Universidad de San Carlos de Guatemala
Facultad de Ingeniería
Escuela de Ingeniería Mecánica Eléctrica

RESTAURACIÓN DE IMÁGENES DISTORSIONADAS MEDIANTE TÉCNICAS DE PROCESAMIENTO DIGITAL Y COMPARACIÓN ENTRE DOS MÉTODOS DE RESTAURACIÓN

Jorge Alejandro Vettorazzi González
Asesorado por el Ing. Enrique Edmundo Ruíz Carballo

Guatemala, abril de 2007

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

RESTAURACIÓN DE IMÁGENES DISTORSIONADAS MEDIANTE TÉCNICAS DE PROCESAMIENTO DIGITAL Y COMPARACIÓN ENTRE DOS MÉTODOS DE RESTAURACIÓN

TRABAJO DE GRADUACIÓN

PRESENTADO A LA JUNTA DIRECTIVA DE LA
FACULTAD DE INGENIERÍA
POR:

JORGE ALEJANDRO VETTORAZZI GONZÁLEZ
ASESORADO POR EL ING. ENRIQUE EDMUNDO RUÍZ CARBALLO

AL CONFERÍRSELE EL TÍTULO DE
INGENIERO ELECTRÓNICO

GUATEMALA, ABRIL DE 2007

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA

FACULTAD DE INGENIERÍA

NÓMINA DE JUNTA DIRECTIVA

DECANO	Ing. Murphy Olympo Paiz Recinos
VOCAL I	Ing. Glenda Patricia García Soria
VOCAL II	Ing. Alba Maritza Guerrero de López
VOCAL III	Ing. Miguel Ángel Dávila Calderón
VOCAL IV	Br. Kenneth Issur Estrada Ruiz
VOCAL V	Br. Elisa Yazminda Vides Leiva
SECRETARIA	Inga. Marcia Ivonne Véliz Vargas

TRIBUNAL QUE PRACTICÓ EL EXAMEN GENERAL PRIVADO

DECANO	Ing. Sydney Alexander Samuels Milson
EXAMINADOR	Ing. Juan Carlos Córdova Zeceña
EXAMINADOR	Ing. Erwin Efraín Segura Castellanos
EXAMINADOR	Ing. Fernando Manuel Barrera Pérez
SECRETARIO	Ing. Pedro Antonio Aguilar Polanco

HONORABLE TRIBUNAL EXAMINADOR

Cumpliendo con los preceptos que establece la ley de la Universidad de San Carlos de Guatemala, presento a su consideración mi trabajo de graduación:

RESTAURACIÓN DE IMÁGENES DISTORSIONADAS MEDIANTE TÉCNICAS DE PROCESAMIENTO DIGITAL Y COMPARACIÓN ENTRE DOS MÉTODOS DE RESTAURACIÓN,

tema que me fuera asignado por la Dirección de la Escuela de Ingeniería Mecánica Eléctrica, el 19 de noviembre de 2004.

Alejandro Vettorazzi

Jorge Alejandro Vettorazzi González

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

Guatemala, 20 marzo de 2007

Ingeniero
Julio Cesar Solares Peñate
Coordinador Área de Electrónica
Escuela de Ingeniería Mecánica Eléctrica

Estimado Ingeniero:

Por este medio le informo que he revisado el trabajo de graduación titulado: **Restauración de Imágenes distorsionadas mediante técnicas de procesamiento digital y comparación entre dos métodos de restauración,** elaborado por el estudiante Jorge Alejandro Vettorazzi González, Carné 1999 11000.

El mencionado trabajo llena los requisitos para dar mi aprobación, e indicar que el autor y mi persona somos responsables por el contenido y conclusiones de la misma.

Atentamente

Ing. Enrique Edmundo Ruiz Carballo
ASESOR

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

Guatemala, 27 de marzo 2007.

Señor Director
Ing. Mario Renato Escobedo Martìnez
Escuela de Ingenierìa Mecànica Elèctrica
Facultad de Ingenierìa, USAC.

Señor Director:

Me permito dar aprobación al trabajo de Graduación titulado: **Restauración de Imàgenes distorsionadas mediante tècnicas de procesamiento digital y comparación entre dos mètodos de restauración, des**arrollado por el estudiante; Jorge Alejandro Vettorazzi Gonzàlez, por considerar que cumple con los requisitos establecidos para tal fin.

Sin otro particular, aprovecho la oportunidad para saludarle.

Atentamente,

ID Y ENSEÑAD A TODOS

Ing. Julio Cesar Solares Peñate
Coordinador Area de Electrònica

JCSP/sro

UNIVERSIDAD DE SAN CARLOS
DE GUATEMALA

FACULTAD DE INGENIERIA

El Director de la Escuela de Ingenierìa Mecànica Elèctrica, después de conocer el dictamen del Asesor, con el Visto Bueno del Coordinador de Area, al trabajo de Graduación del estudiante; Jorge Alejandro Vettorazzi Gonzàlez titulado: **Restauración de Imàgenes distorsionadas mediante tècnicas de procesamiento digital y comparación entre dos mètodos de restauración,** procede a la autorización del mismo.

Ing. Mario Renato Escobedo Martìnez
DIRECTOR

UNIVERSIDAD DE SAN CARLOS
FACULTAD DE INGENIERIA
DIRECCION ESCUELA DE INGENIERIA
MECANICA ELECTRICA
DE GUATEMALA

GUATEMALA, 10 DE ABRIL 2,007.

Universidad de San Carlos
de Guatemala

Facultad de Ingeniería
Decanato

Ref. DTG. 111.2007

El Decano de la Facultad de Ingeniería de la Universidad de San Carlos de Guatemala, luego de conocer la aprobación por parte del Director de la Escuela de Ingeniería Mecánica Eléctrica, al trabajo de graduación titulado: **RESTAURACIÓN DE IMÁGENES DISTORSIONADAS MEDIANTE TÉCNICAS DE PROCESAMIENTO DIGITAL Y COMPARACIÓN ENTRE DOS MÉTODOS DE RESTAURACIÓN,** presentado por el estudiante universitario **Jorge Alejandro Vettorazzi González,** procede a la autorización para la impresión del mismo.

IMPRÍMASE.

Ing. Murphy Olympo Paiz Recinos
DECANO

Guatemala, abril de 2007

UNIVERSIDAD DE SAN CARLOS DE GUATEMALA
DECANO
FACULTAD DE INGENIERIA

/gdech

AGRADECIMIENTOS A:

Mis Padres	Por haberme enseñado los caminos por los cuales he de caminar durante mi vida, porque con su amor no tuve falta de nada.
Mi Asesor	Ing. Enrique Ruiz Carballo, gracias por todo su apoyo y ayuda brindada para la realización del presente trabajo de graduación.
Departamento de Matemática	Por la oportunidad que se me brindo para laborar en ese Departamento durante 2 años y medio; en especial al Ing. Arturo Samayoa quien fue una guía durante mi estancia laboral. Así también agradezco a las siguientes personas: Ing. Rodolfo Samayoa, Ing. Vera Marroquín, Lic. Ángel Arévalo, Ing. Luís Córdova, Lic. Enrique Pazos y al Ing. Alberto Boy por haberme brindado su amista.
Mis Amigos	Maria José Herrera, Efraín Paiz, Marco García y Ángel López, Mesala Palencia, Pedro Obregón, Steeve Toledo, Pamela Vega, Marco Tulio García, Fernando Mazariegos, Mario Silvestre, Sergio Wolford, Sergio Carcamo, José Córdova, Edmar, Miguel, Carlos Bolaños, José Asencio, Carlos Melgar, Jorge Ordeñoez, Otto Paau, Walter Reyes, Pedro Morales, Juan Manuel Flores, Cesar

Montejo, Ingrid Batres, Juan Carlos Brolo, Moisés Ordóñez, Mario Mérida, Juan Fernando López, Mynor Abimael y Andrea; quienes a lo largo del tiempo se han convertido en algo más que amigos, en hermanos

DEDICATORIA

A:

Dios

Por sus grandes bendiciones en mi vida y por permitirme comenzar una nueva etapa.

Mis Padres

Jorge Mario Vettorazzi Gándara y Sonia Maria de Los Ángeles de Vettorazzi por haberme brindado todo el apoyo necesario y su amor incondicional en todo momento.

Mis Hermanos

Verónica, Claudia y Mario.

ÍNDICE GENERAL

ÍNDICE DE ILUSTRACIONES **V**
LISTA DE SÍMBOLOS **VII**
GLOSARIO **IX**
RESUMEN **XI**
OBJETIVOS **XIII**
INTRODUCCIÓN **XV**

1. INTRODUCCIÓN AL PROCESAMIENTO DE IMÁGENES DIGITALES **1**
1.1 Imágenes Digitales 1
1.1.1 Definición de Imagen e Imágenes Digitales 1
1.1.2 Conceptos Básicos 2
1.1.3 Representación de Imágenes Digitales 4
1.2 Análisis Matemático en el Procesamiento de Imágenes 7
1.2.1 Procesamiento Lineal de Imágenes 7
1.2.1.1 Sistemas Lineales 7
1.2.1.2 Operadores Lineales 9
1.2.2 Convolución 9
1.2.3 La Transformada de *Fourier* y el Dominio de la Frecuencia 10
1.2.3.1 La Transformada de *Fourier* 12
1.2.3.2 La Transformada Discreta de *Fourier* 13
1.2.3.3 La Transformada Rápida de *Fourier* 14

2. DISTORSIÓN EN IMÁGENES DIGITALES **17**
2.1 Modelo para el proceso de distorsión en una imagen 18
2.2 Ruido en Imágenes y su modelación 19

2.2.1 Ruido Gaussiano 20
2.2.2 Ruido Gamma 21
2.2.3 Ruido Exponencial 22
2.2.4 Ruido Uniforme 23
2.2.5 Ruido de Impulso 25
2.2.6 Ruido Periódico 26
2.3 La Función de Degradación 27
2.3.1 Estimación de la Función de Degradación 28
2.3.1.1 Estimación por Experimentación 29
2.3.1.2 Estimación Mediante Modelación Matemática 30

3. RESTAURACIÓN DE IMÁGENES, EL MÉTODO DE FILTRADO INVERSO Y EL METODO DE FILTRADO DE *WIENER* 31
3.1 Modelo de Degradación/Restauración de una Imagen 32
3.2 El Método de Filtrado Inverso 34
3.2.1 Filtro Pseudo Inverso 37
3.3 Método de Filtrado de *Wiener* 39
3.3.1 Error Cuadrático Medio 39
3.3.2 Ecuación del Filtro de *Wiener* 41

4. DESARROLLO DE ALGORITMOS PARA LA APLICACIÓN DEL FILTRADO INVERSO Y FILTRADO DE *WIENER* 47
4.1 Simulación de la Función de Degradación y Ruido Aditivo en *Matlab* 49
4.2 Diseño de un Algoritmo de Programación para el Filtrado Inverso 51
4.2.1 Diagrama de Bloques 51
4.2.2 Captura de Imagen 52

4.2.2.1 Captura y reconocimiento de un archivo de imagen 53
4.2.2.2 Captura y reconocimiento de una imagen a partir de un archivo de video 57
4.2.2.3 Conversión a Escala de grises y lectura de la imagen 58
4.2.3 Análisis en Frecuencia de los elementos de la imagen 58
4.2.4 Análisis de la Función de Degradación 60
4.2.5 Etapa de Filtrado de la Imagen Degradada 61
4.3 Pruebas, Método del Filtro Inverso y Resultados 62
4.4 Código Fuente de la GUI para la aplicación del Filtro Inverso 68
4.5 Diseño de un Algoritmo de Programación para el Filtrado de *Wiener* 75
4.5.1 Diagrama de Bloques 75
4.5.2 Análisis de la Función de Degradación y Ruido en la Imagen 76
4.5.3 Cálculo de la Relación Señal a Ruido en la Imagen 77
4.5.4 Etapa de Filtrado y Cálculo del Error Cuadrático Medio 77
4.6 Pruebas, Método del Filtro de *Wiener* y Resultados 78
4.6.1 Imágenes Distorsionadas con Ruido 82
4.7 Código Fuente de la GUI para la aplicación del Filtrado de *Wiener* 85

CONCLUSIONES **91**
RECOMENDACIONES **95**
BIBLIOGRAFÍA **97**

ÍNDICE DE ILUSTRACIONES

FIGURAS

1.	Digitalización de una imagen	3
2.	Representación de una imagen digital	4
3.	Homogeneidad en un sistema lineal	8
4.	Transformada de *Fourier* y Transformada Inversa de *Fourier*	11
5.	Modelo para la degradación de una imagen	18
6.	Distribución Gaussiana	21
7.	Distribución Gamma	22
8.	Distribución Exponencial	23
9.	Densidad Ruido Uniforme	24
10.	Efectos del Ruido en las imágenes digitales	26
11.	Efecto borroso en una imagen	27
12.	Respuesta impulso en una imagen	30
13.	Modelo de Degradación/Restauración de una imagen	33
14.	Imagen restaurada luego de la etapa de filtrado	36
15.	Efectos de la distorsión y ruido en imágenes digitales	50
16.	Diagrama de bloques Filtrado Inverso	51
17.	Diagrama de bloques etapa de captura de la imagen	52
18.	Arreglo matricial para la representación de una imagen	56
19.	Imagen y su espectro calculado en *Matlab*	59
20.	Interfaz GUI para la aplicación del Filtro Inverso	63
21.	Degradación de una imagen con desplazamiento de *pixels*	64
22.	Imagen restaurada luego de la etapa de filtrado	65
23.	Degradación de una imagen con desplazamiento de *pixels*	65

24. Imagen restaurada 66
25. Degradación de una imagen con desplazamiento de *pixels* 66
26. Imagen restaurada 67
27. Diagrama de bloques Filtrado de *Wiener* 75
28. Interfaz GUI para la aplicación del Filtro de *Wiener* 79
29. Degradación de imagen con desplazamiento de *pixels* 80
30. Imagen restaurada luego de la etapa de filtrado 81
31. Degradación de imagen con desplazamiento de *pixels* y ruido 83
32. Imagen Restaurada luego de la primera iteración 83
33. Imagen Restaurada luego de segunda iteración 84

TABLA

I. Formatos de Imágenes compatibles en *Matlab* 55

LISTA DE SÍMBOLOS

k	Número de bits que corresponden a la escala de grises en una imagen
μ	Media de la función de densidad de probabilidad
σ	Desviación estándar de la función de densidad de probabilidad
f(x,y)	Representación de la imagen original
g(x,y)	Representación de la imagen degradada
h(x,y)	Función de Degradación
n(x,y)	Ruido aditivo
f´(x,y)	Estimado de la imagen original
$\mathbf{e^2}$	Error Cuadrático Medio
w(x,y)	Filtro de *Wiener*

GLOSARIO

PIXEL Elemento más pequeño que conforma una imagen digital.

PDS PDS o Procesamiento Digital de Señales, se refiere al estudio de las señales en su representación digital, y de los métodos de procesamiento utilizados para su análisis.

MUESTREO Proceso que toma muestras periódicas de la amplitud de una determinada señal continua

CUANTIZACIÓN Proceso a través del cual se le asigna un valor discreto a la muestra de una señal continua

MATLAB *MatrixLaboratory.* Lenguaje de alto nivel para la computación técnica, de la empresa *Mathworks Inc.*

RESUMEN

El presente trabajo estudia la restauración de Imágenes digitales mediante la aplicación del método del Filtrado Inverso y el Filtrado de *Wiener*, utilizando un algoritmo de programación que permitirá remover los efectos que degradan una determinada imagen, utilizando un modelo de Degradación/Restauración previamente establecido, considerando que los efectos que causan la degradación se conocen con cierto grado de exactitud.

En el primer capítulo se presenta una introducción sobre los conceptos y la forma en la cual se representan las imágenes digitales; se detallan las herramientas matemáticas para el análisis de las imágenes digitales, tales como la Transformada de *Fourier* y la Convolución. El capítulo dos se concentra en los fenómenos que causan la degradación de una imagen digital; se presenta el modelo de degradación y el concepto de Función de Degradación y ruido en imágenes. En el tercer capítulo se describe el modelo de Degradación/Restauración y se explican los métodos de restauración basados en el Filtro Inverso y el Filtro de *Wiener*. Por último, el capítulo cuatro muestra los modelos de restauración por medio de diagramas de bloques y su respectivo algoritmo de programación.

OBJETIVOS

- **GENERAL**

Diseñar un *software* de aplicación para cada uno de los métodos de filtrado propuestos, que permita realizar el análisis de una imagen degradada, con el fin de restaurar la información contenida en ésta.

- **ESPECÍFICOS**

1. Definir el Procesamiento Digital de Imágenes y los procesos matemáticos ligados a éste.

2. Describir el proceso de distorsión en imágenes digitales y sus causas.

3. Describir las principales características y comportamiento del Método de Filtrado Inverso y Pseudoinverso.

4. Describir las principales características y comportamiento del Método de Filtrado de *Wiener*.

INTRODUCCIÓN

Debido al creciente avance en técnicas de procesamiento digital, ha sido posible presenciar cómo su aplicación permite analizar y resolver diversidad de problemas, en campos tales como el tratamiento de señales de audio, diseño de filtros digitales o aplicaciones de procesamiento linear de imágenes (esto sólo por mencionar algunos casos). El tema del presente trabajo de graduación involucra el último de estos ejemplos.

Como se expondrá en los próximos capítulos, el propósito de la restauración de imágenes es el de compensar o deshacer los defectos que degradan una determinada imagen, mediante un previo conocimiento del fenómeno que causa la distorsión; para esto se han desarrollado diversidad de métodos a lo largo de los años, que permiten a través del procesamiento digital, analizar el tipo de distorsión que presenta la imagen y aplicar una solución adecuada para recuperar la información contenida en la imagen.

Para propósitos del presente trabajo, se estudiarán dos de los métodos más utilizados para la restauración de imágenes como son el Filtrado Inverso y el Filtrado de *Wiener*, se estudiarán sus principales características y se propondrá un algoritmo de programación para la aplicación de cada uno de los métodos, de manera que sea posible realizar así una comparativa entre los métodos de restauración estudiados.

1. INTRODUCCIÓN AL PROCESAMIENTO DE IMÁGENES DIGITALES

1.1 Imágenes Digitales

Con el fin de establecer ciertos conceptos relacionados con la restauración de imágenes digitales, es conveniente exponer los principios y conceptos sobre los cuales se basa el procesamiento de imágenes, entre estos, la representación matemática que nos permite trabajar con imágenes en dos dimensiones.

1.1.1 Definición de Imagen e Imágenes Digitales

Una imagen se define como una descripción de cómo varia un parámetro sobre una superficie determinada, por ejemplo, una imagen visual resulta de las variaciones de la luz sobre un plano de dos dimensiones. Sin embargo la luz no es el único parámetro utilizado en el estudio de imágenes, por ejemplo, una imagen puede formarse considerando la temperatura en un circuito integrado, la velocidad de la sangre en una arteria o el movimiento de la tierra durante un terremoto, etc. Estas imágenes generalmente se convierten en imágenes de luz de manera que puedan ser evaluadas por el ojo humano.

Una imagen puede representarse matemáticamente como una función bidimensional, f(x,y), donde x ý y son la representación de "coordenadas espaciales", y la amplitud de f en cualquier par de coordenadas (x, y) es llamada "intensidad" o "escala de grises" de la imagen en ese punto. El termino "escala de grises" es utilizado de una manera continua para referirse a la intensidad de las imágenes monocromáticas.

Se le llama "Imagen Digital" a la imagen cuando los valores de amplitud de x, y ý f son cantidades discretas. Esto es posible mediante un proceso de "muestreo" y "cuantización" en la imagen, a este proceso se le conoce también con el nombre de "digitalización de la imagen".

1.1.2 Conceptos básicos para la representación de Imágenes Digitales

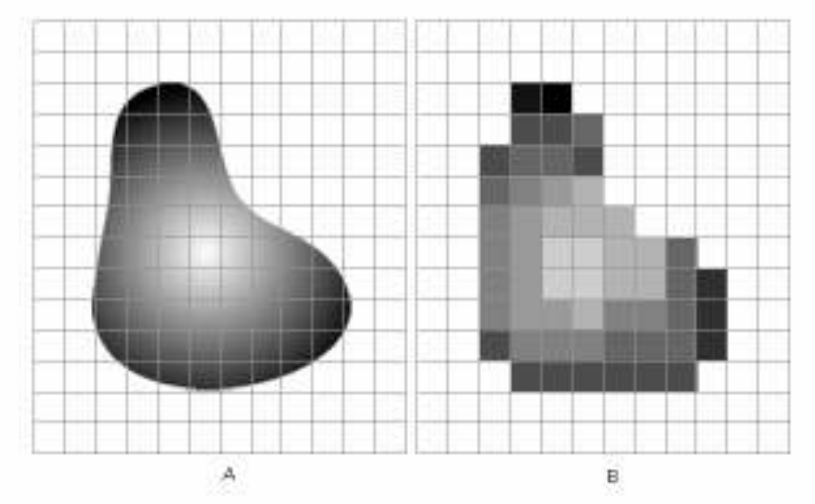
El objetivo principal cuando adquirimos cualquier tipo de imagen, es poder representarla de manera que seamos capaces de procesar el tipo de información que la imagen nos proporciona. Cuando este proceso termina seremos capaces de comprender la información contenida en esta.

Cada elemento en la imagen tiene un valor en particular, cuando la imagen es adquirida, este número esta relacionado con la temperatura de un circuito integrado o a la velocidad de la sangre en una arteria por hacer mención de los ejemplos antes citados. A manera de mostrar esta imagen de una manera "visual", el valor asignado a cada elemento es convertido a una "escala de grises", en donde el menor valor es representativo del negro y el valor mas alto representa el color blanco, los valores intermedios son distintos niveles de grises.

Para crear una imagen digital es necesario convertir la información continua obtenida por medio de sensores a una forma digital; somos capaces de hacerlo por medio del muestreo y cuantización. Por ejemplo si tenemos una imagen continua f(x,y), para digitalizarla necesitamos muestrear la función f(x,y) en sus coordenadas como también en su amplitud. La digitalización de las coordenadas de la imagen es llamada "muestreo" y la digitalización de la amplitud de la imagen es llamada "cuantización". Estos términos no nos son desconocidos ya que son herramienta matemática esencial del Procesamiento Digital de Señales.

Con el fin de exponer más claramente los conceptos de muestreo y cuantización en una imagen bidimensional, podemos ejemplificarlos por medio del la digitalización de una imagen continúa como la que se muestra en la figura 1:

Figura 1. **Digitalización de una imagen a través de muestreo y cuantización**



Fuente: R. Gonzalez, "Digital Image Processing" , Pearson Education 2002

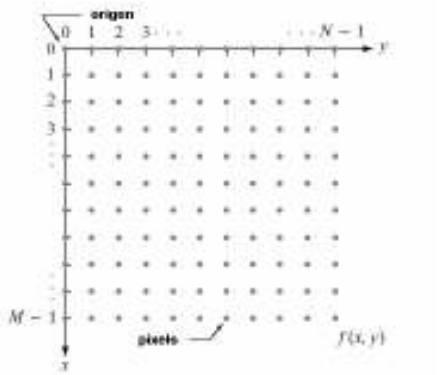
En la figura 1 A, se muestra una imagen continua f(x,y) proyectada por un sistema de sensores, y la figura 1 B, muestra la misma imagen luego de ser muestreada y cuantizada. Claramente podemos observar que la calidad de la imagen se determina por el número de muestras y niveles de escala de grises que se utilicen para muestrearla y cuantizarla.

1.1.3 Representación de Imágenes Digitales

El resultado del muestreo y cuantización es una matriz de M filas y N columnas, y como en la teoría de matrices se le llama una matriz de M X N, los valores de las coordinadas (x,y) de la imagen son convertidos en valores discretos y por claridad y conveniencia se utilizan números enteros para representar estos valores.

La convención utilizada para representar imágenes digitales se muestra en la figura 2. Los valores de las coordinadas en el origen (x, y) = (0, 0) y representan la posición de los *pixels* a lo largo de la matriz de la imagen.

Figura 2. **Representación de una Imagen digital en un plano**



Fuente: R. González, “Digital Image Processing” , Pearson Education 2002.

La notación utilizada para la representación de imágenes digitales, nos permite escribir la matriz M X N como se muestra a continuación:

$$f(x,y) = \begin{bmatrix} f(0.0) & f(0,1) & ..... & f(0,N-1) \\ f(1,0) & f(1,1) & ..... & f(1,N-1) \\ \downarrow & \downarrow & \downarrow & \downarrow \\ f(M-1,0) & f(M-1,1) & ..... & f(M-1,N-1) \end{bmatrix} \qquad (1.0)$$

El lado derecho de la ecuación es por definición una imagen digital. A la intersección de una fila con una columna recibe el nombre de “píxel” (derivado de la palabra en ingles “picture element” o “elemento de la imagen”).

El proceso de digitalización requiere la toma de decisiones acerca de los valores que tomaran M y N y también para el número de niveles en la escala de grises discretos que serán permitidos para cada píxel. No existen requerimientos específicos para los valores de M y N aparte de requerir que estos sean enteros positivos, sin embargo, debido al procesamiento de imágenes, memoria y consideraciones de hardware para muestreo, el numero de niveles en escala de grises es un entero definido por:

$$L = 2^k \qquad (1.1)$$

Donde L representa los niveles discretos de la escala de grises permitidos para cada píxel y k representa el numero de bits que corresponden a la escala de grises en la imagen, por ejemplo una imagen con 256 niveles en su escala de grises generalmente es conocida como una imagen de 8 bits.

El número de bits requerido para guardar una imagen digitalizada es igual a:

$$b = M \times N \times k \qquad (1.2)$$

Esta ultima ecuación nos proporciona una idea del tamaño de una imagen digital, por ejemplo si tenemos una imagen de 8 bits de tamaño 1024X1024 podemos ver que esta tendrá 8,388,608 bits, lo cual no es insignificante y debe tomarse en cuenta a la hora del procesamiento de la imagen.

Una típica imagen digital esta compuesta aproximadamente por 500 filas por 500 columnas. Esta es la calidad de imagen que se utiliza en televisión por ejemplo y en algunos programas de cómputo. Las imágenes con una menor resolución que la mencionada se considera de baja resolución, en algunos casos si la cantidad de píxeles es lo suficientemente baja las imágenes no se verán naturales y generalmente se pueden observar los píxeles individuales como elementos de la imagen. Las imágenes compuestas con mas de 1000 por 1000 píxeles con consideradas excepcionalmente buenas, esta es la calidad de las mejores aplicaciones de computo y televisión de alta definición. En el procesamiento de imágenes digitales es común utilizar 256 niveles de cuantización en la escala de grises de la imagen.

Existen muchas razones para esto una de ellas es que es conveniente para el manejo de datos, ya que esta es la forma en que una computadora guarda información.

Una segunda razón y una de las más importantes es que un tamaño de paso de 1/256 luminosidad es más pequeño de lo que el ojo humano puede percibir.

Una imagen presentada al ojo humano no presentara mejoras utilizando más de 256 niveles de cuantización, esto no significa que algunas imágenes necesiten ser procesadas con mas de 8 bits por píxel ya que la mayoría de imágenes que se procesan en la vida real son representativos de parámetros no visuales.

1.2 Análisis matemático en el Procesamiento de Imágenes

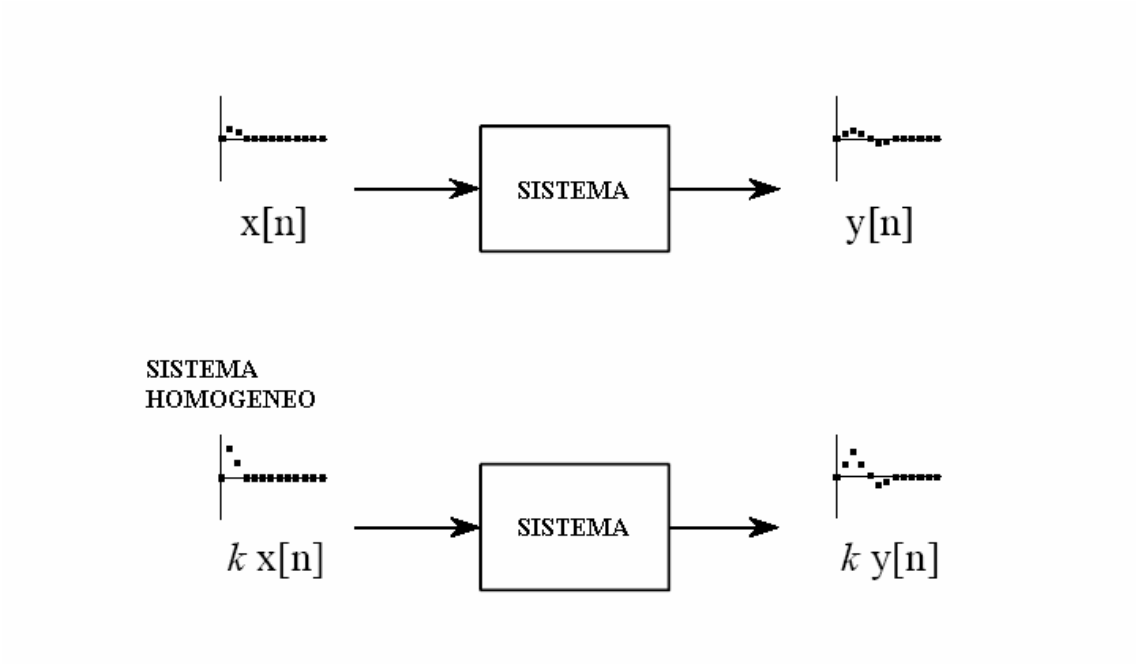
1.2.1 Procesamiento lineal de Imágenes

Un sistema es un proceso que produce una señal de salida en respuesta a una señal de entrada, y se le llama “lineal” a un sistema si este tiene dos propiedades matemáticas: Homogeneidad y Aditividad.

1.2.1.2 Sistemas lineales

Se dice que un sistema es homogéneo si el cambio de amplitud en la entrada del sistema resulta en un correspondiente cambio en la amplitud de la señal de salida. En términos matemáticos, si la señal de entrada de x[n] resulta en una señal de salida de y[n[, entonces una entrada de kx[n] resultara en una salida ky[n], para cualquier señal de entrada y para cualquier constante k.

Figura 3. **Homogeneidad en un sistema lineal**



Para ejemplificar la aditividad en un sistema lineal, consideremos un sistema en donde una entrada $x_1[n]$ produce una salida $y_1[n]$ y una entrada $x_2[n]$ produce una salida $y_2[n]$. El sistema será Aditivo si para la entrada al sistema de las señales $x_1[n] + x_2[n]$ este presenta una salida $y_1[n] + y_2[n]$.

Los sistemas lineales juegan un papel importante en la cuando se crean modelos de Degradación-Restauración de imágenes, modelos que nos permiten comprender el efecto y comportamiento del ruido en la imagen. así mismo las operaciones lineales son excepcionalmente importantes en el procesamiento de imágenes, este esta basado en dos técnicas en especial así como en el PDS convencional, estas son: la *convolución* y el *análisis de Fourier*. El estudio de estos dos conceptos nos proporciona las principales herramientas en el procesamiento y restauración de imágenes además de proveernos de una visión más amplia acerca de las características de las imágenes digitales.

1.2.1.3 Operadores Lineales

Sea H un operador cuya entrada y salidas son imágenes. Se dice que H es un operador Lineal si para cualquiera de dos imágenes f y g y para cualquiera dos escalares a y b:

$$H(af + bg) = aH(f) + bH(g) \quad (1.3)$$

En otras palabras, el resultado de aplicar un operador lineal a la suma de varias imagines es idéntica al resultado de aplicar el operador a las imágenes de manera individual, multiplicar los resultados por las constantes apropiadas y sumar el resultado.

1.2.2 Convolución

Se le denomina convolución a una función, que de forma lineal y continua, transforma una señal de entrada en una nueva señal de salida. La función de convolución se expresa mediante el símbolo *.

En un sistema unidimensional, se dice que g(x) convoluciona f(x) cuando

$$f(x)*g(x) = \int_{-\infty}^{\infty} f(x')g(x-x')dx' \quad (1.4)$$

En el caso de una función continua, bidimensional, como es el caso de una imagen monocroma, la convolución de f(x,y) por g(x,y) será:

$$f(x,y)*g(x,y) = \int_{-\infty}^{\infty}\int_{-\infty}^{\infty} f(x'-y')g(x-x',y-y')dx'dy' \quad (1.5)$$

Y en un sistema discreto, como el de las imágenes digitalizadas la convolución de la función f(x,y) por g(x,y) en la que g(x,y) es una matriz de M filas por N columnas, es:

$$f(x,y)*g(x,y) = \sum_{m=0}^{M-1}\sum_{n=0}^{N-1} f(m,n)g(x-m,y-n) \qquad (1.6)$$

Donde x = 0,1,2,....., M y y = 0,1,2,..., N.

1.2.3 La Transformada de Fourier y el dominio de la frecuencia

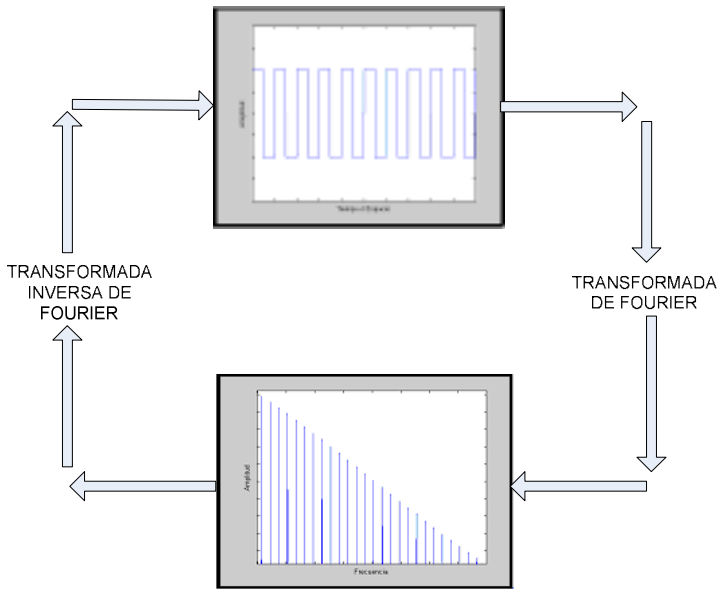
Una transformada puede definirse simplemente como el mapeo a partir de un sistema de coordenadas a otro. Por ejemplo, la rotación de una imagen es una transformada, este sistema es distinto del sistema original pero cada coordenada en la imagen original tiene una coordenada correspondiente en la imagen rotada. La transformada codifica la información de una imagen completamente y puede ser exactamente reconstruida. Aunque existen muchos tipos diferentes de transformaciones de imágenes que pueden utilizarse, la más utilizada es la Transformada de *Fourier*; especialmente debido a un algoritmo de programación muy eficiente conocido como la Transformada Rápida de *Fourier*.

El uso de la Transformada de *Fourier* tiene muchas aplicaciones en la vida real y es una herramienta esencial en la restauración de imágenes, la transformada de *Fourier* convierte coordenadas espaciales (en el caso de las imágenes) en frecuencias.

El Teorema de *Fourier* nos declara que nos es posible formar una función cualquiera unidimensional f(x) como una serie de sumas de términos senoidales y cosenoidales, y el dominio de la frecuencia debe su nombre a que los dos parámetros de una curva senoidal son la amplitud y la frecuencia. Podemos decir con toda certeza que la Transformada de *Fourier* es el método matemático utilizado para moverse dentro y fuera del dominio de la frecuencia.

El hecho de que una imagen se pueda convertir en una representación del dominio de la frecuencia implica que la imagen puede contener información de alta frecuencia o de baja frecuencia. Por ejemplo, si el nivel gris de una cierta porción de una determinada imagen cambia lentamente a través de las columnas, entonces sería representado en el domino de la frecuencia como función del seno o del coseno teniendo una frecuencia baja. Una cierta cosa que cambia rápidamente, por ejemplo un borde, tendrá componentes de alta frecuencia.

Figura 4. **Transformada de *Fourier* y Transformada Inversa de *Fourier***



Si conocemos la información en frecuencia contenida en la imagen, es posible entonces diseñar los filtros necesarios para la restauración de la imagen.

1.2.3.1 La Transformada de Fourier

La Transformada de *Fourier* de una función f(x) describe la cantidad de cada termino de frecuencia que debe ser agregada para describir la forma de f(x) y esta se encuentra dada por la siguiente ecuación:

$$F(w) = \int_{-\infty}^{\infty} f(x) e^{-jwx} dx \qquad (1.7)$$

En donde el uso de la notación exponencial viene dada por el uso de la identidad de euler:

$$e^{-jwx} = \cos(wx) - jsen(wx) \qquad (1.8)$$

Una de las características más importantes de la Transformada de *Fourier*, es que dada F(u), es posible recuperar la función que se encuentra en el dominio espacial, esto es f(x), mediante:

$$f(x) = \int_{-\infty}^{\infty} F(w) e^{jwx} dx \qquad (1.9)$$

Esta última ecuación es la representación de la Transformada Inversa de *Fourier*, que nos permite como ya lo mencionamos en el párrafo anterior, recuperar nuestra señal original.

1.2.3.2 La Transformada Discreta de Fourier

Hemos mostrado la representación de la Transformada de *Fourier* y la Transformada Inversa de *Fourier* para funciones continuas pero nos es necesario definir la representación de estas como funciones discreta. La Transformada de *Fourier* Discreta se define como:

$$F(u) = \frac{1}{M}\sum_{x=0}^{M-1} f(x)e^{\frac{-j2\Pi ux}{M}} \qquad \textit{para u = 0, 1, 2....., M – 1} \qquad \textit{(1.10)}$$

Por lo que podemos observar, la integral se convierte en la suma de todos los puntos muestreados. Para el caso de la Transformada inversa, esta viene dada por la siguiente ecuación:

$$f(x) = \sum_{u=0}^{M-1} F(u)e^{\frac{j2\Pi ux}{M}} \qquad \textit{para x = 0,1,2,.....,M – 1} \qquad \textit{(1.11)}$$

Debido a las propiedades matemáticas de la Transformada de *Fourier*, esta nos permite realizar extensiones para evaluar funciones en una dimensión, dos dimensiones o incluso tres dimensiones dependiendo de la aplicación deseada, esto es posible realizando las sumatorias (en el caso discreto) sobre dos o tres variables (como sea el caso) en vez de sobre una. Ya que en el plano espacial x ý y son ortogonales, también lo son las dimensiónales u ý v, lo que nos permitirá transformar separadamente en cada dirección. En el caso de una imagen bidimensional, es posible realizar una transformación unidimensional para cada línea horizontal de la imagen produciendo así un resultado intermedio con un valor complejo para cada punto, luego se realizaría una segunda transformación en cada columna vertical dándonos estas el resultado de una transformación bidimensional.

Por lo tanto la Transformada de *Fourier* de una imagen f(x,y) de tamaño M X N esta dada por medio de la siguiente ecuación:

$$F(u,v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x,y) e^{-j2\Pi(ux/M + vy/N)} \qquad (1.12)$$

Al igual que en el caso unidimensional, esto es correcto si se trabaja para los valores de u = 0, 1, 2,....,M – 1; así como también para los valores de v = 0,1,2...,N – 1. De manera similar la Transformada Inversa de *Fourier* viene dada por la expresión:

$$f(x,y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u,v) e^{j2\Pi(ux/M + vy/N)} \qquad (1.13)$$

Esto es válido para x = 0,1,2,..., M – 1 y para y = 0,1,2,.....,N – 1. Estas dos ecuaciones se conocen como “Par de Transformadas Discretas de *Fourier* Bidimensionales”. Las variables u y v son las transformadas o variables de frecuencia.

1.2.3.3 Transformada rápida de Fourier

Una de las razones por la cual la Transformada Discreta de *Fourier* se ha convertido en una herramienta esencial en el procesamiento de señales ha sido debido al desarrollo de la Transformada Rápida de *Fourier*.

La Transformada Rápida de *Fourier* es simplemente un algoritmo que nos permite calcular de manera más rápida la Transformada Discreta de *Fourier*.

Este algoritmo es increíblemente eficiente reduciendo el tiempo de computo, si esta no estuviera disponible muchas de las técnicas utilizadas en el procesamiento digital de señales no serian practicas. Las optimizaciones necesarias para acelerar el cálculo son trucos de programación en parte estándares (tales como calcular algunos de los valores por adelantado y fuera del bucle en el programa) y técnicas en parte matemáticas.

Si evaluáramos la Transformada de *Fourier* de M puntos, se requiere directamente del orden de M^2 operaciones aritméticas para obtener el resultado, la Transformada Rápida de *Fourier* nos permite obtener el mismo resultado calculando la misma operación en únicamente MlogM operaciones.

Ya que la Transformada Inversa de *Fourier* es la misma que la Transformada de *Fourier*, únicamente con el signo opuesto en el exponente y un factor de 1/M, el algoritmo de la Transformada Rápida de *Fourier* puede ser adaptado de una manera muy fácil para su cálculo.

2. DISTORSIÓN EN IMÁGENES DIGITALES

El propósito de la Restauración de Imágenes es el de “reconstruir” o “recuperar” una determinada imagen teniendo un conocimiento previo del fenómeno de degradación. Aunque esto no puede realizarse de manera perfecta, las técnicas de restauración de imágenes permiten mejorar la calidad de la información contenida en una imagen.

Este proceso de degradación en una imagen generalmente es modelado por medio de una “Función de Degradación“, que junto con los efectos del “Ruido Aditivo” opera sobre una imagen f(x, y) para producir una imagen degradada g(x, y). Si se cuenta con suficiente información acerca del los efectos de un determinado fenómeno degradante en la imagen, es posible mediante técnicas matemáticas crear modelos que permitan analizar la imagen degradada a fin de poder desarrollar procedimientos para restaurar la imagen. Es decir, si ya se cuenta con la imagen degradada g(x, y) y si se tiene conocimiento acerca de algunas propiedades de la Función de Degradación y del ruido que contamina la imagen, es posible obtener un estimado de la imagen original. Mientras más exacto sea el conocimiento de la Función de Degradación y del ruido que contamina la imagen, más cercana estará la estimación de la imagen restaurada a la imagen original.

2.1 Modelo para el proceso de distorsión de una Imagen

Si la Función de Degradación de una imagen es un proceso lineal, entonces la imagen degradada se representa en el dominio espacial por:

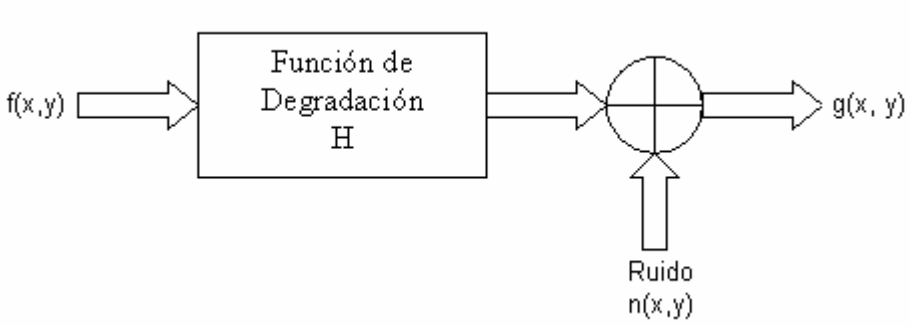
$$g(x, y) = h(x, y)*f(x, y) + n(x, y) \qquad (2.0)$$

Donde h(x, y) representa la Función de Degradación y n(x, y) representa el ruido aditivo que afecta a la imagen adquirida. Esta ecuación puede representarse de otra manera recordando que la convolución en el dominio espacial es igual a la multiplicación en el dominio de la frecuencia, por lo tanto la ecuación anterior puede representarse también por:

$$G(x, y) = H(x, y)F(u, v) + N(u, v) \qquad (2.1)$$

Estas ecuaciones pueden modelarse mediante el siguiente sistema de bloques mostrado en la figura 5, de esta manera es representada la degradación de una imagen de acuerdo a los efectos de la Función de Degradación y efectos del Ruido Aditivo.

Figura 5. **Modelo para la degradación de una imagen**



2.2 Ruido en Imágenes y su modelación

Entre las principales fuentes de ruido presentes en una imagen digital podemos mencionar el ruido debido a la digitalización de la imagen por causa del desempeño de los sensores utilizados para la digitalización. Estos sensores pueden dañarse por diversos motivos, entre ellos puede mencionarse la calidad, condiciones climáticas y hasta vejez por mencionar solo algunos ejemplos. Así mismo otra fuente importante de ruido es el causado por interferencia cuando la imagen digital es transmitida a través de un canal de transmisión, si por ejemplo la imagen es transmitida a través de un sistema inalámbrico es posible que exista interferencia electromagnética debida a fenómenos atmosféricos que podrían agregar ruido a la imagen digital.

Lo que en realidad es de interés en el estudio del ruido presente en las imágenes digitales, es el comportamiento estadístico del componente de ruido en la escala de grises de la imagen, este comportamiento estadístico puede considerarse como una variable aleatoria caracterizada por una "Función de Densidad" de probabilidad.

La función de densidad se utiliza en la ciencia estadística con el propósito de conocer cómo se distribuyen las probabilidades de un evento en relación al resultado del evento.

Entre las funciones que permiten modelar el ruido con el fin de comprender y estudiar los efectos del mismo en las imágenes se encuentran la distribución gaussiana, la distribución uniforme, la distribución gamma, la distribución exponencial y la función de impulso llamada también "sal y pimienta", por mencionar las más utilizadas.

2.2.1 Ruido Gaussiano

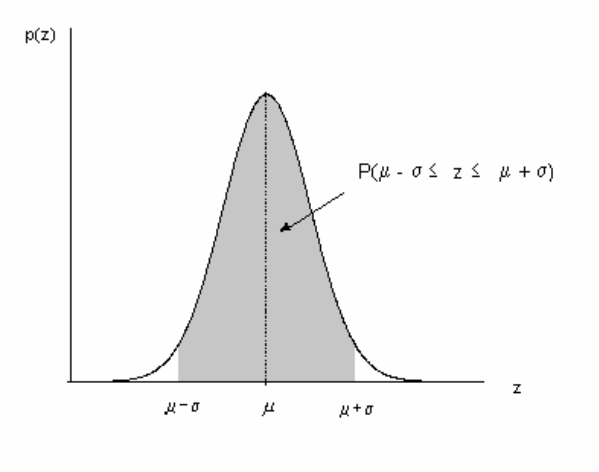
Se le llama gaussiano debido a que la distribución del ruido es semejante a una distribución gaussiana de una determinada media y varianza. Puede decirse que esta distribución es la distribución continua de probabilidad más importante en todo el campo de la estadística y su curva describe con mucha exactitud muchos de los fenómenos que ocurren en la naturaleza, entre estos se encuentran los efectos del ruido en una imagen dada. Los modelos de ruido basados en esta distribución son utilizados frecuentemente debido a su conveniencia matemática. El ruido gaussiano tiene una función de densidad dada por la ecuación:

$$p(z) = \frac{1}{\sqrt{2\pi\sigma}} e^{-(z-\mu)^2/2\sigma^2} \qquad (2.2)$$

En donde z representa el nivel de gris, μ es la media del valor de z y σ es la desviación estándar. A partir de la ecuación anterior puede deducirse que aproximadamente el 70% de los datos de la imagen se encuentran en el intervalo ($\mu - \sigma$, $\mu + \sigma$), mientras que el 95% de los datos se encontrara en el intervalo ($\mu - 2\sigma$, $\mu + 2\sigma$).

La grafica de la distribución gaussiana se muestra en la Figura 6

Figura 6. **Distribución Gaussiana**



Fuente: Walpole & Myers, "Probabilidad y Estadística para Ingenieros, Pearson Education 1999.

2.2.2 Ruido Gamma

Se le conoce así debido a que su comportamiento puede modelarse mediante la llamada distribución gamma, esta deriva su nombre de la bien conocida función gamma que se estudia en muchas de las áreas de la matemática. El ruido gamma tiene una función de densidad dada por la siguiente ecuación:

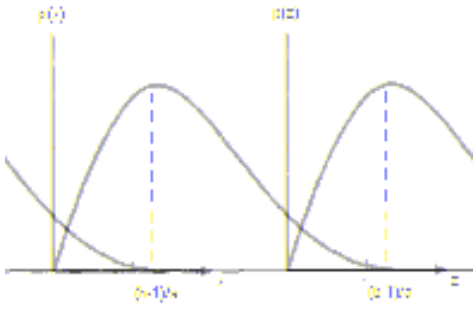
$$p(z) = \begin{cases} \dfrac{a^b z^{b-1}}{(b-1)!} e^{-az} & para \quad z \geq a \\ 0 & para \quad z < a \end{cases} \qquad (2.3)$$

En donde a es un número mayor que cero y b es un entero positivo, la media y la varianza de esta función de densidad están dadas por las siguientes ecuaciones:

$$\mu = \frac{b}{a} \quad ; \quad \sigma^2 = \frac{b}{a^2} \tag{2.4}$$

La grafica de esta distribución se muestra en la figura 7.

Figura 7. Distribución Gamma



Fuente: Walpole & Myers, "Probabilidad y Estadística para Ingenieros, Pearson Education 1999.

2.2.3 Ruido Exponencial

La Función de densidad para el ruido exponencial esta dada por la siguiente ecuación:

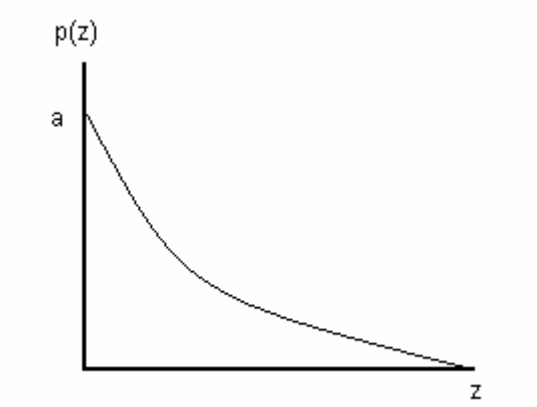
$$p(z) = \begin{cases} ae^{-az} & para \quad z \geq 0 \\ 0 & para \quad z < 0 \end{cases} \tag{2.5}$$

La media y la varianza para esta función de densidad están dadas por las ecuaciones siguientes

$$\mu = \frac{1}{a} \qquad ; \qquad \sigma^2 = \frac{1}{a^2} \tag{2.6}$$

Y su grafica esta dada por la siguiente figura

Figura 8. **Distribución Exponencial**



Fuente: Walpole & Myers, "Probabilidad y Estadística para Ingenieros, Pearson Education 1999.

2.2.4 Ruido Uniforme

Como en los anteriores casos el Ruido Uniforme se modela de acuerdo a una distribución estadística en particular, la distribución uniforme es una distribución de probabilidad cuyos valores tienen la misma probabilidad. La función de densidad del Ruido Uniforme esta dada por la siguiente ecuación:

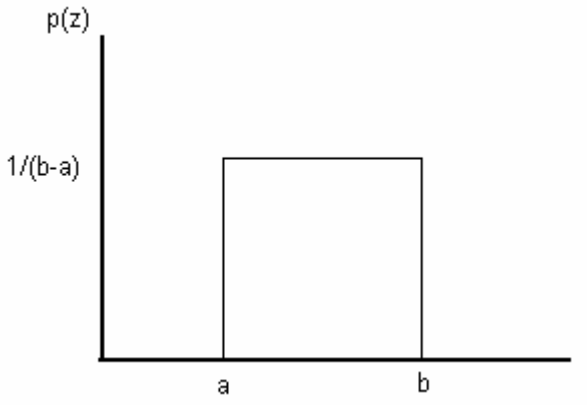
$$p(z) = \begin{cases} \frac{1}{b-a} & si \quad a \le z \le b \\ 0 & \text{en cualquier otro caso} \end{cases} \qquad (2.7)$$

La media y la varianza para esta función de densidad están dadas por las siguientes ecuaciones

$$\mu = \frac{a+b}{2} \qquad ; \qquad \sigma^2 = \frac{(b-a)^2}{12} \qquad (2.8)$$

La gráfica de la función de densidad del Ruido Uniforme es mostrada en la figura 9

Figura 9. **Densidad Ruido Uniforme**



El ruido uniforme puede utilizarse para generar cualquier otro tipo de distribución de ruido y es comúnmente utilizado con el fin de degradar imagen para la evaluación de algoritmos de restauración de imágenes digitales.

2.2.5 Ruido de Impulso

Es causado principalmente por fallas en el funcionamiento de los sensores encargados de capturar una imagen o por errores de tiempo cuando se produce el proceso de digitalización de la imagen. Su función de densidad esta dada por la ecuación:

$$p(z) = \begin{cases} P_a & para \quad z = a \\ P_b & para \quad z = b \\ 0 & \text{en cualquier otro caso} \end{cases} \tag{2.9}$$

Si b>a, puntos claros aparecerán en la imagen y si a>b aparecerán puntos oscuros corrompiendo la imagen. En el caso de que P_a y P_b sean cero, el Ruido de Impulso es llamado “unipolar” y si ninguna de las probabilidades es cero y especialmente si son aproximadamente iguales, la imagen se encontrara contaminada por puntos claros y oscuros por lo cual este tipo de ruido recibe el nombre de “Sal y Pimienta”.

Al momento de digitalizar una imagen este tipo de ruido generalmente toma valores extremos en la imagen (cercanos a los valores del negro o muy cercanos a los valores del blanco), esto es debido a que los impulsos de ruido pueden negativos o positivos y generalmente suele suponerse que los valores de a y b se encuentran “saturados”, ya sea en sus valores máximos o en sus valores mínimos cuando se digitaliza la imagen. Para una imagen de 8 bits esto significaría que a=0 (negro) y b= 255 (blanco).

Con el fin de poder observar los efectos del ruido gaussiano en una imagen dada (Figura 10a) es posible utilizar el programa de computo “*Matlab*” y su función “imnoise” para poder agregar ruido gaussiano y también ruido de impulso a una imagen digital como se muestra en la Figura 10b, y en la Figura 10c respectivamente.

Figura 10. Efectos del Ruido en las imágenes digitales

a

b

c

2.2.6 Ruido Periódico

El ruido periódico en una imagen se origina generalmente debido a interferencia eléctrica o electromecánica durante la adquisición de la imagen. Debido a sus características, el ruido periódico puede reducirse significativamente mediante filtrado en el dominio de la frecuencia ya que el ruido periódico tiende a producir picos de frecuencia en el espectro de *Fourier* que son fácilmente identificables.

2.3 La Función de Degradación

Distorsión en una imagen puede ocurrir en forma de un efecto borroso debido a fluctuaciones de la señal durante las mediciones en el intervalo de tiempo cuando la imagen es adquirida, también puede darse debido a lentes imperfectos, movimiento de los objetos o dispositivos al momento de la captura de la imagen y también debido a la cuantización espacial de la imagen, un ejemplo se muestra en la Figura 11.

Figura 11. **Efecto borroso en una imagen**

La degradación de una imagen puede ser “espacialmente invariante” o “espacialmente variante”. La degradación espacialmente invariante afecta a todos los píxels de igual forma en una imagen y ejemplos de esta pueden verse por mal enfoque de las cámaras, movimiento de cámaras al momento de tomar la fotografía y también puede ser causada el poco tiempo de exposición al tomar la imagen.

La degradación espacialmente variante no afecta a todos los *pixels* por igual, por lo que es mucho más difícil de modelar que una degradación espacialmente invariante; aunque esta degradación puede modelarse como una degradación espacialmente invariante sobre regiones muy pequeñas.

2.3.1 Estimación de la Función de Degradación

Existen tres métodos principales para estimar la función de degradación a utilizar para la restauración de una imagen:

1) Observación
2) Experimentación
3) Modelación matemática

2.3.1.1 Estimación mediante observación

Si se cuenta con una imagen degradada sin tener ningún conocimiento acerca de la función de degradación H. Una forma de estimar esta función es recopilando información de la imagen misma; con el fin de hacer esto de una manera correcta suele observarse pequeñas secciones de la imagen empezando por las secciones que cuentan con estructuras simples (como por ejemplo partes del fondo de la imagen o partes de algún objeto en la imagen) y en las que el ruido sea mínimo. En el caso de que la imagen se encuentre borrosa, mediante observación puede crearse una imagen reconstruida del área observada (que realmente es un estimado de la imagen original en el área observada).

Si se denota como g(x, y) a la imagen borrosa y a la imagen reconstruida $f^1(x, y)$. Entonces es posible asumir que la función de degradación estará dada en ausencia del ruido mediante la ecuación:

$$H_s = \frac{G_s(u,v)}{F_s^1(u,v)} \tag{2.10}$$

De las características de la ecuación anterior, puede deducirse H(u,v) en el caso de contar con una degradación espacialmente invariante.

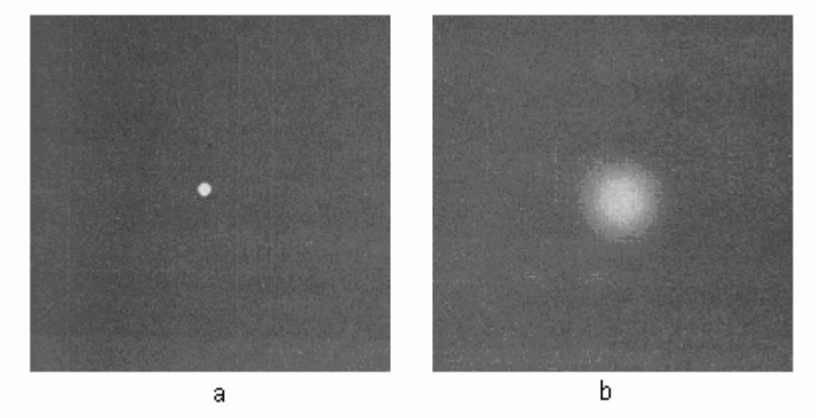
2.3.1.2 Estimación por experimentación

Cuando existe la disponibilidad de contar con equipo similar al que genero la imagen degradada, generalmente es posible determinar con mucha exactitud la naturaleza de esta degradación. El proceso de experimentación consiste en tomar una imagen similar a la imagen degradada con el mismo equipo modificando algunos ajustes en el, el objetivo es acercarse lo mas posible a la imagen original. Luego la idea consiste en obtener una respuesta impulso a la degradación capturando la imagen de un pequeño punto de luz utilizando la misma configuración en el equipo. Esto es debido a que la Transformada de *Fourier* de un impulso es constante por lo que la función de degradación estará dada mediante la siguiente ecuación:

$$H(u,v) = \frac{G(u,v)}{A} \tag{2.11}$$

Esto puede observarse mejor mediante la Figura 12

Figura 12. **Respuesta impulso en una imagen**



Fuente: R. González, “Digital Image Processing” , Pearson Education 2002.

En la figura 12a se muestra la imagen de un punto de luz brillante, y en la figura 12b se muestra la misma imagen degradada.

2.3.1.3 Estimación mediante Modelación Matemática

La estimación matemática ha sido utilizada por muchos años debido a que provee comprensión acerca del fenómeno de degradación en una imagen. Se utiliza principalmente para la modelación del efecto borroso en una imagen debido al movimiento y en modelos de turbulencia atmosférica utilizados en astronomía.

Los modelos matemáticos, el análisis de la imagen y mucha experimentación combinado con un claro entendimiento de la matemática hacen posible estimar de manera exitosa la función de degradación.

3. RESTAURACIÓN DE IMÁGENES, EL MÉTODO DE FILTRADO INVERSO Y EL MÉTODO DE FILTRADO *WIENER*

La Restauración de imágenes permite remover ciertas imperfecciones que se encuentran en la imagen contando para ello con cierto conocimiento previo del fenómeno que ha causado que esta contenga ruido, presente distorsiones o ambas. Esto es posible a través de varios métodos que han sido desarrollados a través del tiempo y que varían dependiendo de la aplicación.

Con base a lo expresado en el párrafo anterior, los métodos utilizados para la restauración de imágenes reciben el nombre de “Objetivos”, -a diferencia de los métodos que permiten modificar una imagen para resaltar ciertas características para la extracción de determinada información y que por lo tanto reciben el nombre de “Subjetivos”- ya que se enfocan en la recuperación de la información contenida en la imagen, esto se debe a que es posible utilizar conocimiento previo de cómo esta debería verse.

En el presente capítulo se estudiarán 2 de los filtros más utilizados (así como posiblemente los más estudiados) en la restauración de imágenes digitales, estos son:

- Método del Filtrado Inverso
- Método del Filtrado de *Wiener*

El objetivo de estos es el de obtener una imagen estimada f'(x,y) a través del filtrado de la imagen observada, que minimizara los efectos de la degradación (eliminar la borrosidad, eliminar ruidos, etc.) a través del análisis en frecuencia de la imagen y la comprensión de los fenómenos que causan la perdida de información en la imagen

Para lograr lo anterior, es necesaria la comprensión de las herramientas matemáticas a utilizar, tales como el análisis matricial y el análisis de *Fourier* (conceptos que se han estudiado en el Capítulo 1) así como también el estudio y comprensión de fenómenos tales como el ruido presente en la imagen (estudiado en el Capítulo 2).

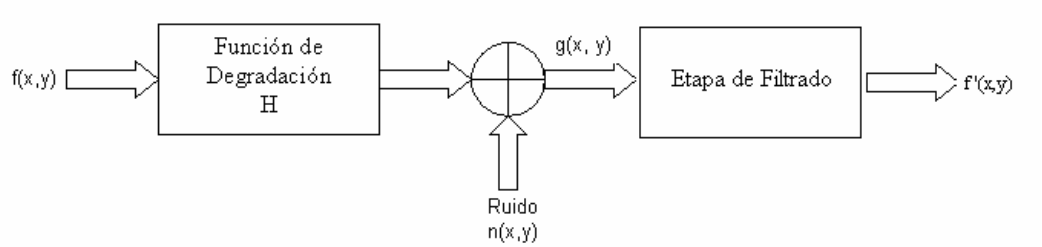
Previo al estudio de ambos métodos de Restauración es necesario exponer el Modelo de Degradación/Restauración de una imagen, el cual será de gran utilidad cuando se expongan los conceptos relacionados al Filtrado Inverso y el Filtrado de *Wiener*.

3.1 Modelo de Degradación/Restauración de una Imagen

En el capítulo anterior se presento el modelo para la degradación de una imagen digital, en donde f(x,y) representaba la imagen digitalizada, h(x,y) la función de degradación y n(x,y) representaba el ruido que contamina la imagen.

A continuación se presente al Modelo de Degradación/Restauración que incluye una etapa de Filtrado que permite eliminar el Ruido contenido en la Imagen ó corregir a cierto nivel los efectos de la degradación en la imagen digital.

Figura 13. **Modelo de Degradación/Restauración de una imagen**



En el caso del modelo mostrado,

f(x,y) = imagen original
g(x,y) = imagen distorsionada
n (x,y) = ruido aditivo
g'(x,y) = imagen restaurada

En base al modelo podemos observar que la imagen distorsionada se forma cuando f(x, y) es afectada por una función de degradación y posiblemente por ruido aditivo. La imagen restaurada resulta luego de que g(x, y) pasa a través de una etapa de filtrado.

Es necesario recalcar el hecho de que no es posible restaurar al 100% una imagen distorsionada. Esto se debe a varias limitaciones que pueden ir desde la falta de información acerca de la función de degradación, hasta el caso en que cierta parte de la información original no se encuentre presente ya en la imagen degradada. Aún así, algunos de los métodos que han sido desarrollados hasta el día de hoy para completar los procesos que permiten la restauración de imágenes, logran una aproximación bastante fiel de la imagen original utilizando programas de computación diseñados específicamente para este fin, algunos de estos programas necesitan de computadoras con una gran capacidad de procesamiento como por ejemplo las computadoras que se encargan de procesar y analizar las fotografías tomadas por telescopios espaciales como el Hubble, telescopio utilizado para explorar regiones del espacio que se encuentran a gran distancia.

Los diversos tipos de ruido que pueden estar presentes en una imagen se han presentado en el capítulo 2, así como también los efectos de la distorsión en la imagen y la Función de Degradación, por lo que a continuación se presentan dos de los Métodos para Restauración de Imágenes más utilizados en la actualidad.

Es necesario reconocer que para la implementación de nuestra parte de estos Métodos de Restauración, es necesario el uso de programas de cómputo que faciliten la cantidad de cálculos matemáticos necesarios, en especial las operaciones con matrices. Tal es el caso del paquete de computo "*Matlab*", que además de manejar análisis matricial a un nivel muy avanzado, proporciona una serie de herramientas matemáticas que resultan muy útiles para el análisis de Imágenes digitales (se pueden mencionar las herramientas para el análisis de *Fourier*), como veremos más detalladamente en el ultimo capítulo, donde nos enfocaremos al diseño e implementación de un algoritmo de programación para el Método de Filtrado Inverso y el Método de Filtrado de *Wiener*.

3.2 El Método de Filtrado Inverso

Entre las diferentes técnicas desarrolladas para llevar a cabo la restauración de imágenes digitales, el método del filtrado inverso sobresale debido a que su aplicación permite una primera aproximación sobre la imagen original, para imágenes que se encuentran borrosas o fuera de foco.

Este método se basa en obtener un estimado de la imagen original dividiendo la transformada de la imagen degradada por una función de degradación conocida:

$$F'(u,v) = \frac{G(u,v)}{H(u,v)} \qquad (3.0)$$

En donde

F’(u,v) = Estimación de la Imagen original
H(u,v) = Transformada de *Fourier* de la Función de Degradación
G(u,v) = Transformada de *Fourier* de la imagen degradada

La función de degradación h(u,v) se obtiene en la mayoría de casos, mediante observación y experimentación (tal y como se explico en el capítulo 2). Mientras se tenga un mejor conocimiento del fenómeno que causa la degradación, será mejor la respuesta del filtro inverso.

Si recordamos que g(u,v) representa la imagen degradada y que su transformada de *Fourier* es igual a G(u,v) = H(u,v)F(u,v) + N(u,v) tendremos que sustituyendo en la ecuación anterior esta se transforma en

$$F'(u,v) = F(u,v) + \frac{N(u,v)}{H(u,v)} \qquad (3.1)$$

Esta ecuación indica que si se conoce con cierta exactitud la función de degradación de la imagen y esta no contiene ruido (es decir n(u,v) = 0), será posible tener una buena aproximación de la imagen original.

El Método de Filtrado Inverso resulta particularmente útil debido a que presenta una mayor sencillez (si se puede llamar así) al momento de su aplicación, en comparación con otros métodos de restauración.

El Método de Filtrado Inverso también presenta varias limitaciones y se suele utilizar en los siguientes casos:

- Imágenes que se encuentran fuera de foco
- Imágenes que se encuentran libres de contaminación por ruido

En la siguiente figura puede apreciarse un ejemplo de cómo el Filtro Inverso permite restaurar una imagen que se encuentra fuera de foco.

Figura 14 Ejemplo de una imagen restaurada luego de la etapa de filtrado

Fuente: www.fesb.hr/khoros/dipcourse/dip/html-dip/c7/s1/front-page.html

Si por el contrario existiera ruido presente en la imagen a analizar, es muy probable que no sea posible recuperar la imagen distorsionada de buena forma, ya que el ruido presente en cualquier imagen es una función aleatoria cuya transformada de *Fourier* generalmente no se conoce. El efecto de aplicar el Filtro Inverso en una imagen que contiene ruido puede observarse especialmente si el ruido se presenta en alta frecuencia.

Lo anterior puede explicarse estudiando la ecuación del Filtro Inverso y observando que este tiene la forma de un Filtro Pasa Altos.

3.2.1 Filtro Pseudoinverso

Uno de los inconvenientes que presenta el método del Filtrado Inverso se produce cuando los valores de la función de degradación se encuentran cercanos a cero, es posible observar que la función de estimación tendrá valores que se acercarán al infinito, haciendo que la función sea inestable. Una forma de solucionar esto es limitando la función de degradación, no permitiendo que esta contenga valores cercanos a cero o nulos. En este caso la función de degradación se tendrá que definir de la siguiente manera:

$$H'(u,v) = \begin{cases} \dfrac{1}{H(u,v)} & H(u,v) \neq 0 \\ 0 & H(u,v) = 0 \end{cases} \qquad (3.1)$$

Donde

H’(u,v) = Función de Degradación delimitada

Esta es una forma sencilla de definir nuevamente la Función Degradación, en aplicaciones donde se requiere mayor exactitud, a los valores cercanos a cero (no necesariamente cero) se les asigna un valor, el cual generalmente se define conociendo las características de la imagen que se trabajará.

Para poder aplicar la anterior ecuación es posible recurrir a la ayuda de *Matlab*, el cual permite declarar este tipo de funciones en un algoritmo para lograr así la aplicación deseada. Es importante destacar que al momento en que se define de nuevo la Función Degradación como en la ecuación anterior el método de filtrado pasa a llamarse “Método de Filtrado PseudoInverso”.

3.3 Método de Filtrado de *Wiener*

El Filtro de *Wiener* lleva su nombre gracias al Matemático Norbert *Wiener* quien lo planteo en el año de 1942 y a partir de su formulación ha sido utilizado en áreas tales como el Procesamiento de Audio y Procesamiento de Imágenes digitales, debido a que permite obtener resultados óptimos al momento de su aplicación.

El método de restauración basado en el Filtrado de *Wiener*, consiste al igual que con el Filtro Inverso, obtener una imagen estimada, Su propósito principal es el de retirar el ruido que ha contaminado determinada señal utilizando como base la estadística, el Filtro de *Wiener* pretende obtener una imagen estimada, tal que se minimice el error cuadrático medio.

3.3.1 Error Cuadrático Medio

Se le llama Error Cuadrático Medio al estimador que indica el valor esperado del "error" al cuadrado, donde el "error" en este caso representa la cantidad por la cual la imagen estimada difiere de la imagen original, esta diferencia ocurre en el proceso de restauración de imágenes debido a que existen ciertos parámetros que no se conocen, por ejemplo, podemos mencionar el hecho de que el ruido presente en una imagen sea una función aleatoria y por lo tanto no conocida del todo y que por lo tanto su porcentaje extracción en una determinada imagen que se encuentra corrompida no sea del 100%.

El Error Cuadrático Medio se define mediante la siguiente ecuación:

$$e^2 = E\{(f'(x,y) - f(x,y))^2\} \qquad (3.2)$$

En donde

e^2 = Error Cuadrático Medio

E{} = Valor esperado del argumento

f = matriz que contiene los datos de la imagen digital

f' = estimado de la imagen original f

Para comprender la importancia del ECM y de acuerdo con el Profesor Bernardo A. Calderón de la Universidad de Antioquia en Colombia (http://bochica.udea.edu.co/~bcalderon/3_propiedadesestimadores.html) es posible expresar la ecuación anterior de la siguiente manera:

$$e^2 = E(f'(x,y)^2-2f(x,y)f'(x,y)+f(x,y)^2) \quad (3.3)$$

Que a su vez puede reescribirse como

$$e^2 = E(f'(x,y)^2) -2f(x,y)E(f'(x,y)) + f(x,y)^2 \quad (3.4)$$

Si se suma y resta $[E(f'(x,y))]^2$ en ambos lados de la ecuación se tiene que:

$$e^2 = \{E(f'(x,y)^2)-[E(f'(x,y))]^2\} + \{[E(f'(x,y))]^2-2f(x,y)E(f'(x,y))+ f(x,y)^2\} \quad (3.5)$$

$$e^2 = V(f'(x,y)) + [f(x,y)-E(f'(x,y))]^2 \quad (3.6)$$

Donde

V(f'(x,y)) = Varianza muestral de la imagen estimada

$[f(x,y)-E(f'(x,y))]^2$ = el cuadrado de la diferencia entre la imagen estimada y la imagen original (llamado "sesgo")

En base a lo anterior y tomando como base lo referido por el Profesor Bernardo Calderón en su documento acerca de los Estimadores Puntuales, el ECM involucra las dos propiedades más importantes de un estimador, donde la varianza de la imagen estimada y el sesgo deben ser lo más pequeña posible para que, en nuestro caso, la imagen estimada sea una aproximación fiel.

El Error Cuadrático Medio permite por lo tanto evaluar la eficiencia del filtro a utilizar, ya que permite comparar la muestra original con la estimada para obtener el filtro óptimo, el cual se define como aquel que minimiza el ECM. Es decir que a un menor error, lógicamente existirá mayor precisión.

3.3.2 Ecuación del Filtro de *Wiener*

En el apartado anterior se ha aclarado que el objetivo principal del Filtro de *Wiener* es el de obtener un estimado f'(x,y) de la imagen original f(x,y) de manera que el ECM sea mínimo, donde el ECM se representa mediante la ecuación:

$$e^2 = E\{(f'(x,y) - f(x,y))^2\} \qquad (3.7)$$

El verdadero problema radica en encontrar una solución a la ecuación anterior que permita minimizar el ECM, para esto es necesario asumir ya que requiere grandes conocimientos en el área de la Matemática Estadística (que van más allá del objeto de este texto) si no se imponen algunas limitaciones, se asumirá lo siguiente:

- Los valores en la escala de grises de imagen estimada f'(x,y) se considerarán una función lineal de los valores en la escala de grises de la imagen degradada.

- Se asume que el ruido y la imagen original no tienen correlación entre si, es decir que su covarianza es igual a cero (se debe recordar que la covarianza es una medida que indica cuanto cambian dos variables entre si) y además se considera también que ya sea el ruido o la imagen original tendrán media cero. Esto suposición permite expresar entonces lo siguiente:

$$E\{f(x,y)n(x,y)\} = E\{f(x,y)\}\, E\{n(x,y)\} = 0 \qquad (3.8)$$

Teniendo en cuenta lo anterior es posible entonces encontrar un ECM mínimo que cumpla con lo que se ha asumido anteriormente.

Si se considera de nuevo el Modelo de Degradación de una imagen, se tiene que la imagen corrompida o degradada será igual a:

$$g(x,y) = h(x,y)*f(x,y) + n(x,y) \qquad (3.9)$$

En donde como se ha estudiado en el Capítulo 2 h(x,y) representa la función de degradación y n(x,y) el ruido en la imagen. Entonces, el estimado de la imagen original por medio del Filtrado de *Wiener* deberá ser igual a:

$$f'(x,y) = w(x,y)*g(x,y) \qquad (3.10)$$

Donde:

w(x,y) = Filtro de *Wiener*

Si separamos en factores la ecuación del ECM y sustituimos por la ecuación anterior tendremos lo siguiente:

$$e^2 = E[f(x,y)^2] + E[f'(x,y)^2] - 2E[f(x,y)f'(x,y)] \quad (3.11)$$

$$= E[f(x,y)^2] + E[\{w(x,y)*g(x,y)\}^2] - 2E\,[f(x,y)\{w(x,y)*g(x,y)\}] \quad (3.11)$$

Sustituyendo en la ecuación anterior tenemos

$$= E[f(x,y)^2] + E[\{w(x,y)*(h(x,y)*f(x,y) + n(x,y)\}^2] - 2E[f(x,y)\{w(x,y)*(h(x,y)*f(x,y)+n(x,y))\}] \quad (3.12)$$

De acuerdo al libro “Image Processing, principles and applications” de los autores Tinku Acharya y Ajoy K Ray, la ecuación anterior puede expresarse de la siguiente manera:

$$e^2= R_f(x,y) + w(x,y)*h(x,y)*R_f(x,y)*h(x,y)*w(x,y)+w(x,y)*R_n(x,y)*w(x,y)- 2w(x,y)*h(x,y)*R_f(x,y) \quad (3.13)$$

En donde

$R_f(x,y) = E[f(x,y)^2]$

$R_n(x,y) = E(n(x,y)^2]$

Para encontrar el Filtro de *Wiener* Óptimo es necesario igualar e^2 a cero y derivar respecto a [w], si luego se aplica la transformada de *Fourier* al resultado anterior se llegará a la siguiente ecuación:

$$W = \frac{H^*(u,v)S_f(u,v)}{\left|H(u,v)\right|^2 S_f(u,v) + S_n(u,v)} \qquad (3.14)$$

En donde

$S_n(u,v)$ = Densidad Espectral de Potencia del Ruido = $\left|N(u,v)\right|^2$

$S_f(u,v)$ = Densidad Espectral de la Imagen Original = $\left|F(u,v)\right|^2$

H(u,v) = Función de degradación

H*(u,v) = Complejo Conjugado de H(u,v)

Si se divide $S_f(u,v)$ entre el Numerador y el Denominador y si además se toma en cuenta que el producto de una cantidad compleja con su conjugado es igual a la magnitud del complejo al cuadrado, para multiplicar y dividir la expresión por H(u,v), se llegará a la siguiente expresión:

$$W(u,v) = \frac{1}{H(u,v)} \frac{\left|H(u,v)\right|^2}{\left|H(u,v)\right|^2 + S_n(u,v)/S_f(u,v)} \qquad (3.15)$$

Esta última expresión es conocida como el “Filtro de *Wiener*”. Por lo tanto es posible definir nuevamente el estimado de la imagen original de la siguiente manera:

$$F'(u,v) = \left(\frac{1}{H(u,v)} \frac{\left|H(u,v)\right|^2}{\left|H(u,v)\right|^2 + S_n(u,v)/S_f(u,v)} \right) G(u,v) \qquad (3.16)$$

En la ecuación anterior al termino $S_n(u,v)/S_f(u,v)$ se le conoce como la “Relación Señal a Ruido” y representa el margen que existe entre la información en la imagen y el ruido aditivo.

Es necesario tomar en cuenta que en muchos casos no se conoce con exactitud la imagen original de manera que la relación señal a ruido pueda calcularse, por lo que esta se sustituye por una constante “K” y mediante repetidas iteraciones mediante prueba y error se llega al mejor resultado. Por el contrario, si en caso fuera posible tener la matriz de datos de referencia, la relación señal a ruido puede calcularse y no será necesario realizar múltiples iteraciones.

4. DESARROLLO DE ALGORITMOS PARA LA APLICACIÓN DEL FILTRADO INVERSO Y FILTRADO DE *WIENER*

El desarrollo de una aplicación de *software* que permita realizar el proceso de restauración de imágenes distorsionadas es el objeto del presente capítulo, mediante la aplicación de un algoritmo de programación será posible observar la manera en la que el Filtrado Inverso y el Filtrado de *Wiener* permiten recuperar la información contenida en determinada imagen con la ayuda del análisis de *Fourier* y la comprensión de fenómenos como el ruido y la distorsión causada por los efectos borrosos en una imagen.

Las aplicaciones en la vida real son innumerables y pueden ir desde el análisis de imágenes obtenidas por dispositivos de vigilancia y seguridad, el análisis de imágenes en el campo de la medicina (donde se puede combinar con otros procesos para extraer la información deseada en al imagen), hasta la restauración de cualquier imagen corrompida por causas diversas al momento de enviarla como una forma de datos a través de un medio, que puede ser desde cobre, hasta un medio inalámbrico.

En apartados anteriores se ha presentado la flexibilidad de *Matlab* para realizar diversas operaciones de programación gracias a su capacidad para el desarrollo de algoritmos de funciones matemáticas establecidas, lo que permite una mayor facilidad al momento de realizar los procedimientos matemáticos necesarios para la implementación de los filtros y a su vez permite ahorrar una cantidad significativa de tiempo. *Matlab* es también una plataforma que permite desarrollar programas con una adecuada interacción entre el usuario y la pc, esto, a través de la Interfaz Grafica de Usuario (conocida como GUI), utilizada en varios lenguajes de programación visual.

El capítulo se encuentra dividido en 3 secciones, de tal forma que puedan cubrirse todos los conceptos estudiados en el presente trabajo, es decir, primero se describirá la manera en que *Matlab* realiza el proceso de contaminación de la imagen digital, simulando procesos que ocurren de manera natural. Luego se desarrollarán los algoritmos para el procesamiento de imágenes a través del Filtrado Inverso y el Filtrado de *Wiener*.

Para la aplicación de los algoritmos se seguirán utilizando imágenes en escala de grises, ya que proveen una mejor comprensión del proceso de Degradación/Restauración. Debe recordarse que el proceso para la restauración de imágenes a color es igual y debe aplicarse de igual forma para la matriz de datos de intensidad de los colores Rojo, Azul y Verde, contenidos en el arreglo de datos de una imagen RGB.

4.1 Simulación de la Función de Degradación y Ruido Aditivo en Matlab

Con el fin de evaluar el desempeño de los filtros estudiados en el presente trabajo es necesario primero desarrollar una aplicación que sea capaz de modificar la apariencia de las imágenes que se analizarán de una manera similar a lo que sucede en la vida real, basándose en el modelo de Degradación presentado en el presente trabajo. De acuerdo a este modelo, una imagen resulta degradada en cualquiera de los 3 siguientes casos.

1. Mediante convolución de la imagen original por una función de degradación.
2. Mediante la acción de ruido aditivo.
3. Mediante la acción simultanea de las opciones 1 y 2

Matlab permite realizar la adición de ruido aditivo y generar funciones de degradación de acuerdo a modelos establecidos, ó crearla de acuerdo a la información que se tenga de los fenómenos que causan la distorsión. Es posible por lo tanto generar una función de degradación que cause un efecto borroso como el causado ya sea por la corta exposición al momento de la captura de la imagen, o al movimiento al momento de la captura de la imagen con la función “fspecial” la que permite modificar de manera lineal el desplazamiento de los *pixels* en la imagen así como también el ángulo de desfase.

En las siguientes imágenes se muestra el efecto de distorsión generado por la adición de ruido en una imagen afectada por una función de degradación h(x,y), de esta manera es posible ejemplificar de mejor manera el modelo de degradación de la imagen.

Figura 15 a) Imagen Original; b) Imagen afectada por distorsión lineal; c) Imagen afectada por distorsión lineal y ruido gaussiano

a)

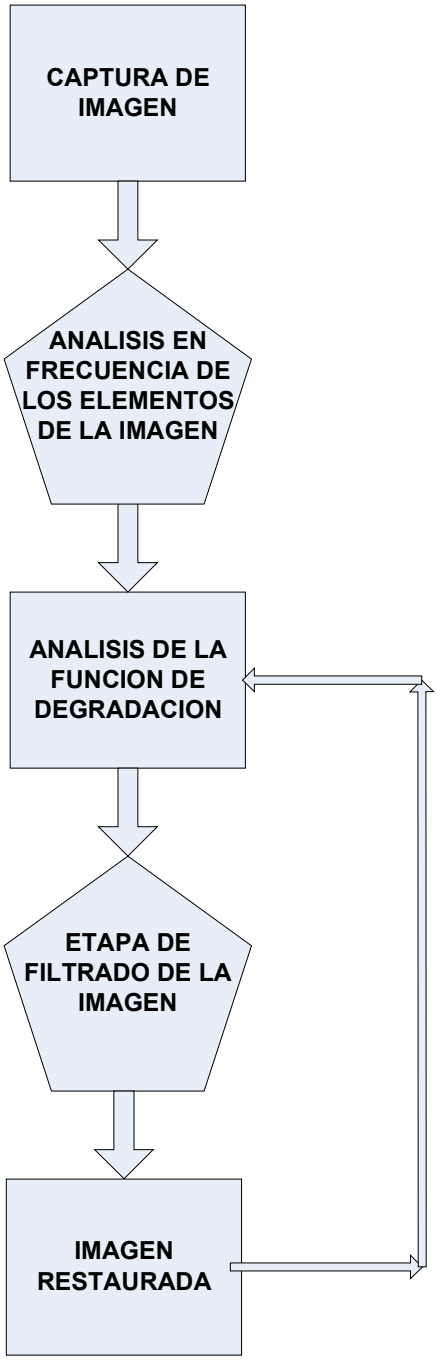
b)

c)

4.2 Diseño de un Algoritmo de programación para el Filtrado Inverso

4.2.1 Diagrama de Bloques

Figura 16 Diagrama de Bloques Filtrado Inverso



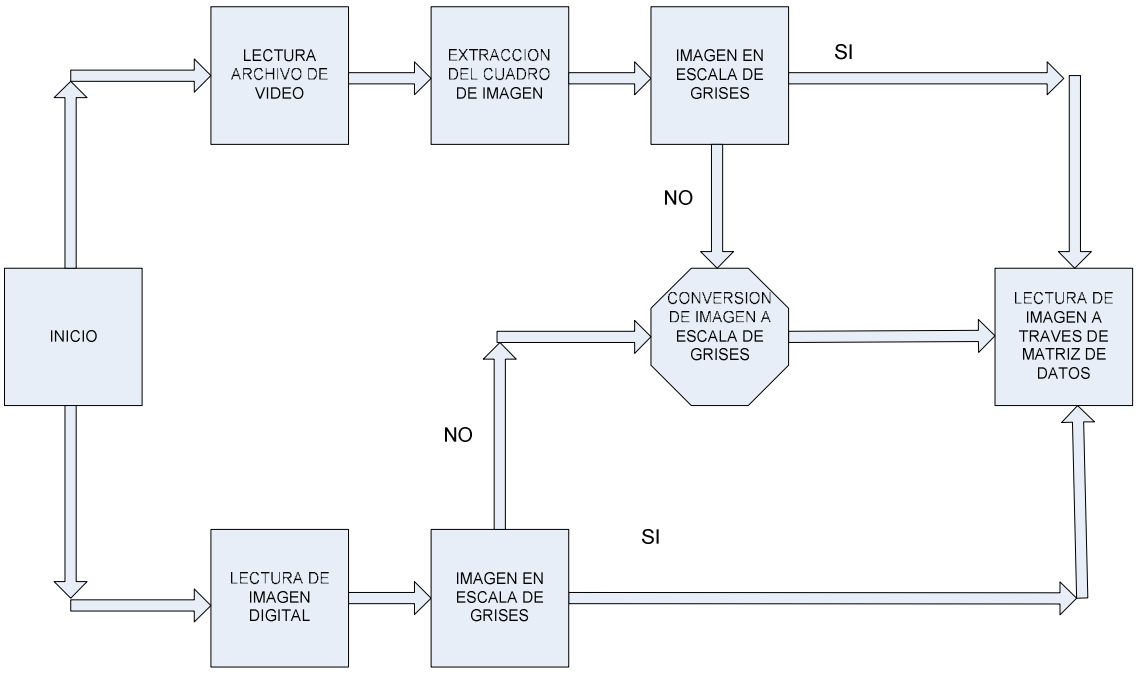
4.2.2 Etapa de Captura de la Imagen

En esta etapa inicial se definen las herramientas que serán de utilidad al momento de analizar determinada imagen, es necesario entonces explicar la manera en que *Matlab* realiza los siguientes pasos:

- Captura de una imagen digital
- Captura de una imagen digital a partir de un archivo de video
- Reconocimiento de la Imagen y Conversión de la imagen a escala de grises.
- Lectura de la imagen a partir de la matriz de datos

Lo anterior puede visualizarse a través del siguiente diagrama de bloques

Figura 17 Diagrama de bloques para la etapa de captura de la imagen



4.2.2.1 Captura y reconocimiento de un archivo de Imagen

El primer paso para lograr el análisis de una imagen a través de un algoritmo de programación es el de capturar la imagen, esta puede ser adquirida de un equipo como una cámara digital o a través de un dispositivo capaz de convertir una determinada fotografía en una imagen digital (como se ha presentado en el capítulo 1, esto es posible a través del muestreo y la cuantización) como por ejemplo de un dispositivo scanner.

Matlab es capaz de "leer" un archivo de imagen y capturar los cuadros de un archivo de video para luego convertirlos en un arreglo matricial de M filas por N columnas en el caso de una imagen en escala de grises, en donde cada elemento de la matriz corresponde a un único píxel ó en el caso de una imagen a color, un arreglo matricial de M filas por N por 3 (en donde el primer arreglo pertenece a la intensidad del color rojo en los *pixels*, el segundo para mostrar la intensidad del color azul en los *pixels* y el tercero para la intensidad del color verde que conforman la imagen).

Para poder realizar lo anterior, *Matlab* utiliza la función "Imread" para leer una imagen de un archivo grafico, y es necesario instalar el toolbox de procesamiento de imágenes de *Matlab*, que contiene las herramientas que permiten trabajar con gráficos.

La función Imread trabaja con los siguientes tipos de imágenes:

- Imágenes Binarias: Tipo de imágenes en donde el arreglo matricial se encuentra formado por unos y ceros, los cuales representan el color blanco y el negro respectivamente como los únicos colores presentes en la imagen.

- Imágenes Indexadas: Estas imágenes consisten en 2 arreglos matriciales, el primer arreglo contiene la posición de los *pixels* en la imagen y la segunda indica la intensidad de los colores rojo, verde y azul presentes en la imagen (llamada por eso “mapa de colores”), las imágenes indexadas utilizan un mapeado directo entre los valores en la matriz de *pixels* y la matriz en el mapa de colores para relacionar los datos y formar así la imagen.

- Imágenes de Intensidad: Se representan como un arreglo matricial de datos MxN en donde los valores de cada píxel especifican valores de intensidad, a este tipo de imágenes se les conoce como imágenes en “escala de grises”, al igual que en las imágenes binarias, el 0 representa el color negro y el 1 el color blanco, los valores comprendidos entre estos dos números representan las diversas intensidades de grises en la imagen. Este tipo de imágenes es el utilizado a lo largo del presente trabajo para ejemplificar el proceso de restauración de imágenes.

- Imágenes Truecolor: A diferencia de las imágenes indexadas, este tipo de imágenes no utilizan un arreglo matricial para el mapa de colores y al igual que el caso anterior, se representan como un arreglo matricial de datos de la forma MxNx3 tal y como se ha mencionado arriba, en donde los valores de cada píxel especifican valores de intensidad para los colores rojo, verde y azul que conforma la imagen. A este tipo de imágenes se les conoce como imágenes “RGB” ó Imágenes a “Color”.

La Sintaxis para la función Imread es la siguiente:

- A = imread (nombre del archivo, fmt)

En donde A representa la matriz de datos que se formará a partir de la imagen y fmt representa el formato de la imagen que se analizará.

La función Imread soporta varios tipos de formatos de imagen y sus respectivas variantes, lo que claramente representa una ventaja al momento de tener que realizar el análisis de imágenes degradadas cuando se cuenta con más de un formato de imagen disponible. A continuación se muestra la tabla con los formatos de imagen reconocidos por la función Imread.

Tabla I. Formatos de imágenes compatibles en *Matlab*

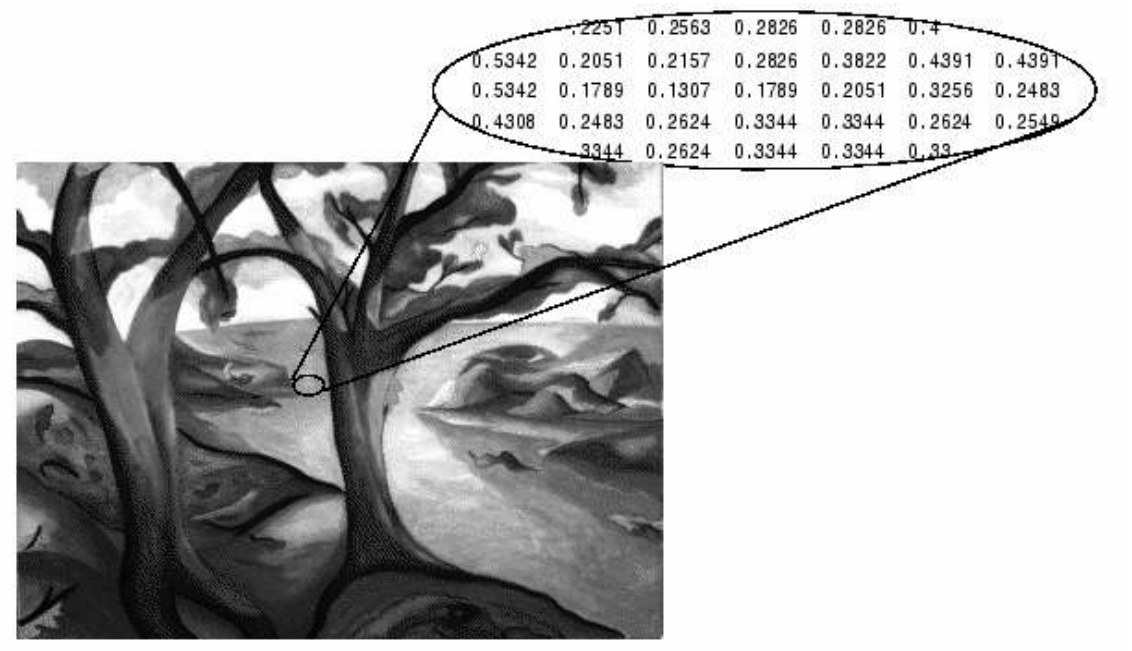
FORMATO	NOMBRE COMPLETO	VARIANTES
'bmp'	Windows Bitmap (BMP)	1-bit, 4-bits, 8-bits, 16-bits, 24-bits, y 32-bits sin compresión e imágenes de 4-bit y 8-bit de compresión RLE (Run-length encoding)
'cur'	Windows Cursor resources (CUR)	1-bit, 4-bits, y 8-bits sin compresión
'gif'	Graphics Interchange Format (GIF)	Imágenes de 1-bit a 8-bits
'hdf'	Hierarchical Data Format (HDF)	Imagen de 8-bits de datos, con o sin mapa de color asociado e imágenes de 24-bits.
'ico'	Windows Icon resources (ICO)	1-bit, 4-bits, y 8-bits sin compresión
'jpg' ó 'jpeg'	Joint Photographic Experts Group (JPEG)	Cualquier variante del formato JPEG entre los que se incluyen: imágenes en escala de grises en 8 ó 12 bits con compresión y sin compresión, e imágenes RGB en 24 ó 36 bits con compresión o sin compresión.
'pbm'	Portable Bitmap (PBM)	Imágenes de 1 bit con codificación ASCII
'pcx'	Windows Paintbrush (PCX)	Imágenes de 1-bit, 8-bits, y 24-bits
'pgm'	Portable Graymap (PGM)	Imágenes con codificación ASCII (plain) con profundidad de color arbitraria, o imágenes con codificación binaria con hasta 16 bits de valores en escala de grises
'png'	Portable Network Graphics (PNG)	Imágenes en escala de grisees de 1-bit, 2-bits, 4-bits, 8-bits, y 16-bits; imágenes indexadas 8-bits y 16-bits e imágenes RGB de 24-bits y 48-bits.
'pnm'	Portable Anymap (PNM)	PNM no es realmente un formato de imagen, sino que es una denominación que identifica e incluye cualquiera de los tres formatos “Portable Bitmap” que son los siguientes: PBM, PGM y PPM.
'ppm'	Portable Pixmap (PPM)	Imagen con codificación ASCII con profundidad de color arbitraria o Imagen con codificación binaria de 16 bits por componente de color.
'ras'	Sun Raster (RAS)	Imágenes de 1 bit bitmap, 8 bits indexada, 24 bits RGB y 32-bits RGB
'tif' ó 'tiff'	Tagged Image File Format (TIFF)	Cualquier variante del formato TIFF, entre los que se incluyen: imágenes de 1 bit, 8 bits y 24 bits con compresión y sin compresión, imágenes en escala de grises de 16 bits, imágenes de 16 bits indexadas e imágenes RGB de 48 bits.

Fuente: www.themathworks.com

Es necesario tomar en cuenta el tamaño de la imagen que se analizará, ya que mientras está sea de mayor calidad tendrá mayor cantidad de información y por ende la matriz de datos será mayor por lo que el tiempo de captura de la imagen y procesamiento se incrementará notablemente. Por su gran flexibilidad se utilizarán imágenes en formato jpg en el desarrollo de los algoritmos de aplicación en este capítulo.

La figura a continuación muestra la manera en que *Matlab* realiza el arreglo matricial de la información contenida en un sector en la siguiente imagen de intensidad.

Figura 18 Arreglo matricial para la representación de una imagen digital



Fuente: http://www.mathworks.com/access/helpdesk/help/toolbox/images/

4.2.2.2 Captura y reconocimiento de una Imagen a partir de un archivo de Video

También es posible extraer una imagen de un archivo de video, esto es posible ya que el video en si consiste en una serie de imágenes colocadas en secuencia a fin de lograr una escena en movimiento. A estas imágenes se les conoce comúnmente con el nombre de “cuadros” de imagen.

Esto resulta de gran aplicación, para una aplicación como el análisis de imágenes provenientes de una red de cámaras de video destinadas a la vigilancia y seguridad, ó para tareas tales como la edición de video por mencionar algunas aplicaciones.

Matlab es capaz de trabajar con el formato de video “AVI” cuyas siglas son diminutivo de “Audio Video Interleave”, este formato de multimedia fue introducido por Microsoft en Noviembre de 1992 como parte de su tecnología de video para Windows, desde entonces se ha convertido en un estándar y su popularidad ha ido en ascenso.

Para inicializar la función de video en *Matlab* el comando “aviread” permite leer un archivo de video en formato avi (sin compresión), esta función descompone la estructura del video en un arreglo matricial parecido al de las imágenes indexadas, donde uno de los arreglos indica la posición de los pixeles en cada cuadro y el segundo muestra la intensidad de los colores en cada cuadro. Para completar la tarea de extracción, la función “getframe” permite la captura de un cuadro de imagen de un video mediante la captura instantánea del cuadro de imagen en el tiempo deseado.

4.2.2.3 Conversión a escala de grises y lectura de la Imagen

Si la imagen se encuentra en un formato RGB, es decir, a color es posible utilizar el comando “rgb2gray” de manera que la imagen que se analizará sea en escala de grises.

Para mostrar la imagen resultante de una matriz de datos, *Matlab* utiliza la función “imshow” que realiza el mismo proceso que la función “imread” a la inversa.

4.2.3 Análisis en frecuencia de los elementos de la Imagen

Como se ha estudiado en el Capítulo 1, la transformación de una imagen al dominio de la frecuencia tiene grandes ventajas, permite reconocer las zonas de alta frecuencia como aquellos que pertenecen a contornos, bordes y detalles finos (en general, a cambios abruptos en la imagen), mientras que las zonas de baja frecuencia se relacionan con objetos grandes y sin cambios significativos dentro de la imagen.

Pero una de sus aplicaciones más importante es en el diseño de filtros digitales para la restauración de imágenes digitales, tal es el caso del Filtrado Inverso y el Filtrado de *Wiener*.

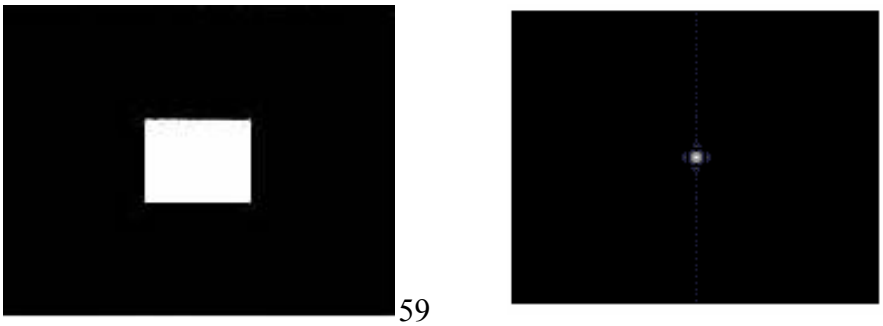
Esto se debe en gran parte al avance del procesamiento por computador y por supuesto a la versatilidad que presenta el análisis de la imagen en el dominio de la frecuencia contra los métodos de restauración basados en el análisis espacial de la imagen.

La transformada discreta de *Fourier* y su inversa se obtienen en *Matlab* utilizando un algoritmo que permite calcular la transformada rápida de *Fourier*. La transformada rápida de *Fourier* de una matriz de MxN se obtiene a través de la función “fft2”. Esta función calcula la transformada de *Fourier* de la matriz de imagen, y la función “ifft2” permite realizar la operación a la inversa. También es posible calcular el espectro de *Fourier* de la imagen a través de la función “abs” que permite calcular la magnitud de cada elemento de la matriz de datos, esto lo realiza mostrando el resultado de la raíz de la suma de los cuadrados de la parte real y parte imaginaria resultante de la transformada de *Fourier*.

Un ejemplo de la transformación al dominio de la frecuencia de una imagen sencilla puede observarse en la figura abajo mostrada, utilizando el siguiente código en *Matlab*

- I = imread (‘c:\cuadro.jpg’); lee el archivo de imagen llamado cuadro que se encuentra en el disco raíz c: y lo archiva en una matriz “I”
- J = fft2 (I); calcula la transformada discreta de *Fourier* de la matriz de datos I
- K = fftshift(J); sitúa los valores de frecuencia cero en medio de la matriz de datos, permitiendo visualizar de mejor manera los componentes de espectro de *Fourier*
- L = abs (K); toma la parte real de los datos de la matriz
- Imshow (K, []); muestra el espectro de *Fourier* de la imagen

Figura 19 Imagen y su espectro calculado en *Matlab*



Existen también otros comandos tales como “size” y “whos” los que permiten conocer datos importantes de la imagen tales como el número de filas y columnas de la matriz de imagen para el primer caso y la información del arreglo matricial de la imagen en el segundo caso.

4.2.4 Análisis de la Función de Degradación

Esta es una de las etapas de mayor importancia para lograr una buena aplicación del Filtro Inverso, ya que aparte de que se debe conocer con cierta precisión el fenómeno que causa la distorsión o efecto borroso en la imagen, existe el inconveniente de que si la función que modela este fenómeno contenga tener valores cercanos a cero, no será posible realizar efectuar el filtrado inverso. Por lo tanto el algoritmo debe ser capaz de analizar y definir un valor “n” para los puntos donde H(u,v) no exista. Esto dará lugar, como se ha estudiado en el capítulo 3 a la definición del Filtro Pseudoinverso donde:

$$H'(u,v) = \begin{cases} \dfrac{1}{H(u,v)} & H(u,v) \neq 0 \\ n & H(u,v) = 0 \end{cases} \qquad (4.0)$$

El problema que se debe resolver es que valor n se utilizará, si se escoge un valor muy elevado y no hay presencia de ruido en la imagen, es posible que esta pierda información importante en caso de que la matriz de datos a evaluar contenga gran cantidad de ceros.

Por lo general en caso de no existir ruido el valor de n se ajusta lo más cercano a cero, permitiendo así la aplicación correcta del filtro pseudoinverso, sin permitir mayores distorsiones en la imagen.

Por el contrario si la imagen contiene ruido se suele escoger un valor n cada vez más grande, esto se debe a que el Filtro Inverso (por ser este una forma de Filtro Pasa Altos) tiende a amplificar el ruido que se presenta en alta frecuencia, y un valor alto de “n” permitirá atenuar en cierta medida el ruido en la imagen. Esto aparte de no resultar nada práctico, genera también perdida de datos en la imagen, por lo que no es un método de fiar en caso de que la imagen se encuentre contaminada con ruido.

4.2.5 Etapa de Filtrado de la Imagen Degradada

En esta etapa se centra en el desarrollo del algoritmo capaz de realizar la restauración de la imagen a través del Filtrado Inverso, haciendo uso de las herramientas matemáticas que provee *Matlab* de manera flexible tal que permita variaciones en su código en caso sea necesario para lograr la optimización de la imagen resultante.

4.3 Pruebas del Método de Filtrado Inverso y resultados

El desarrollo de un algoritmo para el proceso de restauración de imágenes implica en si el hecho de que debe realizarse una serie de pruebas para verificar el correcto funcionamiento del algoritmo y a su vez verificar el desempeño del Metodo de Filtrado Inverso.

Para realizar lo anterior se ha creado una GUI en *Matlab*, con el nombre de “FILTRADOINV” (cuyo código se explicará en la siguiente sección), su objetivo es el de capturar una imagen y degradarla utilizando un barrido de pixeles en la imagen para simular los efectos borrosos que se presentan en las imágenes (por las causas estudiadas en el capítulo 2), para luego restaurarla utilizando previo conocimiento de lo que ha causado la degradación.

La GUI se encuentra formada por los siguientes elementos:

Tres botones de aplicación

- Lectura de Imagen: Permite la captura de una imagen digital y guarda la matriz de datos en una variable interna del programa.
- Degradación: Agrega un barrido de *pixels* para simular el movimiento borroso en una imagen.
- Filtro Inverso: En esta parte se desarrolla el algoritmo que permitirá restaurar la imagen a su estado original.

Cinco editores para el ingreso de variables

- Lon: En esta variable se guarda el desplazamiento que se le dará a los pixeles en la función de degradación.
- Ang: en esta variable se guarda el angulo de barrido que se le dará a los pixeles en la función de degradación.

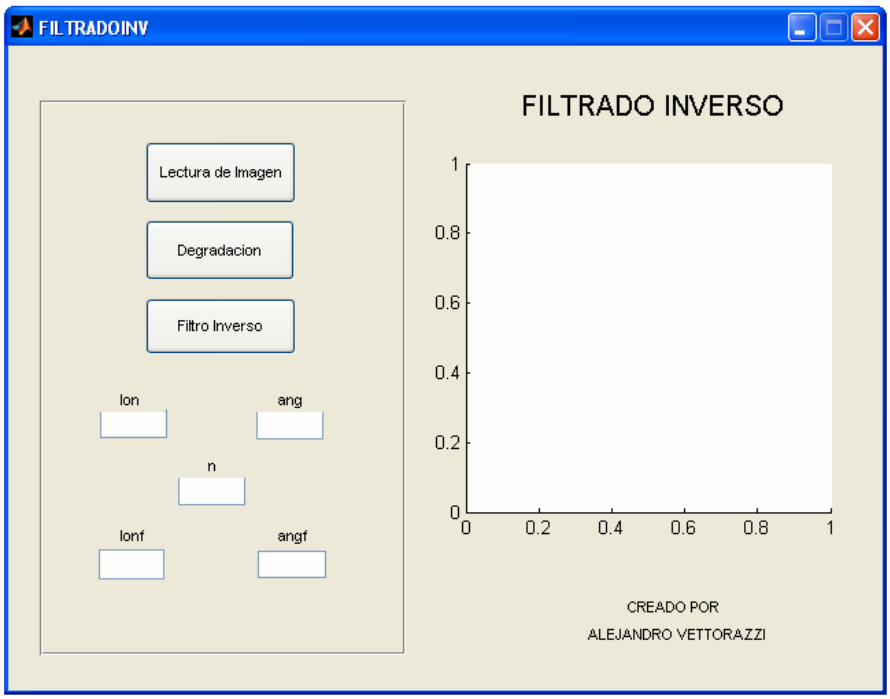
- n: En esta variable se conserva el valor con el cual se definirá H(u,v) en los puntos donde este es cero.
- Lonf: En esta variable se guarda el desplazamiento que se le dará a los pixeles en la función de degradación que se utilizará para Filtrar la imagen
- Angf: En esta variable se guarda el ángulo de barrido que se le dará a los pixeles en la función de degradación que se utilizará para Filtrar la imagen.

Un Cuadro de Ejes

- Mostrará la imagen degradada luego del proceso de distorsión, permitirá también observar la imagen restaurada luego del filtrado.

A continuación se muestra el diseño de la GUI propuesta:

Figura 20 Interfaz GUI realizada en *Matlab* para la aplicación del Filtro inverso



A continuación, se degradará una determinada imagen en escala de grises utilizando un desplazamiento equivalente a 150 pixeles a un ángulo de 65 grados. La imagen resultante puede observarse luego de efectuarse el proceso de degradación, es fácil darse cuenta que no es posible reconocer la información contenida en la imagen.

Figura 21 Degradación de una imagen utilizando los valores mostrados

Si se utilizan las mismas variables en la función de degradación que filtrará la imagen, esta quedara restaurada y solo mostrará pequeños errores a veces causados por los cálculos que realiza el ordenador al pasar del dominio de la frecuencia al dominio del espacio (y viceversa) por causa de las operaciones punto flotante.

La imagen restaurada se puede visualizar en la siguiente figura

Figura 22 Imagen Restaurada luego de la etapa de filtrado

Es de gran importancia conocer los efectos de la degradación en una imagen, mientras mas exacto sea este conocimiento, el resultado final será mejor. Abajo puede observarse un nuevo ejemplo, en donde la imagen se restaurará utilizando una función de degradación que tiene una variación de 0.1 en la variable “lonf” respecto a la variable “lon” de la función de degradación original.

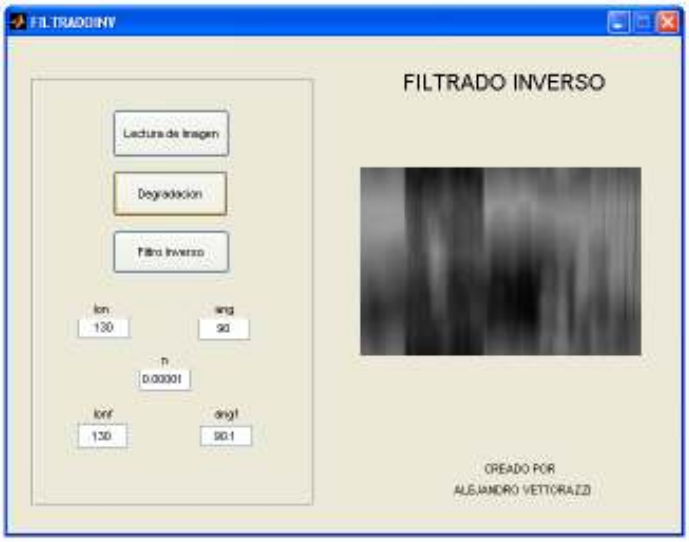
Figura 23 Imagen distorsionada utilizando los valores mostrados

Los efectos que esto tiene sobre la imagen final restaurada pueden apreciarse en la figura 21.

Figura 24 Imagen restaurada

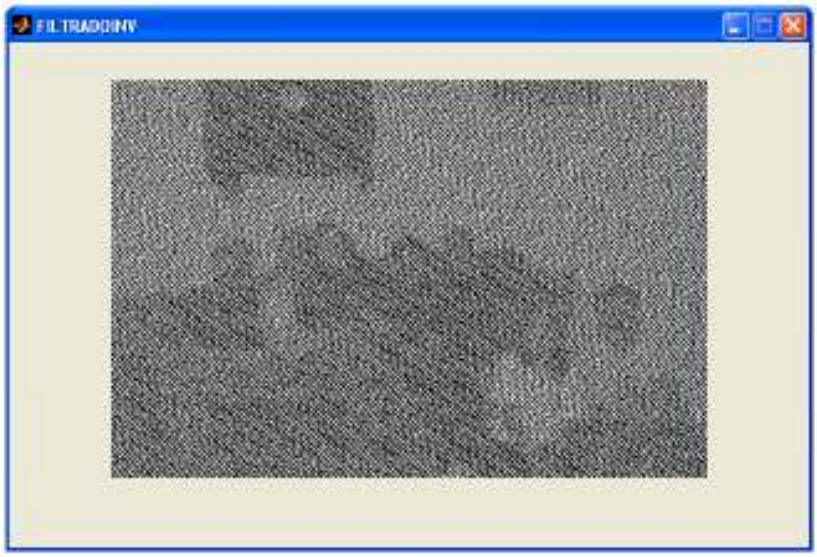
Si en cambio variamos el ángulo de barrido de los pixeles en vez del desplazamiento se tendrá lo siguiente:

Figura 25 Imagen distorsionada utilizando los valores mostrados



Y el resultado de la imagen restaurada será la que se muestra en la siguiente figura.

Figura 26 Imagen Restaurada



En ambos casos fue apreciable el hecho de que no fue posible restaurar de manera adecuada la imagen original, por lo tanto es indispensable conocer las características de la función de degradación H(u,v) que afecta la imagen si se quiere recuperar la información contenida en la imagen.

4.4 Código fuente de la GUI para la aplicación del Filtro Inverso

A continuación se expone el código que forma la aplicación GUI para el Método de Filtrado Inverso.

Instrucciones para la inicialización de la GUI:

```
---------------
function varargout = FILTRADOINV(varargin)

gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @FILTRADOINV_OpeningFcn, ...
                   'gui_OutputFcn',  @FILTRADOINV_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end
 if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT

% --- Executes just before FILTRADOINV is made visible.
function FILTRADOINV_OpeningFcn(hObject, eventdata, handles, varargin)
handles.output = hObject;
guidata(hObject, handles);
---------------
```

El anterior código es generado automáticamente por *Matlab* al momento de definir las variables y componentes del GUI.

```
---------------
function varargout = FILTRADOINV_OutputFcn(hObject, eventdata, handles)
varargout{1} = handles.output;
function pushbutton2_Callback(hObject, eventdata, handles)
global i;
i=imread('c:\foto.jpg');
---------------
```

Se define la variable “i” y se guarda en esta la matriz de datos de la imagen digital.

En caso de que la imagen a examinar sea de tipo RGB se debe incluir la siguiente línea de código:

```
c=rgb2gray(i);
```

De esta manera, la imagen será transformada a escala de grises. En este nuevo caso, la matriz de datos de la imagen digital se guardará en la variable c.

```
---------------
function pushbutton3_Callback(hObject, eventdata, handles)
global i;
global j;
global k;
global d;
global e;
global lon;
global ang;
```

```
global psf;
global otf;
j=im2double(i);
k=fft2(j);
psf=fspecial('motion',lon,ang);
otf = psf2otf(psf,size(i));
d=otf.*k;
e=abs(ifft2(d));
for v=1:size(e,1)
   for w=1:size(e,2);
      if e(v,w)>=1
         e(v,w)=0.999;
      end
      if e(v,w)<=0
         e(v,w)=0.001;
      end
   end
end

imwrite (e, 'c:\2.jpg');
imshow (e)
----------
```

En esta sección se realiza la degradación de la imagen a través de la función de degradación propuesta. Esta función de degradación se transforma en una función de transferencia óptica (a través de la transformada de *Fourier*) para lograra en el dominio de la frecuencia la convolución de la imagen original con la función degradada.

Al aplicar la transformada inversa al resultado de la imagen degradada, con el fin de retornar al dominio del espacio, se debe de limitar los valores dentro de la matriz en 1 y 0 (es necesario recordar que las imágenes de intensidad en escala de grises tienen valores maximos de 0 y 1 donde el color negro y el blanco son representados respectivamente) ya que debido a las operaciones de punto flotante en el ordenador, es posible que ocurran fallos que aunque no sean relevantes matemáticamente, causan que la matriz de datos contenga valores por encima de 1 y debajo de 0.

```
function pushbutton4_Callback(hObject, eventdata, handles)
global e;
global f;
global lon;
global d;
global ang;
global n;
global otf;
global lonf;
global angf;
psf2=fspecial('motion',lonf,angf);
otf2=psf2otf(psf2,size(e));
for l = 1:size(otf2,1)
   for m= 1:size(otf2,2);
      if otf2 (l,m)==0;
         otf2(l,m)=n;
      end
   end
end
p=fft2(e);
R=p./otf2;
Re=ifft2(R);
```

```
for a = 1:size(Re,1)
   for b = 1:size(Re,2)

      if Re(a,b)>=1
         Re(a,b)=0.999;
      end
      if Re(a,b)<=0
         Re(a,b)=0.001;
      end
   end
end
imshow(Re)
---------------
```

En esta etapa se realiza el proceso de Filtrado Inverso, utilizando las variables lonf y angf para formar la función de degradación que se utilizará para el filtrado.

```
 ---------------
function edit1_Callback(hObject, eventdata, handles)
global n;
n=str2double(get(hObject,'String'));
---------------
```

Se guarda el valor de entrada numérico en la variable llamada “n”.

```
---------------
function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
   set(hObject,'BackgroundColor','white');
end
---------------
```

Código generado por *Matlab* como parte del diseño de la Interfaz Grafica.

```
---------------
function edit2_Callback(hObject, eventdata, handles)
global lon;
lon= str2double(get(hObject,'String'));
---------------
```

Guarda el valor de entrada numérico en la variable llamada “lon”.

```
---------------
function edit2_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
---------------
```

Código generado por *Matlab* como parte del diseño de la Interfaz Grafica.

```
---------------
function edit3_Callback(hObject, eventdata, handles)
global ang;
ang=str2double(get(hObject,'String'));
---------------
```

Guarda el valor de entrada numérico en la variable llamada “ang”.

```
---------------
function edit3_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
```

```
function axes1_CreateFcn(hObject, eventdata, handles)
function edit4_Callback(hObject, eventdata, handles)
global lonf;
lonf=str2double(get(hObject,'String'));
---------------
```

Definición de objetos para la Interfaz Grafica, Eje para gráficos de datos e ingreso del valor de entrada numérico en la variable llamada “lonf”.

```
---------------
function edit4_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
function edit5_Callback(hObject, eventdata, handles)
global angf;
angf=str2double(get(hObject,'String'));
 ---------------
```

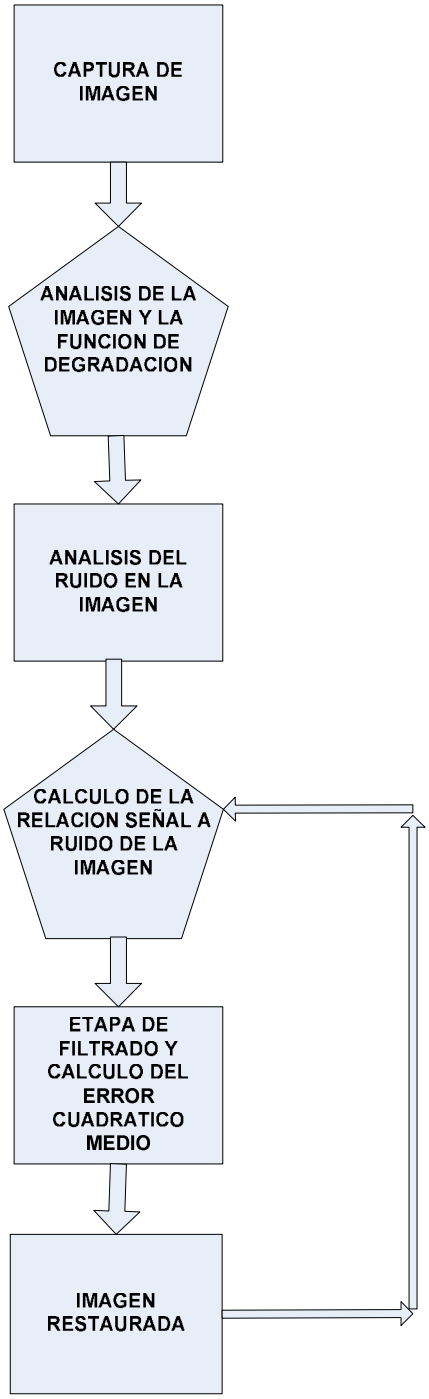
Definición de objetos para la Interfaz Grafica, Eje para gráficos de datos e ingreso del valor de entrada numérico en la variable llamada “angf”.

.

4.5 Diseño de un algoritmo de programación para la aplicación del Filtrado de *Wiener*

4.5.1 Diagrama de Bloques

Figura 27 Diagrama de Bloques Filtrado de *Wiener*



El proceso de Captura de Imagen y Análisis de la Imagen capturada ha sido presentado en el apartado sobre el Filtrado Inverso, su aplicación y algoritmo es igual para el Filtrado de *Wiener* por lo que se proseguirá con el análisis de los fenómenos de distorsión que afectan las imágenes que pueden restaurarse utilizando este método.

4.5.2 Análisis de la Función de Degradación y Ruido en la Imagen

Basado en el modelo de Degradación/Restauración, el éxito fundamental de la restauración de imágenes consiste en la deconvolución de la imagen degradada utilizando la función de degradación que describe de alguna forma el fenómeno de distorsión en la imagen. La calidad de la imagen restaurada esta determinada principalmente por el conocimiento que se tenga de la función de degradación.

A su vez, es necesario conocer (en caso se encuentre presente) las características principales del ruido aditivo en la imagen, este puede modelarse gracias a las propiedades estadísticas que presenta el ruido y por lo tanto se puede determinar así la función de densidad de probabilidad que representa esta distorsión tal y como se ha estudiado en el capítulo 2 del presente trabajo.

4.5.3 Cálculo de la relación Señal a Ruido en la Imagen

Para poder estimar la relación señal a ruido en la imagen es necesario conocer el espectro de potencia de la imagen original, como esto no es posible en un caso real, se asume que las funciones son constantes y que por lo tanto se puede utilizar un valor “K” que a por medio de diferentes iteraciones permitirá dar una mejor aproximación de la señal original.

4.5.4 Etapa de Filtrado y calculo del Error Cuadrático Medio

Al igual que en el Filtrado Inverso, en esta sección el algoritmo de programación en *Matlab* realiza las operaciones matemáticas necesarias para restaurar la imagen corrompida, de manera que este permita realizar el análisis del arreglo matricial resultante.

El éxito del resultado final dependerá de la optimización del error cuadrático medio, si se conoce la imagen original se puede calcular esta relación en *Matlab* a través de la función “mean”, mientras menor sea este error, más fiel será la imagen restaurada con respecto a la imagen original. Por el contrario si el error resulta muy grande será evidente a simple vista el fallo en el proceso de restauración, y se deberá recurrir a realizar de nuevo el procedimiento, utilizando un nuevo valor “K” para la relación señal a ruido.

En la figura abajo mostrada se compara una imagen contaminada con ruido gaussiano con su original, estas se han llamado foto y fotogauss respectivamente, para realizar el cálculo del error cuadrático medio es posible utilizar el siguiente comando

```
Error= mean ((I(:)-J(:)).^2);
```

En donde

I = arreglo matricial de la imagen original

J = arreglo matricial de la imagen restaurada

4.6 Pruebas del Método de Filtrado de *Wiener* y resultados

Al igual que en el Filtrado Inverso, el algoritmo de programación propuesto para la implementación del Filtrado de *Wiener* permitirá verificar el comportamiento del Filtro en situaciones tales como el ruido y el efecto borroso en una imagen.

La GUI propuesta lleva el nombre de Filtrado_*Wiener* y su codigo se explicará en la siguiente sección. El objetivo de esta es el de capturar una imagen y simular el proceso de contaminación a través de ruido aleatorio y borrosidad, para luego utilizar los conceptos estudiados a fin de restaurar la imagen.

La GUI se encuentra formada por los siguientes elementos:

Cuatro botones de aplicación:

- Lectura de Imagen: Permite la captura de una imagen digital y guarda la matriz de datos en una variable interna del programa.
- Degradación sin Ruido: Agrega un barrido de *pixels* para simular el movimiento borroso en una imagen.
- Degradación con Ruido: Además del barrido de *pixels*, agrega ruido aleatorio a la imagen degradada.
- Filtrado de *Wiener*: En esta sección se desarrolla el algoritmo que permitirá restaurar la imagen a su estado original.

Tres editores para el ingreso de variables:

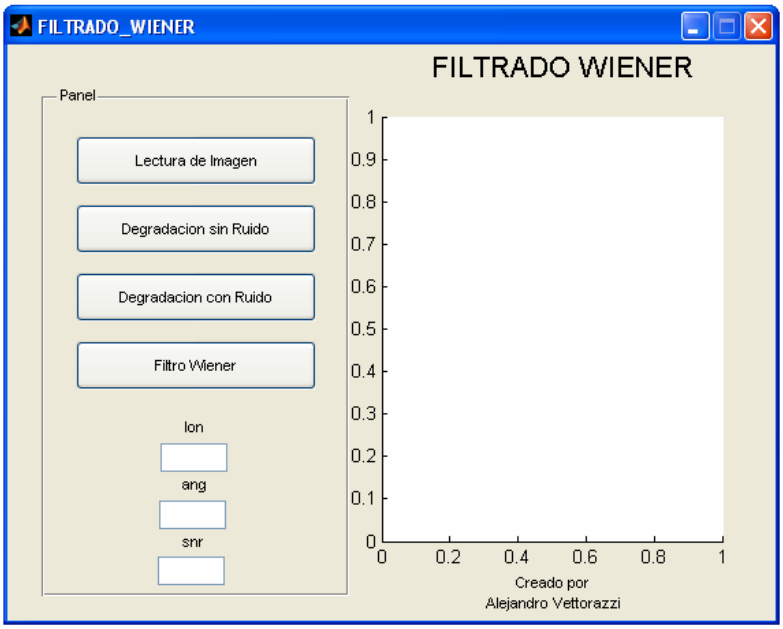
- Lon: En esta variable se guarda el valor del desplazamiento que se le dará a los pixeles en la función de degradación.
- Ang: En esta variable se guarda el ángulo de barrido que se le dará a los pixeles en la función de degradación.
- Snr: En esta variable se guarda el valor de la relación señal a ruido, con el fin de observar el cambio que produce en la imagen la variación de esta.

Un Cuadro de Ejes

- Mostrará la imagen degradada luego del proceso de distorsión, permitirá también observar la imagen restaurada luego del filtrado.

A continuación se muestra el diseño de la GUI propuesta:

Figura 28 Interfaz GUI realizada en *Matlab* para la aplicación del Filtro de *Wiener*



A continuación, se degradará sin ruido una determinada imagen en escala de grises utilizando un desplazamiento de *pixels* equivalente a 25 *pixels* a un ángulo de 35 grados. La imagen resultante puede observarse luego de efectuarse el proceso de degradación, la información en la imagen es ilegible luego de la distorsión.

En ausencia de ruido en la imagen, la relación señal a ruido es igual a cero, por lo que para lograr la restauración de la imagen se debe ingresar este valor en la casilla “snr”.

Figura 29 Imagen distorsionada utilizando los valores mostrados

Es posible observar luego de la restauración, que el Filtro de *Wiener* se comporta de manera similar al Filtrado Inverso en cuanto a la restauración de imágenes afectadas por este fenómeno de distorsión, y se producirá el mismo efecto presentado en las figuras 21 y 23 si la variación de los parámetros en la función de degradación son diferentes aunque sea en un pequeño porcentaje.

Figura 30 Imagen restaurada luego de la etapa de filtrado

El Filtro de *Wiener* permite también contar con otro parámetro de referencia como lo es el Error Cuadrático Medio, por lo que si se tiene la posibilidad de contar con la imagen original y la imagen restaurada es posible observar entonces cuanto difieren una de la otra.

En *Matlab*, el Error cuadrático medio Puede calcularse mediante la siguiente expresión:

```
error=mean((imagenoriginal(:)-imagenrestaurada(:)).^2)
```

Aunque el parámetro más importante en este tipo de imágenes sea el visual, el Error cuadrático medio es una buena referencia al momento de escoger los valores de la relación señal a ruido necesarios para eliminar el ruido en una determinada imagen. Como se conoce con exactitud la función de degradación que distorsiona la imagen es lógico ver que el error cuadrático medio entre la imagen original y la imagen restaurada es casi cero.

4.6.1 Imágenes distorsionadas con ruido

Cuando la imagen presenta además de distorsión lineal, ruido, es cuando realmente puede apreciarse el efecto del Filtro de *Wiener*, si se conocen las características del ruido que contamina la imagen es posible realizar un estimado de la imagen original que será bastante aceptable.

Para adicionar ruido en la imagen se ha utilizado el código siguiente en *Matlab*:

```
ruido = n*randn(size(g));
g=imadd(g,ruido);
```

En donde “g” representa la variable que contiene la imagen que ha sido distorsionada. El termino “n*randn(size(g))” crea una matriz de datos aleatorio del mismo tamaño que la imagen degradada donde “n” será el factor de multiplicación de los valores en la matriz. El comando “imadd(g,ruido)” combinará los elementos de ambas matrices de datos, dando como resultado ruido en la imagen.

Para ejemplificar lo anterior se repetirá el último ejemplo, utilizando los mismos valores para las variables “lon” y “ang” dando como resultado la siguiente imagen:

Figura 31 Imagen distorsionada utilizando los valores mostrados

Como se menciono antes, generalmente no se cuenta con la imagen original para calcular directamente la relación señal a ruido en la imagen, por lo que se escoge una constante cualquiera (en este caso esta constante se llamara “snr) y a través de una serie de iteraciones se espera llegar al resultado correcto. Si se escoge 0.5 como valor para “snr” el resultado será el siguiente:

Figura 32 Imagen restaurada luego de la primera iteración

Aunque el resultado no es tan claro es posible observar que los efectos debido a la borrosidad se han reducido en gran parte, al igual que los efectos del ruido. Es posible observar también que el Filtrado de *Wiener* responde de mejor manera que el Filtrado Inverso en situaciones donde la imagen presenta ruido, además la imagen anterior proporciona una primera aproximación de la forma en que la imagen original debe verse, por lo tanto es necesario escoger otro valor como constante y realizar el proceso de nuevo.

Si en vez de tomar como constante el valor de 0.5 se toma el valor de 0.3, el resultado será el siguiente:

Figura 33 Imagen restaurada luego de la segunda iteración

En la segunda iteración, es posible observar que esta vez el filtrado ha logrado remover más problemas en la imagen permitiendo extraer cada vez más información de esta.

Es posible seguir iterando hasta llegar a un resultado muy exacto, pero si se tiene la oportunidad de contar con una imagen de referencia (en este caso esta imagen sería la original), es posible obtener la verdadera relación señal a ruido e ingresarla a la ecuación del filtro inverso, esto permitiría una restauración casi completa de la imagen degradada.

4.7 Código fuente de la GUI para la aplicación del Filtrado de *Wiener*

A continuación se expone el código que forma la aplicación GUI para el Método de Filtrado Inverso

Instrucciones para la inicialización de la GUI

```
---------------
function varargout = FILTRADO_WIENER(varargin)
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                   'gui_Singleton',  gui_Singleton, ...
                   'gui_OpeningFcn', @FILTRADO_WIENER_OpeningFcn, ...
                   'gui_OutputFcn',  @FILTRADO_WIENER_OutputFcn, ...
                   'gui_LayoutFcn',  [] , ...
                   'gui_Callback',   []);
if nargin && ischar(varargin{1})
   gui_State.gui_Callback = str2func(varargin{1});
end

if nargout
   [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
   gui_mainfcn(gui_State, varargin{:});
end


% --- Executes just before FILTRADO_WIENER is made visible.
function FILTRADO_WIENER_OpeningFcn(hObject, eventdata, handles,
varargin)
% Choose default command line output for FILTRADO_WIENER
handles.output = hObject;
guidata(hObject, handles);
```

```
function varargout = FILTRADO_WIENER_OutputFcn(hObject, eventdata,
handles)
varargout{1} = handles.output;
---------------
```

El código anterior es generado automáticamente por *Matlab* al momento de definir las variables y componentes de la GUI

```
---------------
function pushbutton1_Callback(hObject, eventdata, handles)
global i;
global im;
i=imread('c:\familiag.jpg');
im=im2double(i);
---------------
```

Se define la variable “I” y se guarda en esta, la matriz de datos de la imagen digital.

En caso que la imagen a examinar sea de tipo RGB, se debe incluir las instrucciones para transformar esta en una imagen de intensidad de escalad e grises como se ha visto en el código del Filtro Inverso.

Se define además la variable “im” que convierte la imagen en una imagen de doble precisión.

```
 ---------------
function pushbutton2_Callback(hObject, eventdata, handles)
global im;
global lon;
global ang;
global g;
imf=fft2(im);
psf=fspecial('motion',lon,ang);
otf=psf2otf(psf,size(im));
h=otf.*imf;
g=abs(ifft2(h));
for l=1:size(g,1)
   for m=1:size(g,2)
      if g(l,m)>=1
         g(l,m)=0.9999;
      end
      if g(l,m)<=0
```

```
        g(l,m)=0.0001;
      end
   end
end
imwrite(g,'c:\degradación.jpg');
imshow (g)
---------------
```

En esta sección se realiza la degradación de la imagen a través de la función de degradación propuesta y es el mismo código empleado para degradar las imágenes en la aplicación creada para el Filtrado Inverso.

```
---------------
function pushbutton3_Callback(hObject, eventdata, handles)
global im;
global lon;
global ang;
global media;
global var;
global snr;
global g;
global n;
global ruido;
imf=fft2(im);
psf=fspecial('motion',lon,ang);
otf=psf2otf(psf,size(im));
h=otf.*imf;
g=abs(ifft2(h));
for l=1:size(g,1)
   for m=1:size(g,2)
      if g(l,m)>=1
         g(l,m)=0.9999;
      end
      if g(l,m)<=0
         g(l,m)=0.0001;
      end
   end
end
ruido = 0.1*randn(size(g));
g=imadd(g,ruido);
imwrite(g,'c:\degradación.jpg');
size (g)
imshow (g)
---------------
```

En esta parte se especifica la distorsión utilizando ruido como complemento, y utiliza el mismo código que en las anteriores aplicaciones con la diferencia de integrar una matriz de datos aleatorios que introducirá ruido a la imagen, como es ha explicado en la sección 4.6.1

```
---------------
function edit1_Callback(hObject, eventdata, handles)
global lon;
lon=str2double(get(hObject,'String'));
% Hints: get(hObject,'String') returns contents of edit1 as text
%        returns contents of edit1 as a double
 ---------------
```

Guarda el valor de entrada numérico en la variable llamada “lon”

```
---------------
function edit1_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
---------------
```

Código generado por *Matlab* como parte del diseño de la interfaz Grafica.

```
---------------
function edit2_Callback(hObject, eventdata, handles)
global ang;
ang=str2double(get(hObject,'String'));
---------------
```

Guarda el valor de entrada numérico en la variable llamada “ang”

```
---------------
function edit3_Callback(hObject, eventdata, handles)
global snr;
snr=str2double(get(hObject,'String'));
---------------
```

Guarda el valor de entrada númerico en la variable llamada “snr”.

```
---------------
function pushbutton4_Callback(hObject, eventdata, handles)
global g;
global lon;
global ang;
global media;
global var;
global im;
global ruido;
global snr;
ge = fft2(g);
psf=fspecial('motion',lon,ang);
otf=psf2otf(psf,size(ge));

otfc=conj(otf);
modotf=otf.*otfc;
wien=((modotf./(modotf+snr))./(otf)).*ge;
wiener=ifft2(wien);
imwrite(wiener,'c:\wiener.jpg');
imshow (wiener)
mse=mean((im(:)-wiener(:)).^2)
---------------
```

En esta etapa se realiza el proceso de Filtrado de *Wiener*, se realizan los procedimientos matemáticos de manera que la ecuación del filtrado de *Wiener* sea aplicable.

Al final se extrae el valor del error cuadrático medio para comparar la imagen original con la imagen degradada.

```
 ---------------
function axes1_CreateFcn(hObject, eventdata, handles)

function edit4_CreateFcn(hObject, eventdata, handles)
if ispc && isequal(get(hObject,'BackgroundColor'),
get(0,'defaultUicontrolBackgroundColor'))
    set(hObject,'BackgroundColor','white');
end
  ---------------
```

Fin del Programa

CONCLUSIONES

1. El Método de Filtrado Inverso puede aplicarse con éxito en imágenes distorsionadas, siempre y cuando se cumpla lo siguiente:

    - Se conozca con cierta precisión la función de degradación de la imagen h(x,y) que causa la distorsión.

    - El ruido en la imagen es nulo o su presencia es escasa en la imagen.

2. El Método de Filtrado Pseudo Inverso es una variación del Método de Filtrado Inverso, que restringe los valores en la función de degradación, de tal manera que éste pueda ser aplicado en un algoritmo de programación.

3. El Método de Filtrado de *Wiener* puede aplicarse con éxito en imágenes distorsionadas, siempre y cuando se cumpla lo siguiente:

    - Se conozca con cierta precisión la función de degradación de la imagen, h(x,y) que causa la distorsión.

- Se conozcan las propiedades del ruido que contamina la imagen analizada.

4. *Matlab* en conjunto con el *Image Processing Toolbox* es capaz de realizar las siguientes operaciones dentro del procesamiento digital de imágenes:

   - Realizar la captura de una imagen digital y guardar la información en una variable en memoria, así como también convertir esta imagen entre las diferentes clases de imágenes soportadas por *Matlab*, es decir: RGB, Escala de Grises, Binarias e Indexadas.

   - Realizar las operaciones matemáticas necesarias para trasladar la información de la imagen, del dominio espacial al dominio de la frecuencia y viceversa, a manera de llevar a cabo el proceso de filtrado.

   - Simular procesos de distorsión en las imágenes de acuerdo a modelos propuestos, los cuales representan los casos de distorsión que se presentan al momento de capturar una imagen.

   - Almacenar las imágenes procesadas de acuerdo a las diferentes clases de formatos soportados por *Matlab*.

5. Los algoritmos desarrollados en el presente trabajo permiten restaurar una imagen digital degradada, de acuerdo a los conceptos definidos para el Método de Filtrado Inverso y Método de Filtrado de *Wiener*.

RECOMENDACIONES

1. El conocimiento correcto de los fenómenos que causan la distorsión en determinadas imágenes, puede utilizarse para implementar algoritmos eficientes de programación enfocada a solucionar este problema en particular.

2. Utilizar paquetes de cómputo tales como *Matlab* facilitan el desarrollo de programas enfocados al procesamiento digital de imágenes, gracias a sus funciones matemáticas integradas.

3. Implementar algoritmos de programación para actividades repetitivas (tales como el proceso de traslación al dominio de la frecuencia y al dominio del espacio) a fin de ahorrar tiempo y facilitar su aplicación en soluciones diferentes.

BIBLIOGRAFÍA

1. González, Rafael C. Richard E. Woods. **Digital Image Processing**. Pearson Education, USA, 2002.

2. Pratt , William. **Digital Image Processing**. John Wiley, USA, 2001.

3. Petrou, Maria. **Image Processing, The Fundamentals.** John Wiley, Inglaterra, 1999.

4. Acharya, Tinky. **Image Processing.** John Wiley, USA, 2005

5. Russ, John. **The Image Processing Handbook.** CRC Press, USA, 3era Edición, 1999.

6. http://www.ph.tn.tudelft.nl/Courses/FIP/frames/fip.html. **The Image Processing Fundamentals.**

7. The Mathworks, Inc. Homepage. www.mathworks.com. Febrero 2007.

8. Walpole, Ronald. **Probabilidad y estadística para ingenieros.** McGraw-Hill, USA, 1998.

9. http://bochica.udea.edu.co/~bcalderon/3_propiedadesestimadores.html. Bernardo Calderón. Colombia.