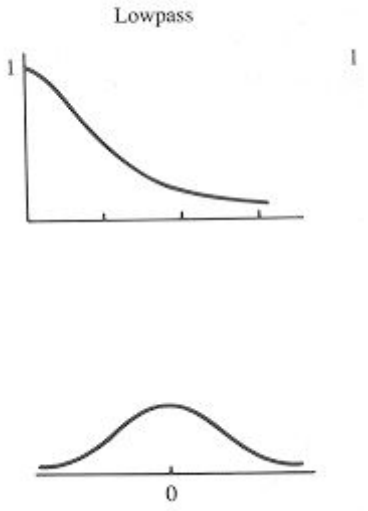


Imagen 20.Filtro en el dominio de la frecuencia y espacio.

Un ejemplo muy utilizado de filtro pasa-bajo es aquel que tiene una máscara de convolución con dimensión 3x3 y que sus nueve coeficientes son igual a 1/9.

Un filtro de paso bajo muy utilizado es aquel cuya máscara de convolución tiene dimensión 3 x 3 y sus nueve coeficientes son iguales a 1/9, es decir:

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Matriz 1. Matriz del filtro pasa- bajo. Matriz 3x3

Lo que este filtro hace es un promedio de los valores de brillo de los pixeles, el cual se conoce como filtro de la media. Consiste en que la suma de sus respectivos coeficientes es igual a 1 (que son positivos).

Si el filtro es aplicada en una área o región de una imagen en el que cada pixel del núcleo tiene el mismo valor de brillo, el resultado es el mismo valor de brillo. Si, por el contrario dicho filtro se aplica en una área donde los valores de brillo de los píxeles cambian rápidamente del negro a blanco y a la inversa, quiere decir, el área de una frecuencia alta espacial, el resultado será un valor medio de gris entre el negro y blanco.

Todo lo mencionado anteriormente produce una imagen de salida compuesta con un resultado de valores medio de gris que varían de una forma muy leve.

Diseño Lógico del algoritmo:

Se llama la función EscalaGrises() para volver la imagen gris
Se declara una matriz para recoger los pixeles en gris

```
valorGrisEditado = new int[ancho][alto];
valorPromedio=0;
tol=1;
cont =0;
Hacer i=0 hasta ancho
   hacer j=0 hasta alto
     hacer m=i-tol hasta i+tol
         hacer n=j-tol hasta j+tol
            valorPromedio = valorPromedio+valorGrisOriginal[m][n]
            cont++;
    }
      }
      valorPromedio = valorPromedio/cont
      valorGrisEditado[i][j] = valorPromedio
      valorPromedio=0;
      cont=0;
   }
}
```

Se llama a la función CrearImagenGris(valorGrisEditado).

Resultados

Imagen 8. Original ***Imagen 21.Con filtro de Pasa Bajos***

Se puede observar en la imagen como el filtro pasa-bajos hace su efecto, como la imagen original se ve perfectamente y pasa a ser una imagen con un evidente filtro de suavizado haciendo que se borre los detalles más finos de dicha imagen.

3.1.2 PASAALTOS 3X3

El filtro de pasa-altos consiste en buscar atenuar las frecuencias bajas sin modificar las frecuencias altas. Este filtro se usa más que todo para quedarse con las propiedades de la imagen en el que los niveles de grises varían de forma brusca, por bordes de la imagen.

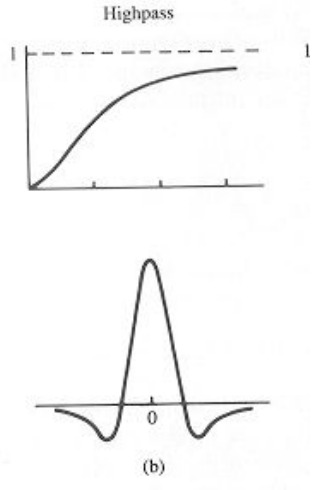


Imagen 22.Filtro en el dominio de la frecuencia y espacio.

Una máscara muy común de pasa-altos, de una dimensión 3x3, es la que tiene un 9 en la posición del centro y un -1 como la vecindad de dicho centro es decir:

-1	-1	-1
-1	9	-1
-1	-1	-1

Matriz 2. Matriz del filtro pasa- alto. Matriz 3x3

En esta matriz, la suma de los coeficientes es 1, y el coeficiente más grande y positivo que es el centro, está rodeado por coeficientes pequeños, esto quiere decir que el pixel central de dicho grupo de pixeles en entrada al procesar tienen una alta influencia por el contrario, los pixeles que están en su alrededor actúan de forma que se oponen a este. Si, dicho píxel central contiene un valor de brillo completamente diferente al de sus vecinos, entonces el efecto es despreciable y el valor de salida es una versión acentuada del valor original del píxel central. La diferencia significativa muestra una transición en los niveles de gris el cual indica la presencia de los componentes que contiene frecuencias altas. Esto quiere decir que la imagen de salida se evidencie la transición de forma acentuada, mencionada anteriormente.

Si sucede de manera inversa, es decir, los valores del brillo de la vecindad son suficientemente grandes para contrarrestar el “peso” del pixel central, su resultado es un promedio de los pixeles involucrados. Si el valor del brillo de todos los píxeles de la matriz son iguales, el resultado será el mismo.

A continuación se mostrará una imagen original y cuando pasa por filtro pasa-altos:

UNIVERSIDAD MILITAR NUEVA GRANADA

PROCESAMIENTO DE IMÁGENES EN EL ESPACIO

Diseño lógico del algoritmo:

Se llama la función EscalaGrises() para volver la imagen gris.
Se declara una matriz para recoger los pixeles en gris que se llama grises 2.
Se inicializa la variable tol a 1.
Se inicializa la variable cont a 0.
Hacer de i=0 hasta i menor que el ancho .
Hacer de j=0 hasta j menor que el alto.
Hacer de m=0 hasta m menor que el alto.
Hacer de n=0 hasta n menor que el alto.
Y que a cada posición de la matriz se vaya sumando al el valor promedio.

}
}
Ahora con el valor promedio se va a pasar el arreglo y se le resta el valor promedio para luego multiplicarlo por 5 .
}
Después se imprime la imagen de grises inicializando las variables de val promedio y cont.
}
}
Se llama a la función crear gris

Resultados

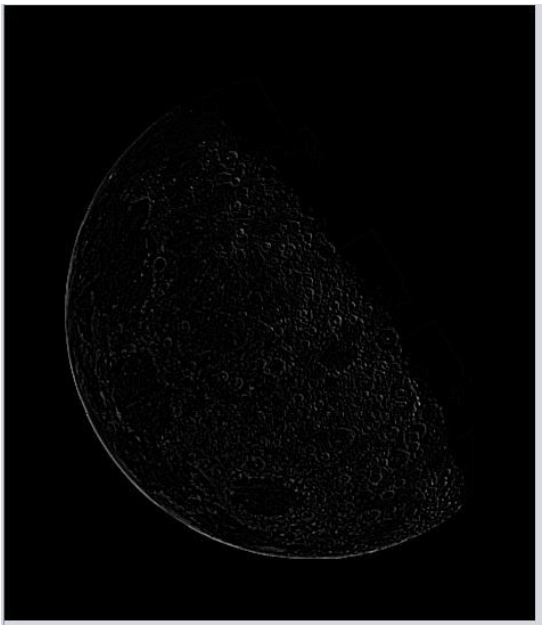


Imagen 8. ***Original*** ***Imagen 23.Con filtro de Pasa Altos***

Al comparar la imagen original con la imagen usando el filtro pasa-altos, se puede evidenciar el efecto de realce en sus detalles más finos, mostrando así el borde de la luna y los diferentes cráteres de esta.

3.1.3 HIGH-BOOST 3X3

High-boost también llamada ´filtro para énfasis de altas frecuencias´, es usado para resaltar los contrastes sin oscurecer la imagen resultante. Para calcular una imagen filtrada High-boost, se debe remontar a cálculo de un filtro pasa-altos que es la resta entre la imagen original y una versión pasada por un filtro pasa-bajos:

$$Pasa\ Alto = Original - Pasa\ Bajo$$

Si la operación para hallar un filtro pasa altos se multiplica por un factor de amplificación A, se obtiene el filtro high-boost:

$$\begin{aligned} High\ Boost &= A \times Original - Pasa\ Bajo \\ &= (A-1) \times Original + Original - Pasa\ Bajo \\ &= (A-1) \times Original + Pasa\ Alto \end{aligned}$$

Si se aplica el valor A=1, da como resultado un filtro pasa alto en su normalidad.
Si se aplica un valor A>1, una fracción de la imagen original se añade al resultado del filtro pasa alto y por lo tanto devuelve de forma parcial las bajas frecuencias. La imagen resultante high-boost se parece más a la imagen original con un nivel de mejora de los bordes y este depende del valor de A.

PROCESAMIENTO DE IMÁGENES EN EL ESPACIO

A continuación se mostrará una máscara 3 x 3 para filtrado high boost:

1/9 X

-1	-1	-1
-1	w	-1
-1	-1	-1

Matriz 3. Matriz del filtro High-boost. Matriz 3x3

En el que w=9 y A>=1

Diseño Lógico del algoritmo:

```
se llama la función EscalGrices() para volver la imagen gris
se declara una matriz para recoger los pixeles en gris

valorGrisEditado = new int[ancho][alto]
valorPromedio=0
tol=1
cont =0
MultiplicadorA=2
hacer i=0 hasta ancho
   hacer j=0 hasta alto
      hacer m=i-tol hasta m+tol
         hacer m=i-tol hasta m+tol
            valorPromedio = valorPromedio+valorGrisOriginal[m][n]
            cont++
         }
      }
      valorPromedio=valorPromedio/cont
      valorGrisEditado[i][j]=(MultiplicadorA*valorGrisOriginal[i][j])- valorPromedio
      si valorGrisEditado[i][j] es menor que 0
         valorGrisEditado[i][j]=0
      }
      si valorGrisEditado[i][j] es mayor que 255
         valorGrisEditado[i][j]=255
      }
      valorPromedio=0
      cont=0
   }
}

Se llama a la función CrearImagenGris(valorGrisEditado).
```

Resultados:

Imagen 12. Original **Imagen 24. Con High Boost**

Al usar el filtro de high-boost se puede notar que la imagen tiene un efecto de resaltar los contrastes sin la necesidad de oscurecer dicha imagen, por lo tanto se ve más definida que la original.

3.1.4 FILTRO BINOMIAL DE TAMAÑO 7X7

El filtro binomial se basa en la construcción de una convolución que produce una respuesta de frecuencia, esta se caracteriza más que todo por la máscara que es formada o depende de la función binomial, expresada de esta forma:

$$f_N(x) = \binom{N}{x} = \frac{N!}{x!\,(N-x)!}$$

Fórmula Binomial

N indica el orden del filtro, para todo valor de x puede ser x=0, 1,…..N.

Este filtro se usa más que todo para suavizar y eliminar el ruido de una imagen.
Su mayor uso se enmarca en el suavizado y la eliminación de ruido dentro de una imagen.

$$1/4096 \times \begin{pmatrix} 1 & 6 & 15 & 20 & 15 & 6 & 1 \\ 6 & 36 & 90 & 120 & 90 & 36 & 6 \\ 15 & 90 & 225 & 300 & 225 & 90 & 15 \\ 20 & 120 & 300 & 400 & 300 & 120 & 20 \\ 15 & 90 & 225 & 300 & 225 & 90 & 15 \\ 6 & 36 & 90 & 120 & 90 & 36 & 6 \\ 1 & 6 & 15 & 20 & 15 & 6 & 1 \end{pmatrix}$$

Matriz 4. Matriz del filtro binomial. Matriz 7x7.

Diseño Lógico del algoritmo:

Se llama la función EscalGrices() para volver la imagen gris
Se declara una matriz para recoger los pixeles en gris

valorPromedio=0
numPisosPascal=7

Se llama a la función TrianguloPascal(numPisospascal) para hacer el triangulo de pascal hasta el piso parametro
se llama a la función CalcularMatriz(numPisospascal) para hacer hacer la matriz con los valores del triangulo de pascal

matrizFiltro = matriz con los valores
sumaMatriz= suma de valores de la matriz de pascal
tol=numPisosPascal/2
cont =0
contM=0
contN=0

```
hacer i=0 hasta ancho
   hacer i=0 hasta alto
      hacer m=i-tol hasta m=i+tol
         hacer n=j-tol hasta n+tol
            valorPromedio = (matrizFiltro[contM][contN]*valorGrisOriginal[m][n])+valorPromedio;
            cont++;
            contM++;
         }
         contM=0;
         contN++;
      }
      valorPromedio=valorPromedio/sumaMatriz;
      valorGrisEditado[i][j]=valorPromedio;
      valorPromedio=0;
      cont=0;
```

```
        contN=0;
        contM=0;
    }
}
```

Se llama a la función CrearImagenGris(valorGrisEditado)

Resultados

Imagen 8. Original ***Imagen 25. Con filtro Binomial***

Se puede concluir que el efecto de filtro binomial se desarrolló efectivamente ya que el objetivo de esta es suavizar y eliminar el ruido, esto se evidencia al comparar la imagen resultante con la original, cabe destacar que al usar una matriz 7x7 el efecto de suavizado es evidente.

3.1.5 FILTRO GAUSSIANO (Especificando la desviación estándar)

Este filtro es parecido al filtro de la media ya que disminuye la nitidez y aumenta la borrosidad, aunque pierde un poco los detalles. Este suavizado es un poco más uniforme.

Para aplicar el filtro se utiliza la siguiente ecuación

$$g(x,y) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Imagen 26. Ecuación del filtro de gauss.

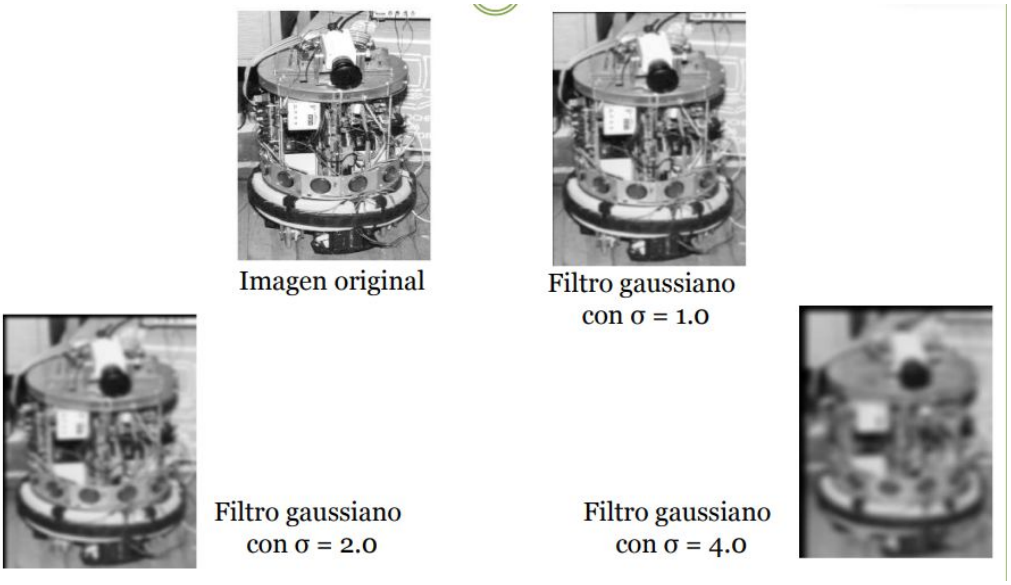


Imagen 27. Filtro de gauss en una fotografía.

Diseño lógico del algoritmo:

Se llama la función EscalGrices() para volver la imagen gris
Se declara una matriz para recoger los pixeles en gris

```
tol=5;
//la tolerancia implica la cantidad de vecinos cercanos que tiene el pixel escogido y se pone como tamño en la matriz
MatrizComparar = new int[(tol*2)+1][(tol*2)+1];
cont =0;
contM=0, contN=0;

hacer i=0 hasta ancho
   hacer i=0 hasta alto
      hacer m=i-tol hasta m=i+tol
         hacer n=j-tol hasta n+tol
            MatrizComparar[contM][contN] = valorGrisOriginal[m][n];
            cont++;
            contM++;
         }
         contM=0;
         contN++;
      }
```

```
        //se llama a una funcion que en este caso se llama calcularGrisGaus y se le envian tres cosas la matriz que rodea al pixel a analizar, el pixel al cual le vamos a cambiar el nivel de gris, y la tolerancia que es el tamaño de la matriz
                valorGrisEditado[i][j]=calcularGrisGaus(MatrizComparar,valorGrisOriginal[i][j],tol);

        //lo que hace esta función es primero sacar la matriz con valores de gauss según el tamaño de tolerancia que se le envíe, luego este lo opera en el píxel original que le queremos cambiar el nivel de gris y luego de esto sale una matriz de tamaño tolerancia que se opera fácilmente sobre la matriz original que se le envía a la función.
                    cont=0
                    contN=0
                    contM=0
                }
            }

            Se llama a la función CrearImagenGris(valorGrisEditado)
```

Resultados

Imagen 8. Original *Imagen 28. Con filtro gaussiano*

Al observar la imagen original con la imagen usando filtro gaussiano, se puede ver que es un filtro de suavizado, similar al filtro de pasa bajos, sin embargo, lo que la caracteriza es que esta causa una disminución de la nitidez donde hay una pérdida de detalles pero es más uniforme.

3.1.6 FILTRO DIGITADO A MANO (con valores en el dominio de los reales tamaño mínimo 3x3)

Diseño lógico del algoritmo:

```
se llama la función EscalGrices() para volver la imagen gris
se declara una matriz para recoger los pixeles en gris

valorGrisEditado = new int[ancho][alto];
valorPromedio=0f
tol=1;
cont =0;
contM=0,contN=0

matrizPersonal =new int[3][3]
matrizPersonal = la matriz que se quiere aplicar personal
hacer i=0 hasta ancho
    hacer j=0 hasta alto
        hacer m=i-tol hasta i+tol
            hacer n=j-tol hasta n+tol
                valorPromedio =
(matrizPersonal[contM][contN]*valorGrisOriginal[m][n])+valorPromedio;
                cont++;
                contM++;
            }
            contM=0;
```

```
            contN++;
        }
        valorPromedio=valorPromedio/Divisor;
        valorGrisEditado[i][j]=pasar a entero(valPromedio);

        si valorGrisEditado[i][j] menor que 0
            valorGrisEditado[i][j]=0
        }si valorGrisEditado[i][j] mayor que 255
            valorGrisEditado[i][j]=255;
        }

        valPromedio=0f;
        cont=0;
        contN=0;
        contM=0;
    }
}
```

Se llama a la función CrearImagenGris(valorGrisEditado).

Resultados

Matriz 5. Matriz del filtro personalizada. Matriz 3x3

Imagen 8.** Original* ***Imagen 29. Con filtro personalizada

Para el filtro personalizado, se usó una matriz cualquiera (Matriz 5) y da como resultado una imagen que pierde un poco la nitidez, pero sin perder totalmente los detalles por lo que no es un filtro de suavizado tan potente como los mencionados anteriormente.

3.2 FILTROS NO LINEALES

Son filtros no lineales ya que la manera de ordenar los valores es con la vecindad de cada píxel y así poder adquirir un valor.

3.2.1 FILTRO DEL MÍNIMO:

Este filtro sirve para eliminar el ruido tipo sal, es decir los pixeles blancos. Lo que sucede es que solo funciona para eliminar este ruido, ya que selecciona el menor valor de intensidad. Al aplicar este filtro la imagen se tiende a oscurecer.

Diseño lógico del algoritmo:

Se llama la función EscalGrices() para volver la imagen gris
Se declara una matriz para recoger los pixeles en gris

```
valorGrisEditado = new int[ancho][alto];
tol=1
cont =0
```

```
Ordenar=new int[9]
 hacer i=0 hasta ancho
   hacer j=0 hasta alto
      hacer m=i-tol hasta i+tol
         hacer n=j-tol hasta n+tol
            Ordenar[cont]=ValorGrisOriginal[m][n]
            cont++
         }
      }
      Se llama a la función ordenarMM() se le pasa como parámetros el arreglo a ordenar y una
opción, si       es 1 devuelve el maximo, si es 2  minimo y si es 3 la mediana.
      grises2[i][j]=OrdenarMM(Ordenar,2);
      cont=0;
   }
}

Se llama a la función CrearImagenGris(valorGrisEditado).
```

Resultado

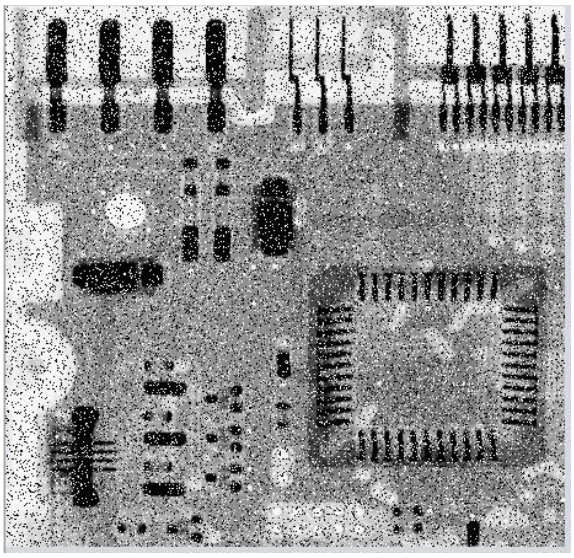


Imagen 30. Original

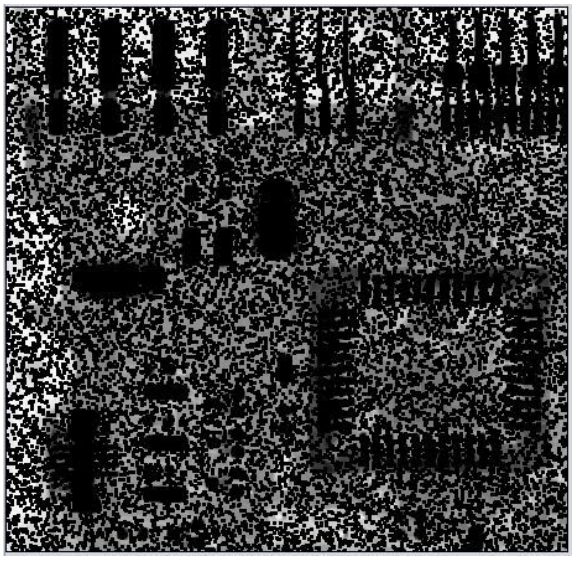


Imagen 31. Con filtro Mínimo

Como se podrá evidenciar, la imagen original se ve clara y al usar el filtro del mínimo causa un oscurecimiento de la imagen, eliminando los pixeles blancos (ruido tipo sal).

3.2.2 FILTRO DEL MÁXIMO

Este filtro sirve para eliminar el ruido tipo pimienta, es decir los pixeles negros. Es decir selecciona el mayor valor de intensidad de valores de nivel de gris. Al aplicar este filtro la imagen se tiende a aclarar.

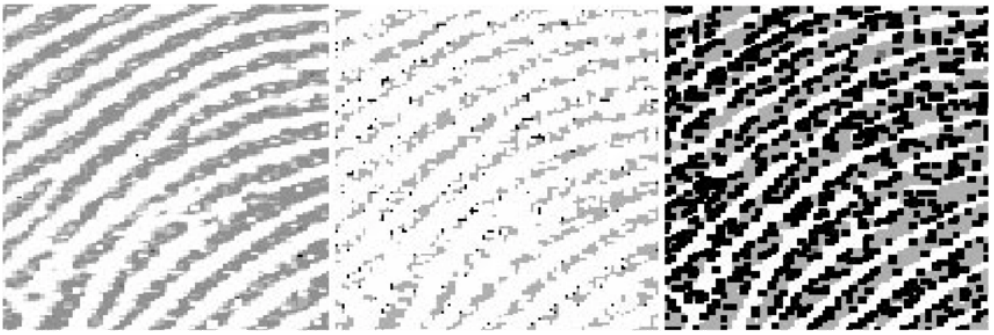


Imagen 31. Filtro de máximo y mínimo de una huella dactilar.

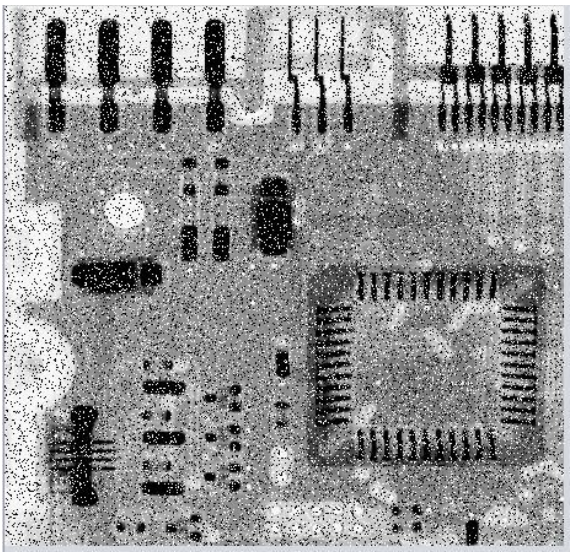
Diseño lógico del algoritmo:

```
Se llama la función EscalGrices() para volver la imagen gris
Se declara una matriz para recoger los pixeles en gris

valorGrisEditado = new int[ancho][alto];
tol=1
cont =0
Ordenar=new int[9]
 hacer i=0 hasta ancho
   hacer j=0 hasta alto
      hacer m=i-tol hasta i+tol
         hacer n=j-tol hasta n+tol
            Ordenar[cont]=grises[m][n]
            cont++
         }
      }
      Se llama a la función ordenarMM() se le pasa como parámetros el arreglo a ordenar y
una opción, si       es 1 devuelve el maximo, si es 2  minimo y si es 3 la mediana.
      grises2[i][j]=OrdenarMM(Ordenar,1);
      cont=0;
   }
}

Se llama a la función CrearImagenGris(valorGrisEditado).
```

Resultados



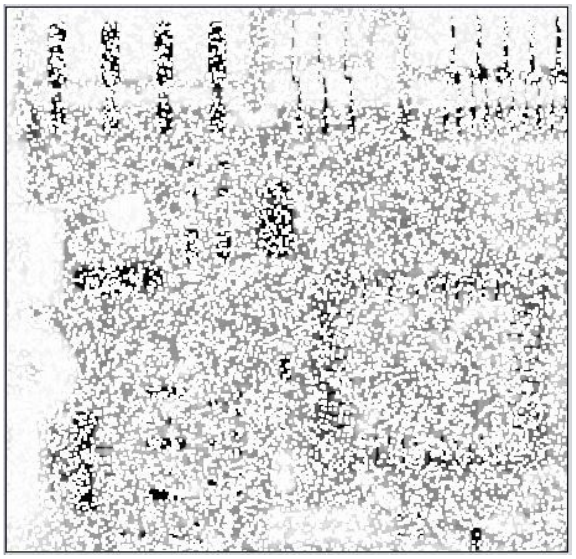


Imagen 30. Original *Imagen 32. Con filtro de Máximos*

Al pasar la imagen original por un filtro de máximos causa un aclaramiento de la imagen, por lo cual se puede concluir que efectivamente sí se eliminaron los pixeles negros (ruido pimienta) de esta.

3.2.4 FILTRO DE LA MEDIANA:

Este filtro lo que hace es seleccionar el valor intermedio de la posición intermedia. Esta operación la hace después de ordenar los valores de menor a mayor de la vecindad de cada pixel. Lo que hace este filtro es atenuar el ruido de sal y pimienta. Algunas veces los inconvenientes que genera son el redondeo de las esquinas de los objetos.

Imagen 33. Filtro de la mediana en una fotografía.

Diseño lógico del algoritmo:

```
se llama la función EscalGrices() para volver la imagen gris
se declara una matriz para recoger los pixeles en gris

valorGrisEditado = new int[ancho][alto];
tol=1
cont =0
Ordenar=new int[9]
 hacer i=0 hasta ancho
   hacer j=0 hasta alto
      hacer m=i-tol hasta i+tol
         hacer n=j-tol hasta n+tol
            Ordenar[cont]=grises[m][n]
            cont++
         }
      }
      Se llama a la función ordenarMM() se le pasa como parámetros el arreglo a ordenar y una
opción, si       es 1 devuelve el maximo, si es 2  minimo y si es 3 la mediana.
      grises2[i][j]=OrdenarMM(Ordenar,3);
      cont=0;
   }
}

Se llama a la función CrearImagenGris(valorGrisEditado).
```

Resultados

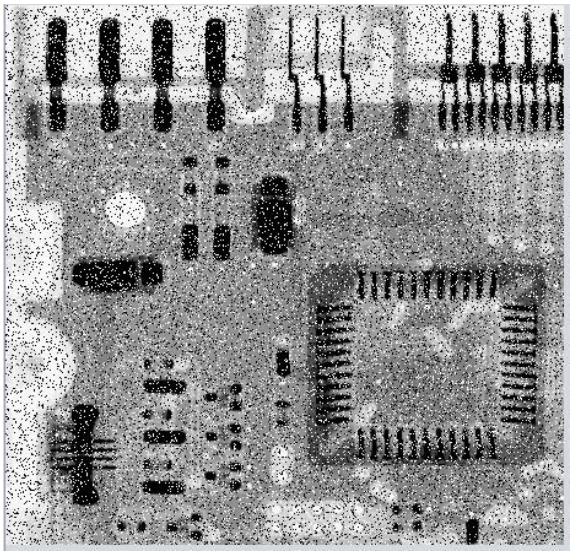


Imagen 30. Original

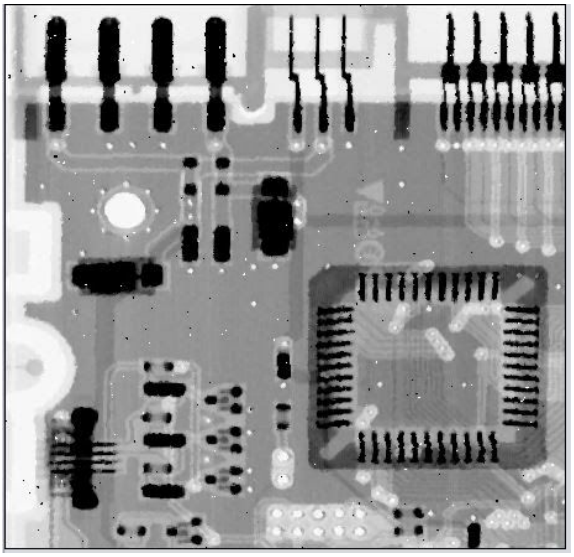


Imagen 34. Con el filtro de la mediana

Se puede probar una diferencia entre la imagen original y la imagen que pasapor el filtro de la mediana, con esta imagen se puede ver una preservación de los bordes y elimina lo que se denomina el ruido impulsional (ruido sal y pimienta).

3.2.3 FILTRO DE LA MODA

Este filtro se substituye el valor de cada píxel por el valor de la moda de la vecindad. La moda se obtiene del valor más frecuente dentro de la vecindad de pixeles, si hay dos modas se promedian para obtener el valor del pixel. El resultado es muy parecido al de la mediana pero el resultado es mucho mejor ya que es más sutil el cambio y no se pierde tanta información, los bordes de los objetos se mantiene.

Diseño lógico del algoritmo:

```
Se llama la función EscalGrices() para volver la imagen gris
Se declara una matriz para recoger los pixeles en gris

valorGrisEditado = new int[ancho][alto];
MatrizComparar = new int[3][3];

int tol=1;
int cont =0;
int contM=0,contN=0;

 hacer i=0 hasta ancho
   hacer j=0 hasta alto
      hacer m=i-tol hasta i+tol
         hacer n=j-tol hasta n+tol
            MatrizComparar[contM][contN] = valorGrisOriginal[m][n]
            cont++;
            contM++
         }
         contM=0
         contN++
      }
// se llama a la función ObtenerModa() la cual recibe como parámetro una matriz 3x3 y retorna un
valor entero con la moda de la matriz que se envió
         valorGrisEditado[i][j]=ObtenerModa(MatrizComparar);
         cont=0;
         contN=0;
         contM=0;
   }
}
Se llama a la función CrearImagenGris(valorGrisEditado).
```

Resultados

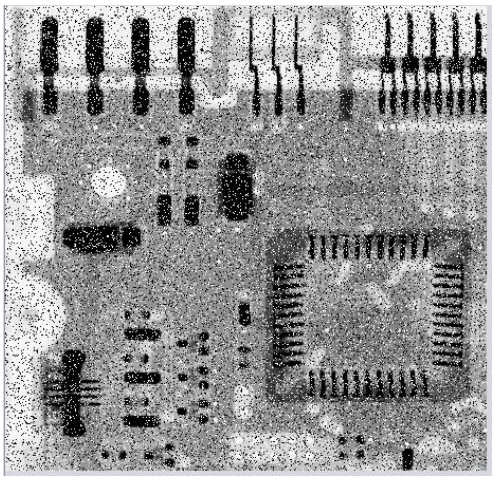


Imagen 30. Original

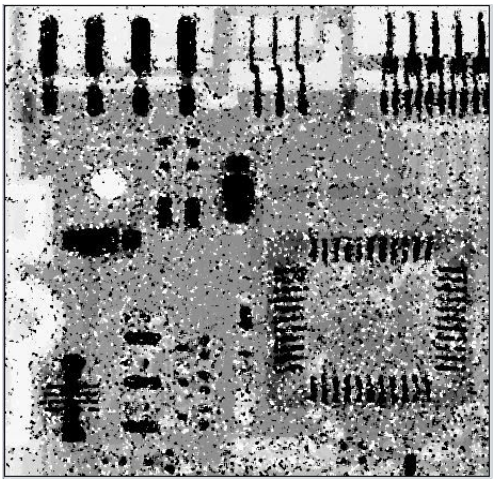


Imagen 35. con el filtro de la Moda

Para el resultado del presente filtro, se puede ver que la imagen no es tan brusca en la que conserva muchos de sus detalles, y a su vez conserva los bordes.

4. BIBLIOGRAFÍA

1. Unovi. Filtrado Espacial. *http://www6.uniovi.es/vision/intro/node41.html*. Accessed *April 27, 2020.*
2. *González, R.C., Wintz, P. (1996), Procesamiento digital de imágenes. AddisonWesley, Tema 3,4, pág 89-269.*
3. R. González and R. E. Woods, *Digital Image Processing, 3rd ed. Upper Saddle River, New Jersey.: Prentice-Hall Inc.,* 2008. [Online]. http://www.imageprocessingplace.com/. Accessed April 27, 2020.
4. Madrigal, Carlos & Madrigal, Jaime & Arbeláez, Juan & Fernández,*" David. Diseño de un sistema biométrico de identificación usando sensores capacitivos para huellas dactilares. Revista de la Facultad de Ingeniería";* 2007.https://www.researchgate.net/publication/262935804/figure/fig7/AS:489883019026440@1493808621931/Figura-7-Filtro-de-maximo-y-minimo.png. Accessed April 27, 2020.
5. Pablo Turmero. "Filtrado espacial de imágenes";2020. *https://www.monografias.com/trabajos108/filtrado-espacial-imagenes/filtrado-espacial-imagenes.shtml.* Accessed April 27, 2020.
6. Image Processing Place. *http://www.imageprocessingplace.com/.* 2020. Accessed April 27, 2020.

7. Image Proccessing. https://www.imageeprocessing.com/2013/04/min-filter-matlab-code.html. Accessed April 27, 2020.
8. Rafael C. González and Richard E. Woods. Digital Image Processing 4th Edition. (2008-2018). Pearson. Accessed April 30, 2020.