Universidad de las Ciencias Informáticas

Facultad 10

Implementación de algoritmos de reducción de ruido en las imágenes

TRABAJO DE DIPLOMA PARA OPTAR POR EL TÍTULO
DE
INGENIERO EN CIENCIAS INFORMÁTICAS

Autores:

Raciel Cepero Ruz

Claudia Cabrera Rodríguez

Tutor:

Ing. Joel Armada Herrera

Ciudad de la Habana, Mayo 2010

AGRADECIMIENTOS

A nuestros padres y familiares por su apoyo incondicional.

A Gilberto por siempre estar ahí.

A nuestro tutor Joel.

A todas aquellas personas que hicieron posible la realización de este trabajo, que nos apoyaron, guiaron y orientaron.

Claudia y Raciel

DECLARACIÓN DE AUTORÍA

Declaramos ser autores del presente trabajo de diploma y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales del mismo, con carácter exclusivo.

Para que así conste firmamos la presente a los ____ días del mes de mayo del año 2010

Raciel Cepero Ruz

_____

Claudia Cabrera Rodríguez

_____

Joel Armada Herrera

_____

OPINIÓN DEL TUTOR

RESUMEN

En un sistema de prensa resulta de vital importancia contar con imágenes de buena calidad a la hora de publicar materiales noticiosos, basándose esto en el hecho de que con una imagen se puede transmitir más información que con cualquier palabra. Sin embargo puede suceder que las imágenes necesarias para publicar un determinado material no tengan la calidad requerida, pues podrían presentar ruidos provocados por interferencias que pudieron surgir a la hora de tomar la foto o de textos que en un momento determinado fueron superpuestos a la misma. En la actualidad existen numerosos algoritmos dedicados a lograr la reducción de ruido en las imágenes, con los cuales se obtienen muy buenos resultados, por lo que la realización de este trabajo va dirigida a la implementación de los mismos teniendo en cuenta las características y particularidades de los medios de prensa cubanos.

ÍNDICE

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

Desde el surgimiento de la imprenta, la prensa escrita ha materializado el relato de los hechos y las opiniones a través de la escritura; teniendo la característica de perdurar en el tiempo, ya que no es volátil, queda asentada y su puede recurrir a esta en cualquier momento. Esto trajo consigo que desde sus inicios ganara gran popularidad entre las personas, convirtiéndolo en un medio imprescindible de información. Pero debido a que una noticia podía ser interpretada desde diferentes puntos de vista, fue necesario incorporar un elemento muy importante con el fin de darle más veracidad a los artículos, la imagen.

Con el paso de los años se fue haciendo frecuente utilizarlas acompañando las notas periodísticas con el fin de captar la atención de los lectores, puesto que son de fácil comprensión y son utilizadas como vehículo fundamental para la transmisión de la información visual y gráfica, dándonos una noción de la noticia incluso antes de leerla, por lo que tener fotografías de alta calidad es vital.

Los medios de prensa cubanos también se han visto obligados a responder a esta necesidad, aunque de una manera poco eficiente ya que estos guardan internamente en sus servidores las imágenes, pero lo hacen a través de un sistema en línea, donde todos los procesos que deberían se automatizados son llevados a cabo manualmente sin tener en cuenta soluciones a problemas tales como:

- El duplicado de imágenes.
- La protección de la propiedad intelectual.
- La detección y extracción de textos.
- El ruido en las imágenes.

Siendo este último un gran inconveniente puesto que en muchas ocasiones tenemos un solo ejemplar de algún acontecimiento importante, pero no puede ser usado porque está presenta el efecto “ruido” que no es más que cuando el valor de un píxel de una imagen no se corresponde con la realidad y con el creciente auge de las cámaras digitales esto se ha vuelto mucho más común, ya que la mayoría de estas emplean sensores de baja calidad, sobre todo cuando se toman fotografías con luz escasa o se emplean altos valores ISO[1] y como los medios de prensa deben caracterizarse por la inmediatez y precisión, presentar este tipo de deficiencias puede traer consigo dificultades en el proceso de publicación de los artículos al no contar actualmente con una forma de reducir el ruido de una manera sencilla y a la vez eficaz.

[1] Sensibilidad que puede tener una película o un sensor a la luz.
[2] Sistema web perteneciente al proyecto Informatización de la Prensa de la Universidad de las Ciencias Informáticas,

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Dada la situación problémica planteada anteriormente surge el siguiente **problema científico:** ¿Cómo reducir el ruido en las imágenes digitales utilizadas por los medios de prensa cubanos?

Los algoritmos de reducción de ruido en las imágenes digitales constituyen el **objeto de estudio**, tomando como **campo de acción** la gestión de las mismas en los medios de prensa cubanos.

Idea de defender: La implementación de los algoritmos de reducción de ruido permitirá hacer uso de un mayor número de imágenes digitales en el proceso de publicación de las mismas en los medios de prensa cubanos.

Para este trabajo se plantea, como **objetivo general**, implementar algoritmos para la reducción de ruido en las imágenes digitales con el fin de mejorar la calidad de la información publicada por los medios de prensa cubanos, surgiendo entonces los siguientes **objetivos específicos**:

- Valorar los algoritmos existentes actualmente para lograr la reducción de ruido en las imágenes.
- Implementar los algoritmos valorados.
- Evaluar los algoritmos en cuanto a tiempo de procesamiento y recursos consumidos.
- Seleccionar los más eficientes para el tratamiento de las imágenes.
- Integrar estos algoritmos al producto Pyxel[2].

Para darle cumplimiento a estos objetivos se han definido las siguientes **tareas investigativas**:

- Seleccionar herramientas libres para acelerar el desarrollo del presente trabajo.
- Comprender las características de las imágenes.
- Interpretar los algoritmos de reducción de ruido.
- Recopilar información acerca de algoritmos paralelos.
- Seleccionar librerías y componentes para facilitar el tratamiento de las imágenes.
- Desarrollar la implementación de los algoritmos.
- Integrar los algoritmos implementados al producto Pyxel.
- Seleccionar una colección de imágenes con ruido para realizar las pruebas.
- Realizar las pruebas.
- Documentar los resultados obtenidos en las pruebas.

[2] Sistema web perteneciente al proyecto Informatización de la Prensa de la Universidad de las Ciencias Informáticas, encargado del almacenamiento y publicación de imágenes digitales.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Resultados esperados: Se pretenden identificar algoritmos que puedan ser usados para la reducción de ruido en las imágenes digitales que serán almacenadas en el producto Pyxel.
Los métodos científicos son los procedimientos que se utilizan para estudiar la realidad, la naturaleza, la sociedad y el pensamiento con el propósito de descubrir su esencia y sus relaciones.

Para la realización de la presente investigación se utilizaron los siguientes **métodos científicos de investigación:**

Métodos teóricos:

- **Analítico-Sintético**: Se realizó una búsqueda y un análisis profundo acerca de los algoritmos, teorías y documentos relacionados con el tema y a partir de esto se hizo una selección de los elementos más útiles para el desarrollo del presente trabajo.
- **Inductivo-Deductivo**: A partir de este método se obtuvo un conocimiento general sobre la reducción de ruido en imágenes sin dejar atrás las singularidades de este tema.
- **Histórico-Lógico:** A partir de este método se realizó un estudio con el fin de conocer la evolución y desarrollo de las técnicas actuales utilizadas para la reducción de ruido en las imágenes.

Métodos empíricos

- **Experimento**: A partir de este método se pudo probar las distintas versiones obtenidas en el transcurso de la implementación.

El presente trabajo de diploma se estructura en 3 capítulos.
Capítulo 1: Fundamentación Teórica. En este capítulo se abordan las principales características de los métodos existentes para la reducción de ruido en las imágenes, así como los principales conceptos relacionados con los mismos. Se realiza también una descripción de las herramientas, lenguajes y metodologías utilizadas en la implementación.
Capítulo 2: Métodos propuestos para la reducción del ruido en las imágenes. Se establece una comparación entre los diferentes algoritmos existentes en cuanto al tiempo de procesamiento y calidad de la imagen resultante, seleccionado los más eficientes para eliminar los diferentes tipos de ruidos.
Capítulo 3: Implementación y prueba. Se lleva a cabo la integración de los algoritmos propuestos al producto Pyxel y se realizan las pruebas con el propósito de demostrar que los mismos son los más eficientes.

CAPÍTULO 1: FUNDAMENTACIÓN TEÓRICA

Introducción

En este capítulo se tratan conceptos importantes con el propósito de comprender los temas relacionados con el proceso de reducción de ruido en imágenes. En él se verá un estudio del estado del arte sobre métodos existentes para la reducción de ruido y sistemas automatizados usados tanto en nuestro país como internacionalmente. Se analizan las principales tecnologías, técnicas, herramientas y metodologías usadas para desarrollar el sistema.

1.1 Informatización de la Prensa

Actualmente la Universidad de la Ciencias Informáticas se encuentra enfrascada en el Plan de Informatización de la Sociedad Cubana y como parte de este programa a partir de mayo de 2005 la UCI presta servicios tecnológicos a los Medios de Comunicación Masivos, donde el proyecto Programa de Informatización de la Prensa es su colaborador. Debido a los pocos resultados alcanzados se tuvo que replantear la Visión General del Proyecto siendo la gestión de imágenes el principal problema a resolver, con el objetivo de darle solución conflictos tales como:

- Demoras en la recuperación de imágenes.
- Recursos duplicados de información sin obtener los beneficios esperados de la redundancia.

Para darle solución a esta situación el proyecto decidió crear un sistema web llamado Pyxel. Este es esencialmente distribuido, es decir, que se pueden instalar todos sus subsistemas en un único servidor o en varios y que a su vez es independiente, permitiendo que otros interactúen con él y aprovechen toda su funcionalidad. El mismo también posibilita el almacenamiento y publicación de imágenes digitales siguiendo como referencia un grupo de estándares internacionales IPTC[3], brindando servicios tales como catalogación, búsquedas, etc.

1.2 Conceptos

Antes de explicar en qué consiste la reducción de ruido es necesario dar una serie de conceptos que ayudarán a un mejor entendimiento del tema planteado.

[3] Organismo internacional dedicado a promover estándares que permitan el intercambio de información entre las principales agencias de noticias del mundo.

1.2.1 Imagen Digital

Una imagen digital es un arreglo bidimensional de píxeles, donde el valor de cada píxel se representa mediante una función f, donde f(x, y) representa el nivel de brillantez, color o intensidad de la imagen en tales coordenadas. Por lo tanto, una imagen en blanco y negro puede ser representada por una matriz de dimensión M×N, donde f(x, y) {0, 1}. Además de la representación en blanco y negro, otras opciones son las imágenes en tonos de gris, para las cuales f(x, y) representa un nivel de intensidad típicamente asociado a un entero entre 0 y 255 y otras más sirven para describir imágenes en color[1].

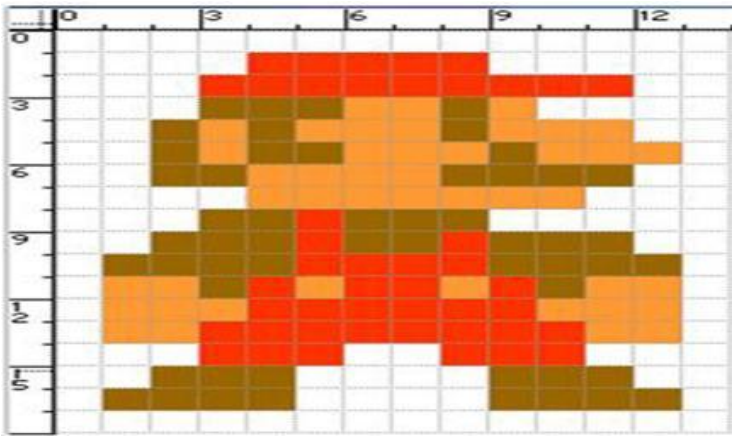

Fig. 1: Representación de una imagen digital

1.2.2 Imagen digital en escala de grises

Una imagen digital en escala de grises es un matriz de MxN elementos numéricos cuyos valores posibles van del 0 (negro) al 255 (blanco), siendo este número la intensidad luminosa en el determinado punto o píxel, por convención el origen de la imagen se encuentra en el extremo izquierdo superior.

Existen 2 tipos de imágenes en escala de grises:

- 1 bit por pixel, blanco y negro.
- 8 bit por pixel, blanco y negro.

En una imagen en escala de grises cada punto de la imagen se almacena en un Byte, donde su valor numérico representa su tono, que puede oscilar entre el blanco (255) y el negro (0). Esto quiere decir que es una imagen donde existen 256 tonos de gris (de 0 a 255, ambos inclusive). Es decir, la profundidad de color es el número de bits que definen cada píxel, que determinan el máximo número de colores que puede tener[2].

1.2.3 Imagen digital a color

Una imagen digital a color está formada por 3 matrices de MxN elementos numéricos cuyos valores posibles van del 0 (negro) al 255 (blanco), siendo este número la intensidad luminosa en cada una de las bandas espectrales del RGB (Rojo, Verde, Azul), de cada punto o píxel (picture element), a diferencia de las imágenes en escala de grises, las imágenes a color requieren de la combinación de las 3 bandas de color, para representar el color de un píxel. Por ejemplo, un determinado punto blanco de una imagen en escala de grises se describiría: P(x, y) = 255, sin embargo en una imagen a colores para describir el color del mismo punto se realizaría así: P(x, y) = (255, 255,255), esto debido a que el (0, 0, 0) corresponde al negro absoluto y el (255, 255, 255) al blanco absoluto).

Fig. 2: Síntesis aditiva de imágenes digitales a color

Con 32 bits por píxel también se siguen utilizando 24 bits para la representación del color. Los 8 bits restantes se utilizan para el denominado **canal alfa**, valor independiente del color que se asigna a cada píxel de la imagen, utilizado para definir el grado de transparencia de cada punto de la imagen. Un valor 0 indica que el punto es totalmente transparente, mientras que un valor 255 indica que será totalmente visible (opaco). Este tipo de imágenes pertenecen al modo RGBA.

Existen diferentes representaciones de imágenes de color (RGB, CMY, HSV, HSI, entre otras). La diversidad de representaciones del color se ha realizado para esclarecer el proceso de intensidad de tres componentes, Rojo, Verde y Azul (o *RGB* por sus siglas en inglés, de donde reciben su nombre). Este espacio se basa en el modelo de síntesis aditiva.

1.2.4 ¿Qué es el ruido en las imágenes?

El ruido en una imagen es la información no deseada que contamina la imagen. Variación en el nivel de gris que sufre un píxel debido a la aportación lumínica de la escena. Las imágenes pueden contener ruidos provocados por fuentes ruidosas, como son sensores ópticos, eléctricos, mecanismos de apertura en cámaras fotográficas, y también debido a la transmisión de dichas imágenes a través de un canal físico[3].

1.2.5 Ruidos de imagen típicos

- **Ruido de "sal y pimienta" o ruido aleatorio:** Consiste en un gran número de trastornos transitorios con una distribución de tiempo estadísticamente al azar. Ocurrencias aleatorias de píxeles completamente blancos y completamente negros. Es un ruido que aparece muchas veces producido por interferencias atmosféricas o por acciones hechas por el humano. Corresponde a la siguiente fórmula:

$$f_{sp}(x,y) = \begin{cases} f(x,y) & r < l_1 \\ 255(\text{salt}) & l_1 < r < l_2 \\ 0(\text{pepper}) & r > l_2 \end{cases}$$

Fig. 3: Imagen con ruido aleatorio

- **Ruido dependiente de la señal o ruido gaussiano**: Este tipo de ruido es producido generalmente por los propios sensores fotoeléctricos o también debido al grano de la película de una fotografía o cinta de video. Es un ruido dependiente de la señal. El nivel de gris de todos los píxeles de la imagen es perturbado con una normal. Es el ruido cuya densidad de probabilidad responde a la Distribución Gaussiana:

$$G = \frac{1}{\sqrt{2\pi\sigma^2}} e^{\frac{(g-m)^1}{2\sigma^1}}$$

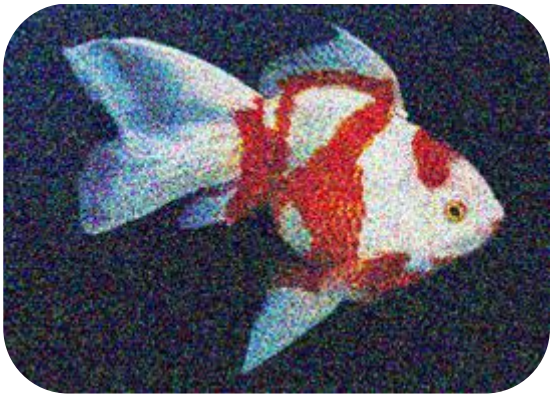

Fig. 4: Imagen con ruido gaussiano

1.2.6 Reducción de ruido

La reducción de ruido es el proceso al que se somete una imagen con la finalidad de reducir la información ruidosa que esta presenta. No hay forma automatizada de detectar el ruido presente en una imagen. Solo se logra mediante la apreciación visual del ojo humano.

1.3 Algoritmos de reducción de ruido

Antes de tratar los algoritmos se deben dar algunos conceptos importantes para el entendimiento de los mismos.

- **Entorno de vecindad:** La vecindad es una matriz bidimensional de valores de píxeles con un número impar de filas y columnas. El píxel de interés que normalmente es reemplazado por un

nuevo valor producto de la aplicación de un algoritmo, se ubica por lo general, en el centro de la vecindad.

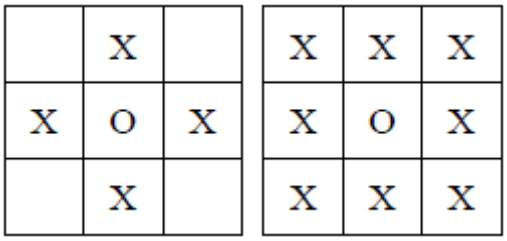

Fig. 5: a) Entorno de vecindad 4 b) Entorno de vecindad 8

- **Convolución:** Cada píxel de la imagen resultado es combinación lineal de varios píxeles vecinos de la imagen original. Las máscaras de convolución tienen por lo general un número impar de filas y columnas y su tamaño frecuentemente es 3x3, su contenido depende del tipo de procesamiento que se desea implantar.

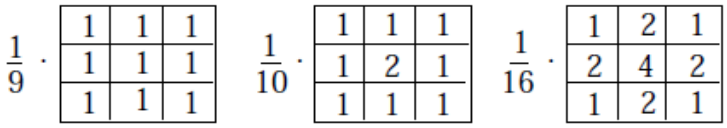

Fig. 6: Tres máscaras muy utilizadas en los filtros de reducción de ruido

Existen diferentes algoritmos dedicados a reducir el ruido en las imágenes digitales. Entre los más frecuentes se encuentran:

- **Filtros lineales:** Convolución de una imagen con una máscara de convolución.
- **Filtros no lineales:** Operación no lineal con los píxeles del entorno de vecindad.
- **Filtros temporales:** Análisis de varias imágenes de la misma escena tomadas en instantes diferentes de tiempo. Por sus características no serán abordados.

1.3.1 Filtros lineales

Los filtros lineales o filtros de convolución están definidos por la siguiente ecuación matemática:

$$g(i,j) = \sum_{k=-n/2}^{n/2} \sum_{l=-n/2}^{n/2} h(k,l) f(i-k, j-l)$$

- Para cada píxel en la imagen de entrada la máscara se ubica en el borde superior de este.
- Los valores de cada píxel en la imagen de entrada se multiplican por los pesos de la máscara correspondiente, todos estos valores se suman.
- El píxel central del entorno de vecindad toma el valor de esta suma.
- Este proceso se repite para toda la imagen, dando como salida una imagen nueva.

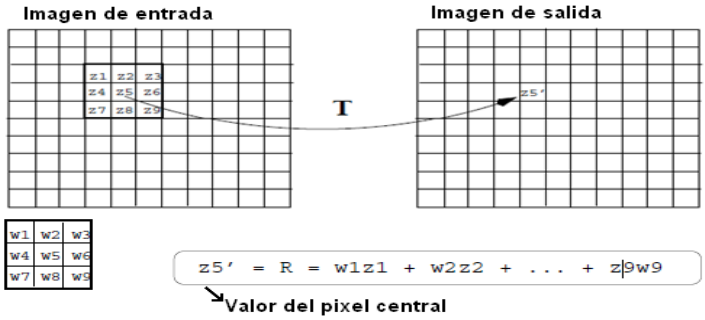

Fig. 7: Proceso de convolución de una imagen

El aplicar el proceso de convolución presupone un problema al aplicar este algoritmo, ya que la máscara inicialmente se sale de la imagen. Esto es solucionado con la llamada "banda basura". Para una máscara de tamaño n x n, donde n es un número impar, la banda basura es (n-1)/2. Al aplicar el algoritmo con una máscara de tamaño 3 x 3 se empezaría por el píxel (1,1) de la imagen de entrada.

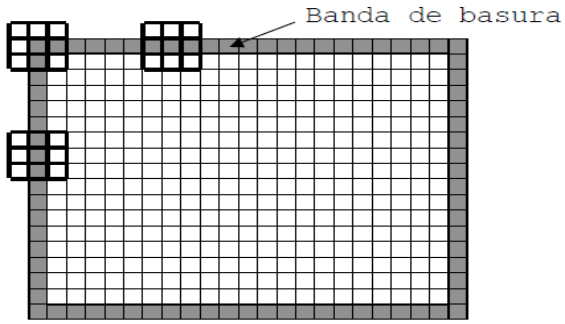

Fig. 8: Banda basura para una máscara de tamaño 3x3

El resultado del proceso de convolución es un filtrado paso bajo, donde el ruido es eliminado con más o menos intensidad según el tamaño de la máscara. No obstante, las técnicas de reducción de ruido presentan varios inconvenientes, ya que se puede observar una especie de pérdida de resolución de la imagen y/o reducción del contraste.

Al hacer pasar la máscara a la imagen se aprecia cómo se suavizan las transiciones más bruscas como pueden ser los bordes de los objetos. Este efecto es lógico, ya que esta parte de la imagen se caracteriza por tener componentes en alta frecuencia que de alguna forma son eliminados tras hacer la operación tipo kernel paso bajo. Existe por lo tanto un compromiso reducción de ruido-resolución esperada. La reducción de ruido es tanto mayor cuanto más grande sea el orden de la ventana (máscara). Por el contrario, la resolución de la imagen (calidad) disminuye con dicho orden.

1.3.1.1 Filtro de la media

La media del ruido en un entorno de vecindad es cero, si calculamos la media eliminamos el ruido. La imagen es constante en un entorno de vecindad, por lo que su valor es igual a la media. Pero esta característica no se cumple en los bordes. Se logra haciendo convolucionar la imagen con alguna de las siguientes máscaras:

$\frac{1}{49}$ X

1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1

$\frac{1}{25}$ X

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

$\frac{1}{9}$ X

1	1	1
1	1	1
1	1	1

Fig. 9: Máscaras más comunes usadas en el filtro de la media

También se puede implementar promediando los valores del entorno de vecindad. Este algoritmo es útil en la reducción de ruido gaussiano y aleatorio, aunque es más apreciable en el primer caso. Debido a sus características es susceptible a difuminar los bordes y a disminuir la nitidez, con la consecuente pérdida de detalles. Mientras mayor sea la cantidad de valores a promediar más ruido se reducirá, pero también habrá una mayor pérdida de detalles.

Fig. 10: a) Imagen con ruido gaussiano b) Imagen resultante del algoritmo de la media con una máscara de 3x3 c) Imagen resultante del algoritmo de la media con una máscara de 5x5

1.3.1.2 Filtro gaussiano

El filtro gaussiano funciona de una manera parecida al filtro de la media. O sea, los píxeles del entorno de vecindad modificarán al píxel central. Pero no todos los píxeles modifican con la misma "fuerza", cuanto más cerca estén del píxel que se está modificando, más valor tendrán en el cálculo del nuevo color. Para calcular la influencia de cada píxel del entorno de vecindad se utiliza la distribución gaussiana.

$$f(x) = ae^{-\frac{(x-b)^2}{2c^2}}$$

F(x) resulta un número real entre 0 y 1 que representa el porcentaje de influencia, a, b y c son constantes reales (a =1), (x-b) es la distancia entre el píxel que está siendo modificado y el píxel del que se está tomando el valor; c es la varianza del ruido. Hay que especificar que mientras mayor sea el valor de c mayor será el efecto de suavizado. A partir de esta función se construirá una máscara con el porcentaje de influencia de cada píxel del entorno de vecindad respecto al píxel central. Este filtro funciona correctamente con máscaras de tamaño 5x5 a 7x7.

0.0183	0.0821	0.1353	0.0821	0.0183
0.0821	0.3679	0.6065	0.3679	0.0821
0.1353	0.6065	1.0000	0.6065	0.1353
0.0821	0.3679	0.6065	0.3679	0.0821
0.0183	0.0821	0.1353	0.0821	0.0183

Fig. 11: Máscara gaussiana 5x5 con c=1

Una vez que se haya obtenido la máscara gaussiana se hace convolucionar con la imagen de entrada. Se suman los valores del resultado de la convolución y se dividen con la sumatoria de los valores de la máscara gaussiana. Este valor se le asigna al píxel central del entorno de vecindad, repitiéndose este proceso para todos los píxeles de la imagen que no pertenezcan a la "banda basura".

Fig. 12: a) Imagen original con ruido gaussiano b),c),d) Con filtrado gaussiano con c=1, 2, 3 respectivamente

Este filtro al igual que el filtro de la media disminuye la nitidez de la imagen con la consecuente pérdida de detalles. Pero esta disminución de nitidez es menor, debido a que el peso de los píxeles vecinos disminuye a medida que nos alejamos del centro. La figura 13 ilustra este proceso. El filtro gaussiano es utilizado sobre todo para reducir el ruido gaussiano, aunque también se puede usar en imágenes con ruido aleatorio.

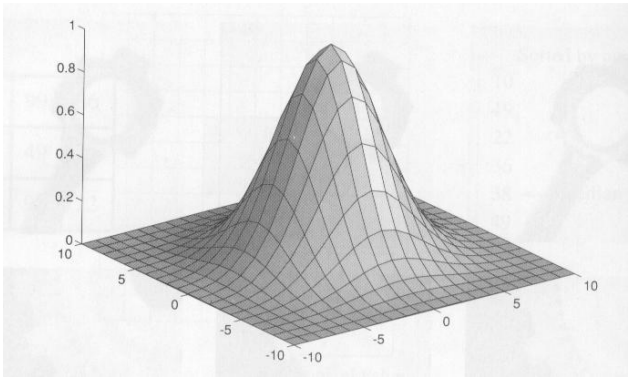

Fig. 13: Distribución gaussiana

1.3.1.3 Filtro binomial

El filtro binomial está hecho para ser eficiente computacionalmente hablando. El hecho de trabajar con pesos enteros supone obviamente una mayor eficiencia que trabajar con números reales. Y si la suma de todos los pesos tiene raíz entera, tanto mejor. En un ordenador, tanto multiplicar como dividir por potencia de 2 no supone ningún esfuerzo, ya que solamente consiste en desplazar bits para la izquierda o la derecha según se trate. De hecho, la mayoría de las máscaras se construyen de esta forma.

$\frac{1}{49}$

1	2	1
2	4	2
1	2	1

$\frac{1}{25}$

3	6	8	6	3
6	14	19	14	6
8	19	25	19	8
6	14	19	14	6
3	6	8	6	3

Fig. 14: Máscara binomial 3x3 y 5x5

Una vez escogida alguna de las máscaras binomiales más comunes se sigue el proceso de convolución con el propósito de restaurar la imagen. Este algoritmo es muy parecido al gaussiano.

1.3.2 Filtros no lineales

En el caso del filtrado lineal, la operación que se implementa es la convolución (en este caso bidimensional) de una matriz de píxeles con otra matriz de coeficientes. En el caso de un filtro no lineal la operación no es la convolución, sino que se le aplica algún algoritmo u operación a la matriz de píxeles.

1.3.2.1 Filtro de la mediana

Una de las técnicas no lineales más utilizadas para el procesado de imágenes, en cuanto a la reducción de ruido, es el filtrado de la mediana. Esta técnica fue desarrollada por Turkey[4] a finales de los años setenta y partió de la idea de conseguir un suavizado de imágenes y eliminación de ruido aplicando una técnica no lineal, pero de simple implementación.

[4] John Wilder Tukey: Uno de los grandes talentos estadísticos del siglo XX, con importantes contribuciones a la topología, Visualización de Información y en especial a la estadística.

La idea es que en un conjunto de píxeles cercanos, valores muy alejados del resto, serán valores que corresponden a píxeles ruidosos. La operación consiste en analizar una matriz de píxeles de N x N y reemplazar el píxel central por el valor de la mediana de todos ellos. Y por supuesto repetir el algoritmo recorriendo toda la imagen.

Aplicando este método se deben escoger ventanas de NxN con N impar, para tener bien diferenciado el píxel central. Tamaños habituales son ventanas de 3x3, de 5x5 y de hasta 7x7. Esto nos lleva a pensar en el hecho de la elección del tamaño de ventana como algo importante, puesto que un valor pequeño puede no eliminar bien el ruido, mientras que un valor demasiado alto es capaz de distorsionar la imagen. Para la elección de la ventana, en definitiva, no existe una regla fija, sino que se trata de escoger el orden que mejor resultados dé con una determinada imagen. En general, se suele decir que un tamaño de ventana es bueno si el número de píxeles ruidosos dentro de la ventana es menor que la mitad de píxeles de la ventana.

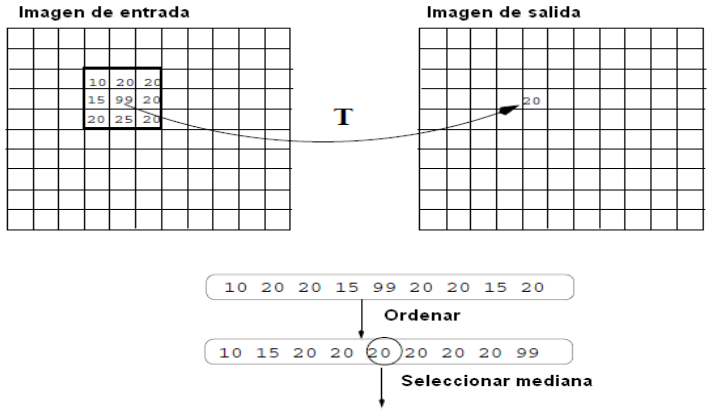

Fig. 15: Proceso del algoritmo de la mediana

Este filtro responde mucho mejor ante escalones de señal. O sea, un filtro de suavizado paso bajo es capaz de distorsionar un borde muy pronunciado, mientras que el filtro de mediana respeta los bordes. Elimina casi por completo el ruido de tipo impulsivo de las imágenes, mientras que un filtro de suavizado lineal no lo elimina, sino que lo difumina: por ejemplo, un punto blanco sobre fondo negro se convertiría en varios puntos gris oscuro. Escoger en el filtro de mediana es la forma de la ventana, lo que habitualmente se denomina máscara. De nuevo la elección de la máscara supone un grado de libertad más a la hora de filtrar nuestra imagen. No genera nuevos valores de grises.

Por tanto el resultado de filtrar una imagen binaria sigue siendo una imagen binaria. Otro de los factores importantes a escoger en el filtro de mediana es la forma de la ventana, lo que habitualmente se denomina máscara. De nuevo la elección de la máscara supone un grado de libertad más a la hora de filtrar nuestra imagen. La forma de la ventana nos dice los píxeles a los cuales se les obtiene la mediana, que finalmente deberá reemplazar al central.

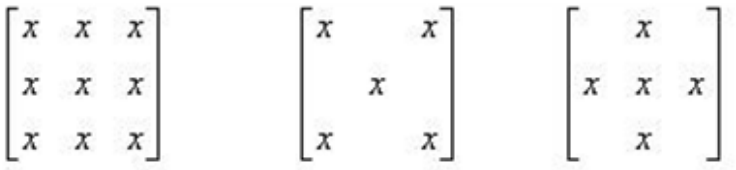

Fig. 16: Máscara cuadrada, en X y en cruz

La ventana cuadrada es la que mayor grado de reducción de ruido provoca, pero también es la que más fácilmente distorsiona la imagen.

- La ventana en cruz es buena para imágenes con muchos bordes.
- La ventana en equis es un término medio entre las dos anteriores.

El filtro de la mediana es útil a la hora de eliminar manchas localizadas de una imagen fotográfica, por ejemplo, deteriorada con el tiempo. También sirve en imágenes capturadas de la televisión con "nieve": ese molesto efecto que aparece si hay interferencias en el aparato de televisión, o bien si la relación señal a ruido no es muy buena. En ese caso de nuevo el filtro de la mediana es capaz de eliminar esas manchas de una forma eficiente sin afectar demasiado al resto de la imagen.

Fig. 17: a) Imagen con ruido aleatorio b) Imagen luego de aplicar algoritmo de la mediana

1.3.2.2 Alpha-Media del Entorno de Vecindad

Se calcula la media de la imagen sobre el conjunto formado por los píxeles de la imagen f en un entorno de vecindad del punto (x, y), eliminados los T de mayor y menor valor.

$$g(x,y) = \frac{1}{P.Q - 2T} \sum_{k=T}^{P.Q-1-T} f(k)$$

Este algoritmo presenta un buen compromiso para imágenes con ruido gaussiano y aleatorio simultáneamente.

1.3.2.3 Punto Medio del Entorno de Vecindad

La nueva imagen se genera a base de hallar la semisuma de los píxeles máximo y mínimo del conjunto formado por los píxeles de la imagen f en un entorno de vecindad del punto (x, y).

$$g(x,y) = \frac{f_{max}(i,j) + f_{min}(i,j)}{2} (i,j) \in S$$

Este algoritmo disminuye la nitidez, causa pérdida de detalles de forma y es más indicado para eliminar ruido gaussiano.

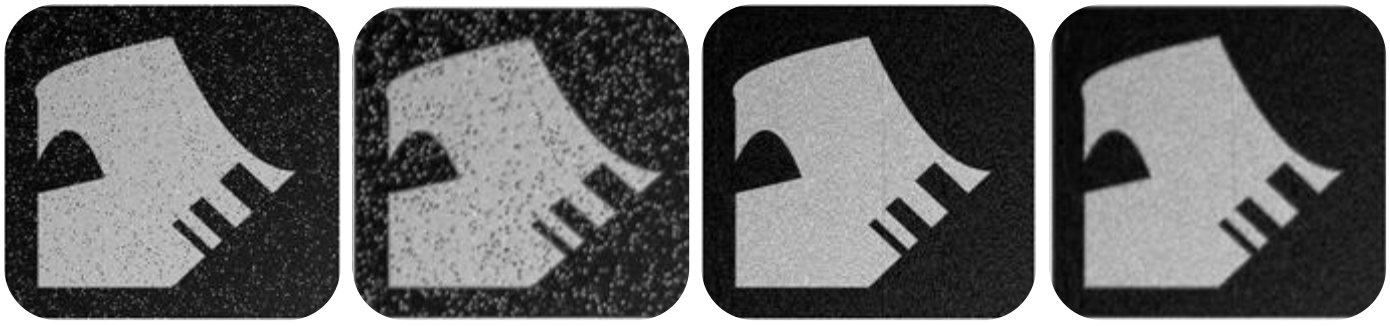

Fig. 18: a) Imagen con ruido aleatorio, b) Imagen con filtro aplicado, c) Imagen con ruido gaussiano, d) Imagen con filtro aplicado

1.3.2.4 Mediana Ponderada del Entorno de Vecindad

La nueva imagen g(x, y) se genera a base de hallar la mediana del conjunto formado por los píxeles de la imagen f, en un entorno de vecindad del punto (x, y), repetidos tantas veces como se indique en la máscara h(u,v). Una máscara h(u,v) muy utilizada:

1	2	1
2	4	2
1	2	1

Fig. 19: Máscara binomial de 3x3

1.4 Metodología de Desarrollo

Colección de documentación formal referente a los procesos, las políticas y los procedimientos que intervienen en el desarrollo del software. La finalidad de una metodología de desarrollo es garantizar la eficacia (p.ej. cumplir los requisitos iniciales) y la eficiencia (p.ej. minimizar las pérdidas de tiempo) en el proceso de generación de software[4].

1.4.1 SCRUM

Scrum es un proceso en el que se aplican de manera regular un conjunto de mejores prácticas para trabajar en equipo y obtener el mejor resultado posible de un proyecto. Estas prácticas se apoyan unas a otras y su selección tiene origen en un estudio de la manera de trabajar de equipos altamente productivos. En Scrum se realizan entregas parciales y regulares del resultado final del proyecto, priorizadas por el beneficio que aportan al receptor del proyecto. Por ello, Scrum está especialmente indicado para proyectos en entornos complejos, donde se necesita obtener resultados pronto, donde los requisitos son cambiantes o poco definidos, donde la innovación, la competitividad y la productividad son fundamentales. También se utiliza para resolver situaciones en que no se está entregando al cliente lo que necesita, cuando las entregas se alargan demasiado, los costes se disparan o la calidad no es aceptable, cuando se necesita capacidad de reacción ante la competencia, cuando la moral de los equipos es baja y la rotación alta, cuando es necesario identificar y solucionar ineficiencias sistemáticamente o cuando se quiere trabajar utilizando un proceso especializado en el desarrollo de producto[5].

1.5 Lenguajes Utilizados

Son herramientas que nos permiten crear programas y software. Disponen de formas adecuadas que permiten ser leídas y escritas por personas y que a su vez resultan independientes del modelo de computadora a utilizar.

1.5.1 Lenguaje de modelado

El lenguaje de modelado es la notación principalmente gráfica que usan los métodos para expresar un diseño. El proceso indica los pasos que se deben seguir para llegar a este.

La estandarización de un lenguaje de modelado es invaluable, ya que es la parte principal del proceso de comunicación que requieren todos los agentes involucrados en un proyecto informático. Si se quiere discutir un diseño con alguien más, ambos deben conocer el lenguaje de modelado y no así el proceso que se siguió para obtenerlo[6].

1.5.1.1 UML

El Lenguaje de Modelado Unificado (UML: Unified Modeling Language) es la sucesión de una serie de métodos de análisis y diseño orientados a objetos, fusionando los conceptos aportados por Booch, OMT y OOSE (Booch, G. et al., 1999).

Entre sus principales ventajas tenemos:

- Se puede usar para modelar distintos tipos de sistemas: sistemas de software, sistemas de hardware, y organizaciones del mundo real.
- Proporciona a los usuarios un lenguaje de modelado visual, expresivo y utilizable para el desarrollo e intercambio de modelos significativos.
- Ofrece nueve diagramas en los cuales modelar sistemas, entre se encuentran el Diagrama de casos de uso, colaboración, secuencia, estado, etc[6].

1.5.2 Lenguaje Python

Lenguaje de programación creado por Guido van Rossum[5] a finales de los años ochenta. Es un lenguaje similar a Perl, pero con una sintaxis muy limpia y que favorece un código legible.

[5] **Guido van Rossum:** Científico de la computación, más conocido por ser el autor del lenguaje de programación Python.

Se trata de un lenguaje interpretado, con tipado dinámico, fuertemente tipado, multiplataforma y orientado a objetos.

Entre sus principales características están su sintaxis simple, clara y sencilla, el gestor de memoria, la gran cantidad de librerías disponibles y la potencia del lenguaje, entre otros, todo esto hace que desarrollar una aplicación en Python sea sencillo y muy rápido.

También permite dividir el código en diferentes módulos que se pueden reutilizar en otros programas implementados en Python simplemente importándolos. Estos proporcionan multitud de aplicaciones para utilizar de base para la creación de programas teniendo también una gran variedad de módulos adicionales que pueden ser descargados y utilizados fácilmente. Algunos de los mismos han sido útiles para la implementación de la presente aplicación:

- PIL: Python Imaging Library, encargada del procesamiento de las imágenes.
- Pyro: Python Remote Object, facilita e implementa el trabajo de algoritmos en paralelo.

Por otra parte al ser un lenguaje de alto nivel, incluye tipos de datos que facilitan el uso de expresiones muy complejas con sentencias relativamente simples y uniendo esto a que no precisa de la declaración de variables y a que hace uso de la identación, hace que tengamos entonces un lenguaje muy sencillo y legible.

1.6 Herramientas

Conjunto de programas y ayudas que dan asistencia a los analistas, ingenieros de software y desarrolladores, durante todos los pasos del Ciclo de Vida de desarrollo de un Software. Conjunto de métodos, utilidades y técnicas que facilitan la automatización del ciclo de vida del desarrollo de sistemas de información, completamente o en alguna de sus fases.

1.6.1 Visual Paradigm

Visual Paradigm para UML es una herramienta profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño orientados a objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación. La herramienta UML CASE también proporciona abundantes tutoriales, demostraciones interactivas y proyectos UML.

1.6.2 Geany

Geany es un pequeño y ligero editor de textos para programadores y desarrolladores web, disponible tanto para Microsoft Windows como GNU/Linux. Dispone de las funciones básicas de un editor de estas características, siendo necesario para su uso tener instalados las librerías gtk2 runtime. Otras de las características son resaltado de sintaxis, cierre de códigos, autocompletados de etiquetas en XML y HTML, soporte de los lenguajes C, Java, PHP, HTML, Python, Perl, PASCAL y lista de símbolos. Disponiendo de versiones para Windows, Debian y Fedora Core.

1.7 ORB (Object Request Broker)

En computación distribuida, un intermediario de peticiones de objetos es una pieza de software que permite a los programadores realizar llamadas de programas de una computadora a otra a través de una red. ORB fomenta la interoperabilidad de los sistemas de objetos distribuidos, ya que permite a los usuarios construir sistemas juntando objetos de diferentes proveedores que se comunican entre sí a través del ORB. También maneja la transformación de las estructuras en proceso de datos hacia y desde la secuencia de bytes, que se transmite por la red. Esto se conoce como clasificación o serialización.

1.7.1 Pyro (Python Remote Objects)

El framework de objetos distribuidos "Pyro" es una maravillosa herramienta para la distribución de varios aspectos de la computación en red entre múltiples equipos. Pyro es un sistema específico de aplicaciones escritas en el lenguaje de programación Python, pero en el concepto es similar a tecnologías como RMI de Java (Remote Method Invocation), CORBA (Common Object Resource Broker Architecture), o incluso XML-RPC (XML Remote Procedure Call). Por supuesto, en consonancia con la filosofía de Python, Pyro es mucho más fácil de usar que sus competidores. Permite también a los usuarios de una red compartir casi todos los recursos y que a su vez pueda ser distribuido entre las diferentes partes[7].

1.8 Análisis de sistemas existentes

En la actualidad existen numerosas empresas en la que la calidad de las imágenes es primordial. Muchas de estas han producido softwares con el propósito de reducir el ruido en las imágenes digitales, efecto nocivo que afecta de manera visible el aspecto de estas. Algunos de los más populares hoy en día son:

- Kodak GEM Professional v2

- Noise Ninja v2.1.2
- Neat Image v6.1
- Adobe Photoshop CS2
- Gimp 2.4

1.8.1 Kodak GEM Professional v2

El kit profesional de la KODAK DIGITAL GEM tiene la funcionalidad premiada para dar a usuarios más control y mejores resultados para reducir ruido y grano en sus imágenes digitales de 8 bits o de 16 bits. El kit profesional de la KODAK DIGITAL GEM proporciona dos únicos algoritmos de gran alcance:

- Supresión de " Ruido/Grueso".
- Supresión de " Ruido/Fino y Grano".

Cada algoritmo se modifica para requisitos particulares para los diversos tipos de ruido y de grano de la imagen, y cada uno tiene controles separados asociados a él. El algoritmo supresión de " Ruido/Grueso" es muy similar al algoritmo usado dentro del enchufe original de la DIGITAL GEM. Este algoritmo es muy agresivo y trabaja mejor para problemas de ruido más extremos. El algoritmo de supresión de " Ruido/Fino y Grano" no es tan agresivo, sino que se usa específicamente para quitar problemas del grano, así como ruido, mientras que preserva más detalle dentro de la imagen.

1.8.2 Noise Ninja v2.1.2

Noise Ninja es la más efectiva y productiva solución para remover ruido y grano de fotografías digitales y escaneadas. Es una herramienta para cualquier persona que esté en una situación de acción rápida, donde se requiere una ISO alta de fotografía. Noise Ninja es usado en siete de los más grandes periódicos norteamericanos. Es la guía de miles de profesionales y fotógrafos amateurs.

1.8.3 Neat Image v6.1

Neat Image es un filtro para reducir ruido y grano en imágenes fotográficas producidas por imágenes digitales y scanners. Neat Image es indispensable en fotografías de luz baja (en lugares cerrados, de noche, sin flash) y alta velocidad (deporte, acción y niños). Provee las mejores técnicas de reducción de ruido. Reduce eficientemente los siguientes tipos de ruidos:

- Ruido de alta ISO en cámaras digitales.

- Granos en imágenes escaneadas e impresas.

1.8.4 Adobe Photoshop CS2

Adobe Photoshop CS2 es un software profesional de edición de imágenes y retoques fotográficos, es una herramienta imprescindible para el tratamiento de fotografías. Dispone de una gran gama de herramientas novedosa.

Entre ellas están:

- Control de varias capas.
- Acceso rápido a ficheros.
- Creación de imágenes para ficheros Web.
- Deformación de imágenes.
- Gran variedad de filtros.
- Procesamiento de lotes de archivos Camera Raw digitales.
- Objetos inteligentes.

Otra novedad de esta versión es la mejora de la visualización de las imágenes digitales y la creación de imágenes en 3D. El filtro de reducción de la versión CS2 de Photoshop funciona muy bien y lo que no es menos importante, es fácil e intuitivo. Además, es el único que ha aprovecha dos CPUs en la práctica.

1.8.5 Gimp 2.4

Este programa de código abierto ofrece múltiples herramientas para el trabajo con imágenes. También brinda un conjunto de filtros para desenfocar imágenes con el propósito de reducir ruido. Entre ellos están:

- Desenfoque Gaussiano.
- Desenfoque simple.
- Desenfoque selectivo.
- Pixelizar.
- Desenfoque de movimiento.

Conclusiones

En este capítulo se definieron conceptos importantes para el entendimiento de los procesos relacionados con la reducción de ruido en las imágenes. Se realizó también un estudio acerca de los principales métodos existentes así como de las tecnologías, técnicas, herramientas y metodologías usadas para desarrollar el sistema.

CAPÍTULO 2: ALGORITMOS PROPUESTOS

Introducción

En el presente capítulo se analizan los algoritmos existentes actualmente para eliminar cada uno de los diferentes tipos de ruidos, analizando detalladamente sus principales características con el fin de escoger los más ideales para la futura integración con Pyxel, donde se hace necesario que los mismos trabajen eficientemente en sistemas de bajas prestaciones.

2.1 Arquitectura de los algoritmos de reducción de ruido en las imágenes

Para la manipulación de imágenes Python tiene una librería llamada PIL (Python Imaging Library) que brinda funciones de carga, muestra y salva de las imágenes; así como operaciones píxel a píxel, convolución y trabajo con algunos filtros.

Los algoritmos propuestos reciben como entrada una imagen ruidosa, cuyo ruido solo puede ser determinado por el usuario, ya que actualmente no hay forma automatizada de hacerlo, esto dependiendo del algoritmo a utilizar varía en cuanto a los pasos a seguir, ya que cada uno presenta diferentes características, las cuales vienen más detalladas posteriormente. Después de aplicados dichos métodos se obtiene una imagen resultante que puede estar parcial o totalmente libre de ruido.

Este mecanismo presenta la siguiente estructura:

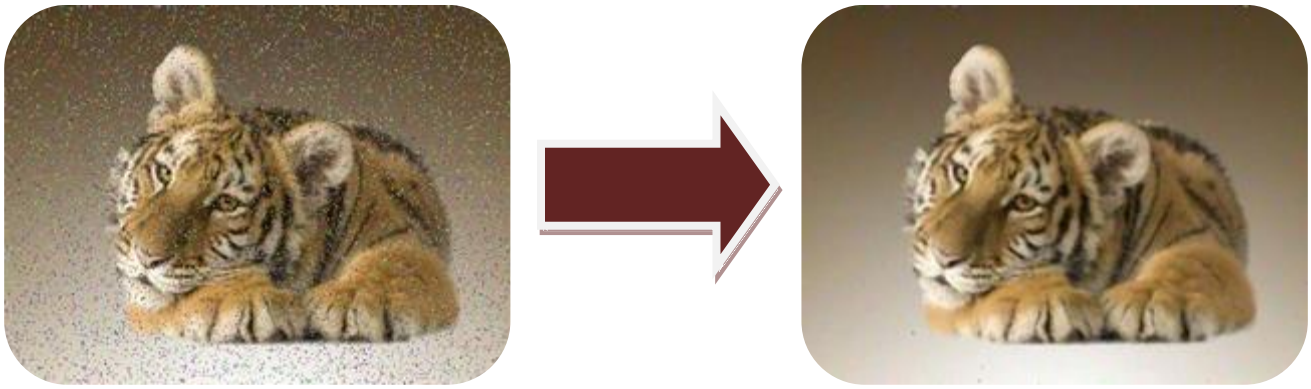

Fig. 20: a) Imagen original con ruido **b) Imagen resultante**

- **Aclaraciones**

Para lograr una óptima reducción de ruido es necesario primeramente saber qué tipo presenta la imagen a tratar, para así poder aplicarle el algoritmo más eficiente, de lo contrario puede que el resultado no sea el esperado.

También es necesario aclarar que las imágenes pueden contener mucho ruido por lo que someterla a dichos algoritmos solo traerá consigo que empeore su estado. Un ejemplo de esto se puede ver en la figura 21.

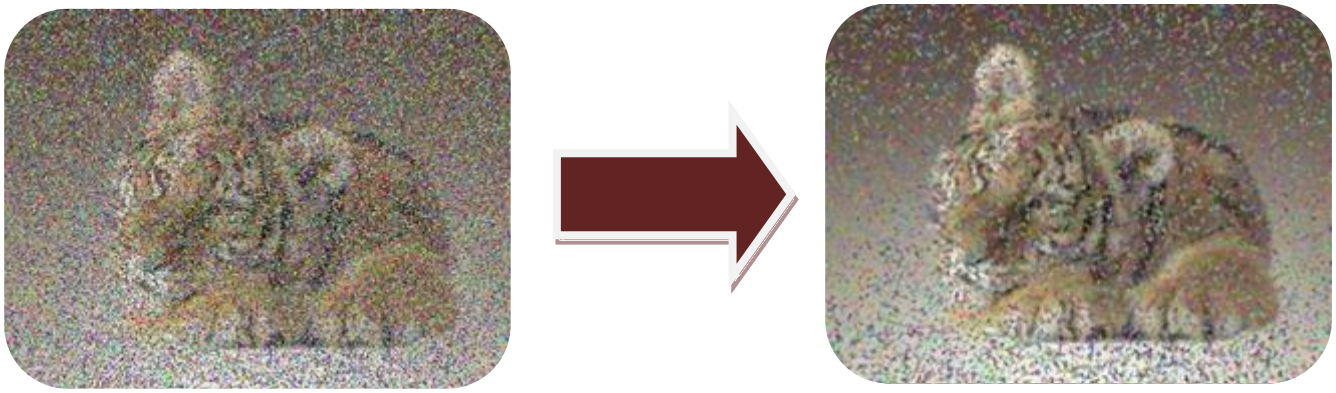

Fig. 21: a) Imagen original con mucho ruido **b) Imagen resultante**

2.2 Algoritmo propuesto para la reducción de ruido gaussiano.

Como se ha explicado anteriormente los filtros lineales funcionan mediante la convolución. Este proceso puede implementarse de varias formas.

La clase Image de Python Imaging Library (PIL) brinda la función getdata(), la cual devuelve una lista de los valores en cada una de las posiciones (x,y) de la imagen. Así, en cada iteración de los algoritmos se accede píxel a píxel a los valores de la imagen de entrada aplicando dicho proceso, aunque esta no es la forma más óptima pues funciona de forma muy lenta.

PIL también implementa la clase ImageFilter para el trabajo con filtros en imágenes. Para el proceso de convolución existe la función llamada ImageFilter.Kernel (sizek, kernel), donde sizek sería el tamaño de la máscara de convolución y kernel una lista o diccionario con los valores de dicha máscara.

Esta función tiene entre sus inconvenientes que solo funciona con máscaras de tamaño 3x3 o 5x5; además de que no soporta el formato .gif.

A pesar de esto, es increíblemente rápido el proceso de obtención de la imagen de salida, siendo en su mayoría inferior a 1 segundo. Debido a esto es preferible implementar los filtros lineales valiéndose de dicha función.

Por lo planteado anteriormente se hace necesario establecer una comparación entre los diferentes algoritmos capaces de reducir el ruido gaussiano con el objetivo de escoger el que mejor comportamiento y características presente para la posterior integración con el producto Pyxel. Se aplicará cada algoritmo con una imagen de prueba de dimensión 288x209 píxeles.

Media:

- Tiempo de ejecución para máscara de 3x3: 0.0296 segundos.
- Tiempo de ejecución para máscara de 5x5: 0.0342 segundos.
- Para lograr una mayor reducción de ruido es necesario aumentar el entorno de vecindad, pero trae consigo que el algoritmo se torne un poco más lento.
- Es el algoritmo que más desenfoque produce y el que más afecta los bordes debido a que la media del ruido no es constante en estos.

Fig. 22: a) Original con ruido gaussiano b) Media con máscara 3x3 c) Media con máscara 5x5

Gaussiano:

- Funciona con la formula gaussiana, aunque se puede trabajar con la siguiente variante que solo toma en cuenta fuerza de los píxeles vecinos y produce un menor desenfoque.

$$G(x,y) = e^{-(x+y)^2/2\sigma^2} \qquad G(x,y) = e^{-(x,y)^2}$$

a) Función gaussiana original b) Función gaussiana modificada

- Debido a que la función Kernel(x, y) de ImageFilter solo funciona con tamaños de máscaras de 3x3 o 5x5 y el filtro gaussiano presenta un buen comportamiento solo para tamaños de 5x5 o 7x7 es necesario fijar la máscara a 5x5. Esto no constituye un problema ya que el valor del umbral determinará la fuerza con que el algoritmo atacará al ruido.
- Tiempo de ejecución para función modificada (umbral=0): 0.0348 segundos.
- Tiempo de ejecución con umbral=1: 0.0350 segundos.
- Tiempo de ejecución con umbral=2: 0.0352 segundos.
- Tiempo de ejecución con umbral=3: 0.0359 segundos.

Fig. 23: a) Imagen original b) Con función modificada c) Con umbral = 1

Mediana Ponderada del Entorno de Vecindad:

- Extremadamente lento, tiempo de ejecución con máscara binomial 3x3: 34.092 segundos.
- Es el único de los filtros no lineales que presenta un buen comportamiento frente a este tipo de ruido.

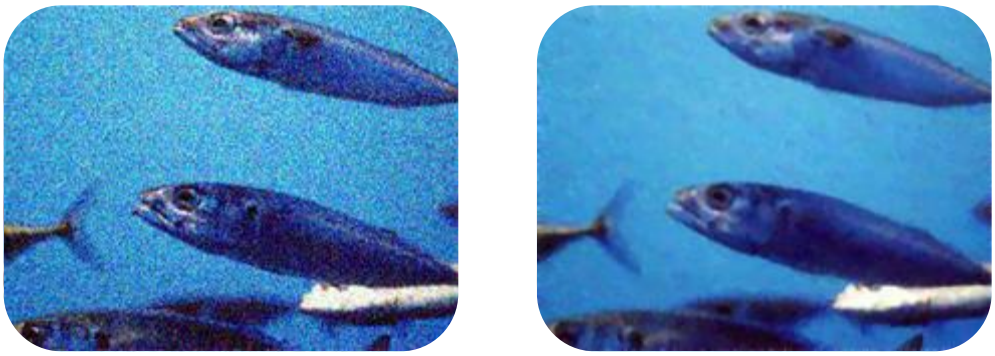

Fig. 24: a) Imagen original b) Con algoritmo de la Mediana Ponderada del Entorno de Vecindad con tamaño 3x3

Binomial:

- Tiempo de ejecución con máscara binomial 3x3: 0.1903 segundos.
- Tiempo de ejecución con máscara binomial 5x5: 0.0341 segundos.
- Para lograr una mayor reducción de ruido es necesario aumentar el entorno de vecindad, lo que incrementa imperceptiblemente la velocidad del algoritmo.
- Funciona bajo el principio de la campana de Gauss, aunque trabaja con pesos enteros lo que provoca que los valores no sean tan exactos como en el filtro gaussiano

Fig. 25: a) Imagen original con ruido gaussiano b) Binomial con máscara 3x3 c) Binomial con máscara 5x5

Para la integración con Pyxel los algoritmos escogidos para la reducción del ruido gaussiano por las características analizadas son el filtro binomial y el filtro gaussiano, pues logran un buen balance entre rapidez y eficacia con la manipulación de las imágenes digitales.

2.3 Algoritmos propuestos para la reducción de ruido aleatorio

Los filtros capaces de lograr un buen resultado con imágenes que presentan ruido aleatorio son en su mayoría los filtros no lineales, los que llevan a cabo alguna operación no lineal. En este caso también puede servir la función getdata() de la clase Image de PIL.

La clase ImageFilter brinda además varios filtros para la reducción de ruido aleatorio. Algunas de las funciones útiles son:

- MinFilter(): Elimina el ruido de tipo sal.
- MaxFilter():Elimina el ruido de tipo pimienta.
- MedianFilter(): Implementación optimizada del filtro mediana. Elimina el ruido de tipo sal y pimienta.

Antes de seleccionar un algoritmo para la reducción del ruido aleatorio se hace igualmente necesario establecer comparaciones entre estos con la finalidad de escoger el más apropiado. Se ha escogido una imagen de prueba de tamaño 1166x508 píxeles.

Punto Medio del Entorno de Vecindad:

- Tiempo de ejecución con máscara de 3x3 es de 124.515 segundos. Los elementos del entorno de vecindad son añadidos a una lista, luego esta es organizada de menor a mayor. Se obtiene la semisuma de los extremos de la lista ordenada. El nuevo valor será el asignado al píxel central.
- Tamaño de entorno de vecindad mínimo es de 3x3.
- Presenta un mal comportamiento frente a este tipo de ruido.

Fig. 26: a) Imagen original con ruido aleatorio b) Punto Medio del Entorno de Vecindad

Mediana Ponderada del Entorno de Vecindad:

- Mucho más lento que los demás algoritmos, su tiempo de ejecución con una máscara de 3x3 es de 336.663 segundos. Primeramente utiliza el proceso de convolución con una máscara binomial. Luego de esto el proceso es igual al de la mediana. El uso de la convolución y del ordenamiento de la lista en cada iteración hacen de este un filtro especialmente lento.
- Tamaño de entorno de vecindad mínimo es de 3x3.
- Presenta un óptimo comportamiento frente a este tipo de ruido.

Fig. 27: a) Imagen original con ruido aleatorio, b) Mediana ponderada del entorno de vecindad.

Mediana:

- De los filtros no lineales analizados es el más rápido, su tiempo de ejecución es de 0.7846 segundos. Los elementos del entorno de vecindad son añadidos a una lista, luego esta es organizada de menor a mayor. La mediana de la lista ordenada será el nuevo valor que recibirá el píxel central.
- Tamaño de entorno de vecindad mínimo es de 3x3.
- Presenta un óptimo comportamiento frente a este tipo de ruido.

Fig. 28: a) Imagen original con ruido aleatorio b) Filtro de la Mediana

Dadas las características de los algoritmos analizados el escogido para reducir el ruido aleatorio y para integrarlo al producto Pyxel fue el filtro de la mediana. Este es muy rápido, además de presentar un comportamiento en la mayoría de los casos espectacular.

Conclusiones

En este capítulo se vieron los algoritmos existentes más usados en la actualidad estableciendo una comparación entre los mismos en cuanto a sus ventajas y desventajas para finalmente escoger los tres más óptimos que eliminen el ruido de una manera sencilla y eficaz para hacer la integración de estos con el producto Pyxel.

CAPÍTULO 3: IMPLEMENTACIÓN Y PRUEBA

Introducción

En este capítulo partiendo de los resultados obtenidos en el diseño se lleva a cabo el flujo de trabajo de implementación y prueba. Se realiza también la integración de los algoritmos tratados al producto Pyxel con el fin de lograr un sistema óptimo y funcional.

3.1 Implementación

En este flujo de trabajo se convierten las clases y objetos en ficheros fuente de manera que estos sean posibles de integrar a Pyxel. El mismo es un sistema distribuido que está compuesto por varios subsistemas, los cuales pueden en potencia correr en varias máquinas conectadas en red. Esto trae varias ventajas:

- Los problemas que ocurran en uno de los subsistemas no se propagan por todo el sistema. Por tanto este es más estable y los errores pueden ser corregidos con mayor facilidad.
- Se puede modificar uno de los subsistemas, incluso reemplazarlo por una nueva implementación, sin tener que detener el sistema completo.

El sistema de reducción de ruido será uno de los futuros subsistemas con que contará Pyxel. En este sentido se implementa el sistema en términos de componentes. Los componentes conforman lo que se conoce como un modelo de implementación al describir los componentes a construir, su organización y dependencia entre nodos físicos en los que funcionara la aplicación.

Entre los componentes podemos encontrar:

- binomial.py: Módulo encargado de filtrar la imagen con el algoritmo binomial.
- gaussiano.py: Módulo encargado de realizar el filtrado gaussiano.
- mediana.py: Módulo encargado del filtrado de la imagen con el algoritmo de la mediana.
- distributed.py: Módulo encargado del procesamiento distribuido de los diferentes algoritmos.

Estos componentes están estrechamente relacionados con las librerías utilizadas para el desarrollo de la aplicación como se muestra en la figura 29.

3.2 Diagrama de Componentes

Esquema o diagrama que muestra las interacciones y relaciones de los componentes de un modelo, entendiéndose como componente a una clase de uso específico, que puede ser implementada desde un

entorno de desarrollo, ya sea de código binario, fuente o ejecutable; dichos componentes poseen tipo, que indican si pueden ser útiles en tiempo de compilación, enlace y ejecución.

Este tipo de diagrama se representa mediante componentes unidos mediante relaciones de dependencia (generalmente de compilación)[8].

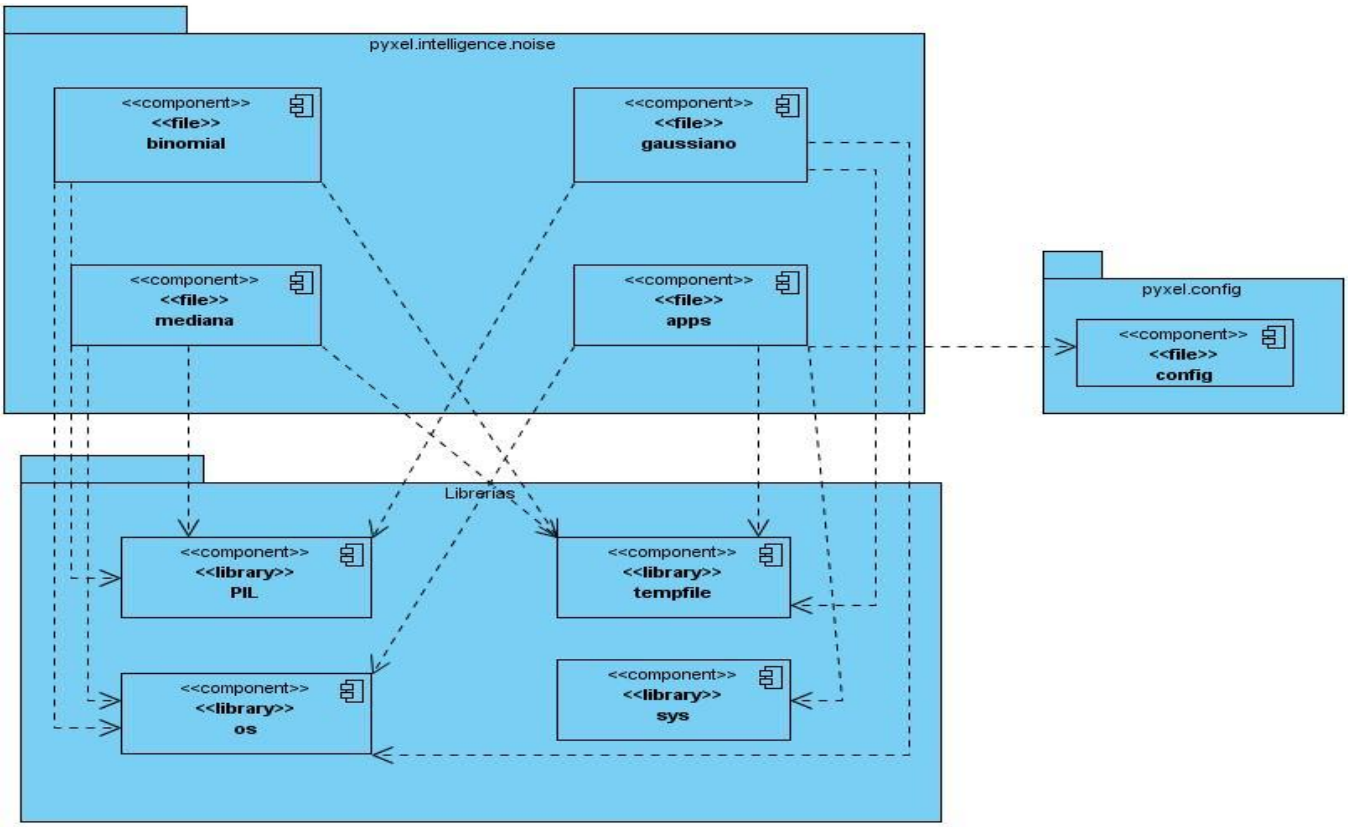

Fig. 29: Diagrama de componentes

En la carpeta **pyxel.intelligence.noise** se encuentran los componentes **binomial.py, gaussiano.py**, **mediana.py** y **apps.py**. A continuación se explican cada uno de ellos detalladamente:

- **binomial.py:** Módulo encargado de reducir el ruido mediante un filtro binomial usando 2 máscaras de convolución.

- **gaussiano.py:** Módulo encargado de reducir el ruido mediante un filtro gaussiano y utiliza para esto el valor umbral que determina la fuerza con que será atacado el mismo.

- **mediana.py:** Módulo encargado de reducir el ruido mediante el filtro de la mediana brindado por la clase ImageFilter de la librería PIL.

- **apps.py:** Módulo que brinda comandos para la consola de Linux para el procesamiento de imágenes con los diferentes filtros.

Para la comunicación Pyxel utiliza un ORB, en particular Pyro. Lo que permite tener objetos remotos y llamar a sus métodos de forma casi natural. Para esto, Pyxel tiene una capa de abstracción sobre Pyro, llamada pyxel.launcher, que simplifica el uso de Pyro para Pyxel y además también logra desacoplar la implementación de los servicios y nodos de Pyro.

Los servicios son simplemente procesos que están corriendo permanentemente. La forma que tienen los servicios de comunicarse entre sí (en Pyxel) es pasándose mensajes. Los nodos son también procesos muy similares a los servicios (de hecho cada nodo corre con la misma infraestructura de los servicios), la diferencia es que pueden haber varios nodos del mismo tipo corriendo en la LAN; no obstante, cada nodo tiene un nombre diferente.

Así, el subsistema **pyxell.intelligence.noise** contará entre sus nodos con los ficheros:

- binomial.py
- gaussiano.py
- mediana.py

Además, como servicios tendrá:

- binomial.py
- gaussiano.py
- mediana.py
- distributed.py

3.3 Experimentos y pruebas

El enfoque principal de los experimentos realizados en esta etapa es asegurarse de que los diferentes algoritmos propuestos son adecuados para la reducción de ruido en imágenes digitales. La implementación fue hecha en Python, utilizando una plataforma Unix, en estaciones de trabajo Ubuntu Linux, Intel(R) Core(TM)2 Duo CPU 2.00Hz.

3.3.1 Pruebas Realizadas

Para llevar a cabo las pruebas se hizo la selección de una colección de imágenes que fueron sometidas a diferentes niveles de ruido. En el caso de las imágenes a las que se les añadió el ruido gaussiano se utilizó Photoshop CS4, y para el caso del ruido aleatorio Gimp 2.4.0.

- **Aclaraciones**

  A las imágenes resultantes luego de aplicarle los algoritmos propuestos se les asignó una categoría según apreciación visual de la calidad de estas.

3.3.1.1 Reducción de ruido aleatorio aplicando el filtro de la mediana

Tabla 1: Resultados obtenidos después de aplicado el filtro de la mediana

Nombre de la imagen	Porciento de ruido	Dimensiones de la imagen	Tiempo de ejecución	Calidad de la imagen resultante
piton.jpeg	5%	1280 x 800	1.51936224837 s	Alta

tortuga.jpeg	10%	1024 x 768	1.76245782829 s	Alta
tomates.jpeg	15%	1280 x 800	1.70172131649 s	Media
perros.jpeg	20%	1024 x 768	1.37334089555 s	Media
morsa.jpeg	25%	1024 x 768	1.58650351386 s	Media
flamenco.jpeg	30%	1024 x 768	2.16483989701 s	Baja
cotorra.jpeg	35%	1024 x 768	2.35228974632 s	Baja
flor.jpg	40%	442 x 500	0.886208722058 s	Baja
caracol.jpeg	45%	1024 x 768	2.23883929382 s	Baja
camello.jpeg	50%	1280 x 800	2.67702459391 s	Baja

Los experimentos realizados con el algoritmo de la mediana arrojaron que para un intervalo de ruido del 0% al 14%, la calidad de las imagenes resultantes es alta, del 15% al 29% media y del 30% en adelante es baja. Se comprobó también que la velocidad del algoritmo es rápida incluso con imágenes de gran tamaño. Por lo visto anteriormente este filtro demostró ser el más óptimo para reducir este tipo de ruido.

- **Aclaraciones**

  En estas pruebas no se contempló una segunda iteración del algoritmo con la imagen resultante. De haberlo hecho esto traería consigo una mayor reducción del ruido pues cada vez que se aplica nuevamente el filtro este va disminuyendo.

Fig. 30: a) Imagen con ruido aleatorio al 5% **b) Imagen resultante**

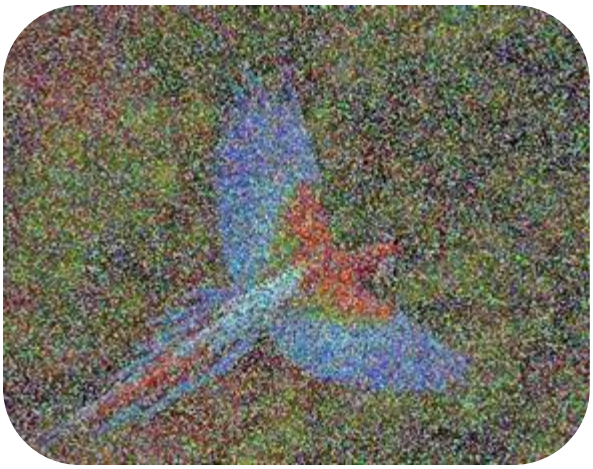
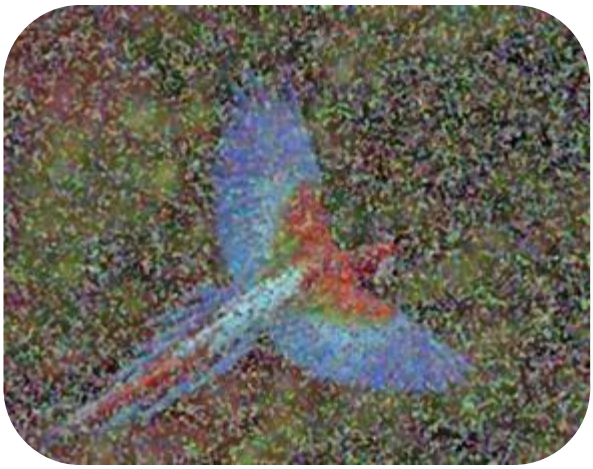

Fig. 31: a) Imagen con ruido aleatorio al 50% **b) Imagen resultante**

3.3.1.2 Reducción de ruido gaussiano aplicando el filtro binomial con máscara de 3 x 3

Tabla 2: Resultados obtenidos después de aplicado el filtro binomial con máscara de 3x3

Nombre de la imagen	Porciento de ruido	Dimensiones de la imagen	Tiempo de ejecución	Calidad de la imagen resultante
tigre.jpg	5%	400 x 300	0.34881827823 s	Alta
mariposa.jpg	10%	400 x 300	0.423359829356 s	Alta
amapola.jpg	15%	442 x 500	0.475208503866 s	Alta
aguila.jpg	20%	1024 x 768	0.962861795761 s	Alta
pez.jpg	25%	1024 x 768	1.05813551793 s	Alta
cocodrilo.jpg	30%	1280 x 800	1.31098670644 s	Media
camello.jpg	35%	1280 x 800	1.31855424743 s	Media
caracol.jpg	40%	1024 x 768	1.14980609519 s	Media
cotorra.jpg	45%	1024 x 768	1.44228736768 s	Media
flamenco.jpg	50%	1024 x 768	1.42083362318 s	Baja

3.3.1.3 Reducción de ruido gaussiano aplicando el filtro binomial con máscara de 5 x 5

Tabla 3: Resultados obtenidos después de aplicado el filtro binomial con máscara de 5x5

Nombre de la imagen	Porciento de ruido	Dimensiones de la imagen	Tiempo de ejecución	Calidad de la imagen resultante
tigre.jpg	5%	400 x 300	0.225398211572 s	Alta
mariposa.jpg	10%	400 x 300	0.383525220637 s	Alta
amapola.jpg	15%	442 x 500	0.504325464271 s	Alta
aguila.jpg	20%	1024 x 768	0.842312021731 s	Alta
pez.jpg	25%	1024 x 768	0.76737617551 s	Alta
cocodrilo.jpg	30%	1280 x 800	1.12576586255 s	Alta
camello.jpg	35%	1280 x 800	1.25717444174 s	Media
caracol.jpg	40%	1024 x 768	1.13573311892 s	Media
cotorra.jpg	45%	1024 x 768	1.01429686542 s	Media
flamenco.jpg	50%	1024 x 768	0.984215990868 s	Baja

Para realizar las pruebas con este algoritmo también se utilizaron imágenes ruidosas con diferentes niveles que van del 0% al 50%, donde se evidencia que aproximadamente hasta con un 30% de ruido las imágenes resultantes aun son de alta calidad. Para llevar a cabo dicho experimento se les aplicó a cada imagen las máscaras de convolución de 3 x 3 y 5 x 5.



Fig. 32: a) Imagen con ruido gaussiano al 5%, b) Binomial con máscara de 3 x 3, c) Binomial con máscara de 5 x 5

Fig. 33: a) Imagen con ruido gaussiano al 50%, b) Binomial con máscara de 3 x 3, c) Binomial con máscara de 5 x 5

3.3.1.4 Reducción de ruido gaussiano aplicando el filtro gaussiano con umbral=0

Tabla 4: Resultados obtenidos después de aplicado el filtro gaussiano con umbral=0

Nombre de la imagen	Porciento de ruido	Dimensiones de la imagen	Tiempo de ejecución	Calidad de la imagen resultante
tigre.jpg	5%	400 x 300	0.371612896614 s	Buena
mariposa.jpg	10%	400 x 300	0.408653585189 s	Buena
amapola.jpg	15%	442 x 500	0.444261430476 s	Buena
aguila.jpg	20%	1024 x 768	1.09408080489 s	Buena
pez.jpg	25%	1024 x 768	1.02027492601 s	Media
cocdrilo.jpg	30%	1280 x 800	1.96198427454 s	Media
camello.jpg	35%	1280 x 800	1.82733518531 s	Baja
caracol.jpg	40%	1024 x 768	1.69867209805 s	Baja
cotorra.jpg	45%	1024 x 768	1.64208591254 s	Baja
flamenco.jpg	50%	1024 x 768	1.60085673977 s	Baja

3.3.1.5 Reducción de ruido gaussiano aplicando el filtro gaussiano con umbral=4

Tabla 5: Resultados obtenidos después de aplicado el filtro gaussiano con umbral=4

Nombre de la imagen	Porciento de ruido	Dimensiones de la imagen	Tiempo de ejecución	Calidad de la imagen resultante
tigre.jpg	5%	400 x 300	0.400384367751 s	Buena
mariposa.jpg	10%	400 x 300	0.383885755233 s	Buena
amapola.jpg	15%	442 x 500	0.486768019931 s	Buena
aguila.jpg	20%	1024 x 768	0.916377348687 s	Buena
pez.jpg	25%	1024 x 768	0.846361361485 s	Buena
cocodrilo.jpg	30%	1280 x 800	1.03185955839 s	Buena
camello.jpg	35%	1280 x 800	1.53765000233 s	Media
caracol.jpg	40%	1024 x 768	1.50966993801 s	Media
cotorra.jpg	45%	1024 x 768	1.38716774663 s	Media
flamenco.jpg	50%	1024 x 768	1.26501362152 s	Baja

Para esta prueba también se tomo una colección de imágenes, las cuales presentaban niveles de ruido del 0% al 50% y se aplicó para cada una de estas el algoritmo gaussiano con umbral 0(más bajo) y umbral 4 (más alto), se puede apreciar que aproximadamente hasta con un 35% la imagen resultante sigue teniendo buena calidad, lo que evidencia la eficiencia del algoritmo.



Fig. 34: Imagen con ruido gaussiano al 5%, b) Imagen con filtro gaussiano y umbral = 0, c) Imagen con filtro gaussiano y umbral = 4

Fig. 35: Imagen con ruido gaussiano al 50%, b) Imagen con filtro gaussiano y umbral = 0, c) Imagen con filtro gaussiano y umbral 4

3.4 Tipos y formatos de imágenes soportados

Cada imagen tiene sus propias características lo que la hacen diferente del resto, ejemplo de esto son los tipos y formatos que presentan cada una de estas, trayendo consigo que estas particularidades tengan que ser tomadas en cuenta a la hora de trabajar con las mismas.

Las pruebas realizadas anteriormente también permitieron observar cuales son los tipos y formatos permitidos para los algoritmos propuestos de reducción de ruido en las imágenes.

Entre los tipos permitidos tenemos:

- Imagen binaria: 1bit por píxel, blanco y negro.
- Imagen en escala de grises: 8 bit por píxel, blanco y negro.
- Imagen a color: 8 bit por píxel, usando una paleta de 256 colores.
- Imagen RGB: 3 bytes por píxel, color verdadero.

  Imagen RGBA: 4 bytes por píxel, color verdadero con una banda de transparencia alpha.

Entre los principales formatos permitidos tenemos:

- JPEG
- JPG
- PNG

- BMP
- TIFF

Siendo este último el formato más utilizado por los sistemas de prensa ya que es capaz de conservar las características de la imagen a través del tiempo. El formato GIF no es soportado por los algoritmos.

Conclusiones

Tras realizar las pruebas para cada uno de los algoritmos de reducción de ruido seleccionados y verificar que realmente son los más eficientes, se llevo a cabo la integración de los mismos con el producto Pyxel, lo cual será de gran ayuda para la gestión de las imágenes que se almacenan en los medios de prensa cubanos.

CONCLUSIONES

Con la culminación del presente trabajo de diploma los autores llegaron a las siguientes conclusiones:

- La realización de un estudio acerca de los aspectos teóricos que rigen la investigación, demostró la inminente necesidad de desarrollar algoritmos capaces de reducir el ruido presente en las imágenes digitales que son utilizadas por los medios de prensa cubanos.
- Ante la necesidad de desarrollar dichos algoritmos fue necesario llevar a cabo una investigación detallada acerca de las técnicas y herramientas más usadas en la actualidad para reducir el ruido en las imágenes.
- Se valoraron los algoritmos más utilizados actualmente para lograr la reducción de ruido en las imágenes y teniendo en cuenta las particularidades de cada uno de estos se realizó la implementación de los mismos, notando la superioridad de algunos y las deficiencias de otros, como es el caso de el filtro de la mediana y el filtro de punto medio del entorno de vecindad respectivamente.
- Se compararon dichos algoritmos en cuanto a tiempo de procesamiento y recursos consumidos y como resultado de esta comparación se seleccionaron los tres mejores para la integración al producto Pyxel, siendo estos el filtro de la mediana, el filtro gaussiano y el filtro binomial.
- Se demostró a través de pruebas documentadas que es eficiente la integración de los algoritmos desarrollados al producto Pyxel, ya que estos mejoraran en gran medida la calidad de las imágenes usadas por los medios de prensa cubanos, de una forma eficiente y rápida.
- Al término de esta investigación se encuentran integrados estos algoritmos al producto Pyxel.

RECOMENDACIONES

Con el fin de lograr una mejor gestión de las imágenes digitales en los medios de prensa se recomienda:

- Continuar investigando acerca de los algoritmos utilizados para la reducción de ruido en las imágenes digitales.
- Implementar estos algoritmos en otros lenguajes de programación con el fin de llevar a cabo comparaciones en cuanto a tiempo de procesamiento y recursos consumidos.

BIBLIOGRAFÍA REFERENCIADA

[1] A. Nivón Ruiz, “RGB: ¿Cómo se hacen las imágenes digitales?” Disponible en: http://laberintos.itam.mx/files/307.pdf.

[2] H. Rodríguez, “La imagen digital y el código binario” Disponible en: http://www.mailxmail.com/curso-iniciacion-imagen-digital/imagen-digital-codigo-binario.

[3] U.M. Hernández, “Reducción de ruido en una imagen digital” Disponible en: http://isa.umh.es/asignaturas/crss/temasvision/t05.pdf.

[4] S. Blanco Cuaresma, “Metodologías de desarrollo” Disponible en: http://www.marblestation.com/?p=644.

[5] X. Albaladejo, “Qué es SCRUM | proyectos Ágiles” Disponible en: http://www.proyectosagiles.org/que-es-scrum.

[6] J.E. González Cornejo, “El Lenguaje de Modelado Unificado (UML)” Disponible en: http://www.docirs.cl/uml.htm.

[7] D. Mertz, “Distributing Computing Introduction to Python Remote Objects (Pyro)” Disponible en: http://gnosis.cx/publish/programming/dc2.pdf.

[8] J.D.A. Acosta Cortes, W. Huertas Díaz, y W.W. Díaz Santa, “Diagrama de Componentes” Disponible en: http://www.blogger.com/feeds/6056993770253793848/posts/default.

BIBLIOGRAFÍA CONSULTADA

[1] M.D.C. Casas Cardoso, M. Pérez Díaz, J.E. Paz Viera, y J.V. Lorenzo Ginori, "Alasbimn Journal - Parámetros objetivos para analizar la calidad de la imagen" Disponible en: http://www.alasbimnjournal.cl/alasbimn/index.php?option=com_content&task=view&id=351&Itemid=149

[2] "Algoritmos para la reducción de ruido en señales" Disponible en: http://catarina.udlap.mx/u_dl_a/tales/documentos/meie/rosas_o_mc/capitulo4.pdf.

[3] "Capítulo 3. Procesamiento de imágenes" Disponible en: http://catarina.udlap.mx/u_dl_a/tales/documentos/lis/ramos_r_m/capitulo3.pdf.

[4] L. Vaentín Gil, "Detección de rasgos en imágenes con ruido: Una aproximación con funciones LISA en procesos puntuales espaciales" Disponible en: http://www.tesisenxarxa.net/TDX-0928105-112252/index_cs.html.

[5] M. Vázquez Acosta, "Ecúmene Pyxel — Ayuda de Ecúmene Pyxel 1.0" Disponible en: http://10.33.20.251:3389/help/.

[6] "Funcionamiento del algoritmo SPIHT" Disponible en: http://coco.ccu.uniovi.es/immed/compresion/descripcion/spiht/algoritmo/algoritmo.htm.

[7] H. Rodríguez, "La reducción de ruido perfecta" Disponible en: http://hugorodriguez.com/articulos/ruido_01.htm.

[8] H.A. García Elías y J.F. Ramírez Cruz, "Método Basado en Redes Neuronales Wavelet para Eliminar Ruido en Espectros Estelares" Disponible en: http://www.lsi.us.es/redmidas/CEDI07/%5B11%5D.pdf.

[9] "Procesamiento de imágenes" Disponible en: http://webdiis.unizar.es/~jdtardos/cdocSPR/2c.Procesamiento.pdf.

[10] U. Miguel Hernández, "Reducción de ruido en una imágen digital" Disponible en: http://isa.umh.es/asignaturas/crss/temasvision/t05.pdf.

[11] "Reducción del ruido de la imagen y defectos JPEG" Disponible en: http://help.adobe.com/es_ES/Photoshop/10.0/help.html?content=WS22CE3EDE-EC58-40ee-9A34-4083DB365429.html.

[12] U. Jaén, "Reducción del ruido en una imagen digital." Disponible en: http://www4.ujaen.es/~satorres/practicas/practica2.pdf.

BIBLIOGRAFÍA CONSULTADA

[13] S. Herold García y M. Escobedo Nicot, “Segmentación de imágenes médicas aplicando Snakes” Disponible en: http://www.santiago.cu/cienciapc/numeros/2007/4/articulo2.htm.

[14] “Tratamiento de imágenes” Disponible en: http://tratamiento-imagenes.blogspot.com/2009_10_01_archive.html.

[15] A. Nivón Ruiz, “RGB: ¿Cómo se hacen las imágenes digitales?” Disponible en: http://laberintos.itam.mx/files/307.pdf.

[16] U.M. Hernández, “Reducción de ruido en una imagen digital” Disponible en: http://isa.umh.es/asignaturas/crss/temasvision/t05.pdf.

[17] H. Rodríguez, “La imagen digital y el código binario” Disponible en: http://www.mailxmail.com/curso-iniciacion-imagen-digital/imagen-digital-codigo-binario.

ANEXOS

Anexo 1: Imágenes con diferentes niveles de ruido aleatorio

5%

10%

15%

20%

25%

30%

35%

40%

45% **50%**

Anexo 2: Imágenes con filtro de la mediana aplicado

5%

10%

15%

20%

25%

30%

35%

40%

45%

50%

Anexo 3: Imágenes con diferentes niveles de ruido gaussiano

5%

10%

15%

20%

25%

30%

35%

40%

45%

50%

Anexo 4: Imágenes con filtro gaussiano con umbral=0 aplicado

5%

10%

15%

20%

25%

30%

35%

40%

45%

50%

Anexo 5: Imágenes con filtro gaussiano con umbral=4 aplicado

5%

10%

15%

20%

25%

30%

35%

40%

45% **50%**

Anexo 6: Imágenes con filtro binomial con máscara de 3x3 aplicada

5%

10%

15%

20%

25%

30%

35%

40%

45% **50%**

Anexo 7: Imágenes con filtro binomial con máscara de 5x5 aplicada

5%

10%

15%

20%

25%

30%

35%

40%

45%

50%