UNIVERSIDAD MICHOACANA DE SAN NICOLÁS DE HIDALGO

Facultad de Ciencias Físico Matemáticas

Mat. Luis Manuel Rivera Gutiérrez

Detección de bordes en imágenes empleando diferencias finitas generalizadas en mallas adaptativas

TESIS

Que para obtener el título de

Maestro en Ciencias en Ingeniería Física

PRESENTA

Fernando Obed Guillen Reyes

Asesor:

Dr. en Cs. Matemáticas Francisco Javier Domínguez Mota

Morelia Michoacán, Noviembre 2016

En Memoria de Paulina Guillen Reyes
1991-1993

Agradecimientos

Al pensar a quien agradecer este logro, me doy cuenta de que lo que he logrado hasta ahora es gracias a mi familia, a mis amigos y a mis profesores, es decir, es gracias a mis padres Fernando y Rosa, a mis hermanos Claudia, Misael, Paulina (q.e.p.d.) y Nahum, a mis abuelitos Rodolfo y María, a mis amigos y profesores que he logrado llegar hasta donde estoy, pues lo único que he hecho yo es estudiar y aprobar exámenes. Aunque hubo momentos difíciles, siempre estaré agradecido por lo que mis padres hicieron por mi, pues cuando se es muy joven, no se tiene la madurez suficiente para entender a veces el actuar de nuestros padres, pero cuando se alcanza cierta madurez, se va comprendiendo los motivos y la forma de educar de nuestros padres, estoy muy agradecido con mis padres, pues mee he dado cuenta que en general mis padres hicieron todo lo posible para que a mis hermanos y a mí no nos faltara nada.

Todos los momentos en que necesité apoyo, ahí estuvieron mis padres, hermanos, abuelos y mis mejores amigos, nunca olvidaré la hospitalidad y cario con la que me trataron mis abuelos cuando más necesitaba durante mis estudios de licenciatura, esos cafés en la maana platicando con mis abuelos, donde mi abuelo me contaba tantas historias, y donde mi abuelita siempre estaba al pendiente de mi bienestar, por lo ella ha sido para mi como una segunda madre.

Agradezco a mis hermanos, que me han dado tantas alegrías, en verdad estoy muy agradecido por todo lo que han hecho por mi, siempre los tengo presentes, los quiero mucho. Agradezco a mi mejor amigo Manuel Fuerte, pues siempre me apoyó en el aspecto social y emocional cuando enfrentaba problemáticas, orientándome en cosas en las que cometía errores, esas pláticas tan interesantes que siempre tenemos cuando nos reunimos, en las que comenzamos hablando de cómo nos ha ido y terminamos hablando de política, religión y del futuro de nuestra nación, he aprendido mucho de ti amigo Manuel, he aprendido mucho de la cultura de mi país, lo que verdaderamente significa ser moreliano, michoacano y mexicano, estoy muy agradecido contigo Manuel, de verdad que mis respetos, gracias por tu amistad. Y como no estar agradecido también con mis amigos Miguel, Paco e Ileana, que junto con Manuel siempre me regaaban cuando comenzaba a encerrarme y olvidar el aspecto social de mi vida, he aprendido muchas cosas de ustedes, gracias amigos por no dejar que me encierre totalmente en el trabajo, estoy muy agradecido con ustedes mis amigos, llevo

alrededor de once aos conociéndolos, siento alegría cada vez que recuerdo todo lo que vivimos en la preparatoria, y el hecho de que hasta la fecha nos sigamos reuniendo es la evidencia de nuestra solida amistad, estoy seguro que nuestra amistad es para toda la vida.

Estoy muy agradecido con mis profesores de la universidad, en especial con el Dr. Mota, pues gracias al Dr. Mota cada vez he adquirido más esperiencia y madurez matemática, he desarrollado habilidades que no sabia que tenía, pero además de aprender del Dr. Mota cómo funciona este negocio de las matemáticas, he aprendido tambieén mucha de nuestra cultura mexicana, le agradezco la paciencia que tuvo al trabajar conmigo, yo se que no fue fácil soportarme, sin embargo usted siempre tuvo la paciencia para ensearme tantas cosas, le agradezco que sin importar lo ocupado que estaba usted, siempre me prestaba tiempo para el apoyo académico y emocional, muchas gracias profe, en los logros posteriores que llegue a tener en la vida, siempre habrá una aportación suya.

Agradezco a Fernanda, Laura Patricia y al Dr. Edgar, gracias por toda su ayuda, gracias por ayudarme en mi salud y aprendizaje, gracias por mejorar mi calidad de vida. También quiero agradecer al amigo Carlos Alvarez que lamentablemente se nos adelanto en el camino, no solo fuiste un amigo para toda mi familia, tu y tu familia han sido importantes para mi y mi familia, gracias por tus consejos y por la amistad de tantos aos, aunque ya no estés entre nosotros, se que estás en un mejor lugar, nunca me parecerá justo que se te hayas tenido que ir de este mundo tan pronto, pero te recordamos con cario.

Finalmente agradezco a mis profesores e investigadores de mi querida universidad con los que he tenido la oportunidad de trabajar, gracias Dra. Karina, Dra. Lourdes, Dr. Tinoco, Dr. Jaime Saavedra, Dr. Garduo, Dr. David Meza, Dr. Manuel, Dr. Rigo, Dr. Sepúlveda, Dr. Hector, Dr. Tejeda, Dr. Hector y al Dr. Mario César, además de agradecerle a Marce por toda su orientación durante mi estancia en la maestría, sinceramente no sé que haríamos en el posgrado sin ti Marce. También les agradezco a mis compaeros de generación de la mestría Migue, Rica, Berna, Cindy, Hugo, David, Elijio y Arturo por el apoyo y por hacer mas divertido el tiempo durante la maestría, muchas gracias también a Erika y Joss por su ayuda y amistad, se que les espera un futuro feliz juntos. Estoy muy orgulloso de ser moreliano, michoacano y mexicano, pero además soy orgullosamente Nicolaita, gracias a mi querida Universidad Michoacana de San Nicolás de Hidalgo.

Resumen

En la presente tesis se tuvo como propósito abordar el problema de detección de bordes, el cual es un problema del procesamiento de imágenes digitales. Principalmente se analizaron dos formas de llevar a cabo la detección de bordes, las cuales son el uso de diferencias finitas en mallas adaptativas y un método propuesto durante el presente trabajo, el cual denominamos algoritmo Fuerte. En el caso de mallas adaptativas se empleó el funcional armónico de suavidad adaptativo. Dado que el principal desafio en la generación de mallas adaptativas es la solución de un problema de optimización a gran escala, entonces se empleó el método LBFGS. Las diferencias finitas, en el caso de mallas adaptativas, son usadas durante la discretización del funcional utilizado. Para la generación de mallas adaptativas se investigaron distintos caminos, tales como la generación de mallas adaptativas usando el método LBFGS y el algoritmo Fuerte en conjunto con el método LBFGS, los resultados obtenidos con estas metodologías son bastante interesantes, pues fueron obtenidas características buscadas en las mallas, como la calidad en detección en los bordes y la convexidad de la malla. Por otra parte, el algoritmo Fuerte surgió durante la elaboración de esta tesis, pues en un intento de reducir la cantidad de datos al generar mallas adaptativas, se plantearon ideas que simplificaban la elección de los datos de la imagen a utilizar, el algoritmo Fuerte tiene como única inspiración la naturaleza local de la función de intensidad de una imagen, así como una clasificación de regiones en dicha función, por tanto las ideas y la formulación del algoritmo Fuerte son batante simples, el agoritmo Fuerte no es costoso computacionalmente y tiene buena calidad en sus resultados. Es importante mencionar que desde la creación del algoritmo Fuerte hasta el final de la redacción de esta tesis, no se encontró ningún otro algoritmo que ya empleara las ideas del algoritmo Fuerte. Otro camino empleado para la detección de bordes fue el filtro de Perona-Malik usando diferencias finitas generalizadas en mallas adaptativas. En la presente tesis se abordaron las nociones básicas de imágenes digitales, la descripción basica de algunos filtros relevantes en la detección de bordes, la teoría general, variacional y adaptativa de la generación de mallas, la formulación del algoritmo Fuerte, el planteamiento de las diferencias finitas genealizadas y los resultados obtenidos con las metodologías mencionadas. Finalmente, presentamos la elaboración de dos implementaciónes, una hecha en el ambiente GUI de Matlab y una implementación propia desarrollada en Java, en las cuales se conjuntan las implementaciones de la teoría para llevar a cabo detección de bordes en image-

nes digitales.

Palabras Clave : Detección de bordes en imágenes digitales, Mallas adaptativas, Diferencias finitas generalizadas, Algoritmo Fuerte, Método LBFGS, Problema de optimización a gran escala.

Abstract

In this thesis the main purpouse was to research the edge detection in digital images, an image processing problem. We mainly analysed two ways to achieve edge detection, first using finite differences in adaptative grids, and second an algorithm that we developed and that will be refer to as strong algorithm. In the case of adaptative grids, an adaptative harmonic functional was used. Since the main problem of generation of adaptative grids is to solve a large scale optimization problem, we used the LBFGS method. The finite differences are used in adaptative grids when the mentioned functional is discretized. To generate adaptative grids, we used different ways to achieve it, using the LBFGS method, and using strong algorithm and LBFGS together, the results obtained are of interest, because we get features as robust edge detection and convexity. On the other hand, we proposed strong algorithm during the elaboration of this work, because in an attempt to reduce data usage in adaptative grid generation, we developed some ideas that simplify wich data to use from the digital image, strong algorithm is based in the local nature of intensity function, as well as a classification of regions in the intensity function, then the ideas that compose strong algorithm are simple, this implies that strong algorithm is not expensive and the results are interesting. It is important to point out that, up to our knowledge, there is not an algorithm to the strong one. Another way that we reseached for edge detection was to use finite difference in adaptative grids in the filter of Perona-Malik. In this thesis we present the basic theory of digital images, a basic description of some famous filters used for edge detection, the theory of general, variational and adaptative grid generation, the formulation of Strong algorithm, the theory of generalized finite differences and the results obtained with the different methodologies mentioned. Finally, in this work we present two implementations, the first implementation was made in the GUI enviroment of Matlab and a the anothe is a own implementation developed in Java, this interfaces group all the theory mentioned to achieve edge detection.

Key words : Edge detection in digital images, Adaptative grids, Generalized finite differences, Strong Algorithm, LBFGS method, Large scale optmization problem.

Índice general

Índice de figuras

Capítulo 1

Imágenes digitales

1.1. Introducción

En este capítulo se describen los conceptos de procesamiento de imágenes que se emplearán en el presente trabajo. Es necesario enfatizar que solamente se presentan los conceptos de imágenes digitales y filtros involucrados en el problema de detección de bordes. De aquí en adelante, cuando se haga referencia a una imagen, estará implícito que se trata de una imagen digital.

Los filtros descritos en este capítulo son: el filtro de Perona-Malik [11], el filtro de Canny [1] y el filtro de Sobel [12], los cuales son algunos de los los filtros más usados para le detección de bordes.

1.1.1. Procesamiento de imágenes

El procesamiento de imágenes es un campo de estudio de las ciencias de la computación que busca obtener información o modificar una imagen a partir de una imagen inicial. Al aplicar tales procesos a una imagen se modifica alguna característica de la imagen; en la figura 1.1 se muestra esta idea.

Figura 1.1: Diagrama que describe el procesamiento de imágenes.

Existen diversos procesos que pueden ser aplicados a una imagen, pero el tipo de proceso empleado depende directamente de los propósitos que se tengan para la nueva imagen, es decir, si se quiere modificar una característica específica de una imagen, entonces se debe elegir un proceso particular que sea capaz de modificar de la mejor manera la característica deseada. Existen distintas modificaciones que con frecuencia se hacen a una imagen, por ejemplo:

- Suavizamiento.
- Eliminación de ruido.
- Segmentación.
- Detección de bordes.

Algunos de los procesos que son empleados para lograr las modificaciones anteriores en una imagen son los llamados filtros, los cuales consisten en la misma idea del diagrama de la figura 1.1. En el siglo XX, con el surgimiento de las computadoras y las nuevas tecnologías, surgieron entre otras cosas, las imágenes digitales, las cuales al pasar el tiempo han evolucionado considerablemente, pues hoy en día las imágenes son empleadas en diferentes áreas de la ciencia. Los posibles usos de un imagen van desde aplicaciones al campo de la salud para realizar diagnósticos, investigación científica, mapas geográficos más precisos y lamentablemente en tecnología bélica.

La detección de bordes de una imagen simplifica el manejo de la misma, pues se reduce significativamente la cantidad de información (datos) que representa a una imagen, es por esto que cualquier mejora en la detección de bordes tiene una repercusión directa en la tecnología.

1.1.2. Descripción computacional de una imagen digital

En la ciencia, una de las principales herramientas usadas en toda su historia ha sido la observación, pues por la capacidad de la observación humana han surgido las interrogantes fundamentales que han llevado a la humanidad hasta donde está. Pero desafortunadamente, la capacidad de observación empleando los sentidos es temporal, y es aquí que las imágenes (entre otras herramientas) entran en juego para el desarrollo científico. Las imágenes son un medio que permite analizar y almacenar gran cantidad de información, con la cual se pueden detectar aspectos que pasarían desapersibidos a simple vista.

Existe una idea intuitiva del concepto de imagen, esta idea generalmente consiste en entender o asociar el concepto de imagen con una fotografía, aunque no del todo incorrecta, esta manera de entender una imagen es limitada, pues una imagen digital es un concepto más general, por lo que es necesario deshacerse

esta asoaciación.

Es importante mencionar que una imagen no está restringida a dos dimensiones, una imagen puede ser tridimensional. Por esto surge un aspecto importante, el cual consiste en como visualizar una imagen. La respuesta computacional es entender el concepto de imagen como una estructura datos, mientras que la respuesta matemática es visualizarla como una función.

Estas maneras de visualizar imágenes se basan en la clase de representación de imágenes [6], pues dado que una imagen contiene información, esta debe estar organizada de alguna forma, este tipo de organización es lo que se llama **representación de una imagen**, el cual es un concepto importante. Para entender en que consiste una imagen, se parte de la clase de representación, la forma más simple y práctica para la representación de una imagen es por medio de la llamada representación espacial. La representación espacial consiste en representar una imagen usando variables espaciales para darle posición a los datos que componen la imagen, en el caso bidimensional los datos que componen la imagen tienen un arreglo matricial.

Bajo la representación espacial, las imágenes se representan por medio de elementos básicos, los cuales en el caso bidimensional son llamados píxeles, y en el caso tridimensional se usa una generalización del píxel llamado vóxel. Una imagen tridimensional consiste en una extensión de la representacion matricial, (ver figuras 1.2 y 1.3).

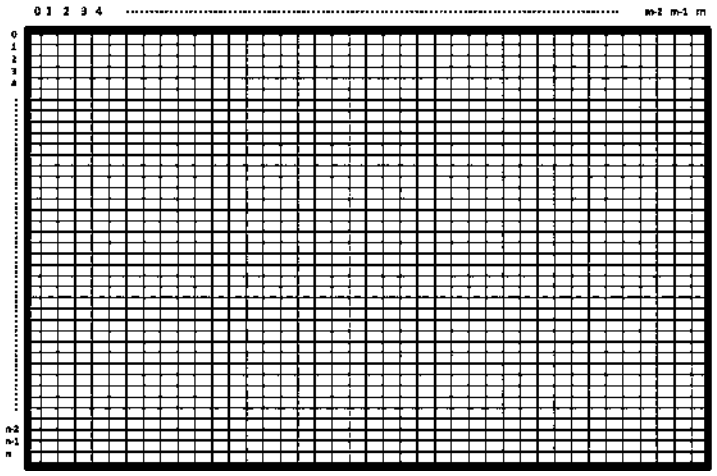

Figura 1.2: Estructura de una imagen en 2D bajo la representación espacial.

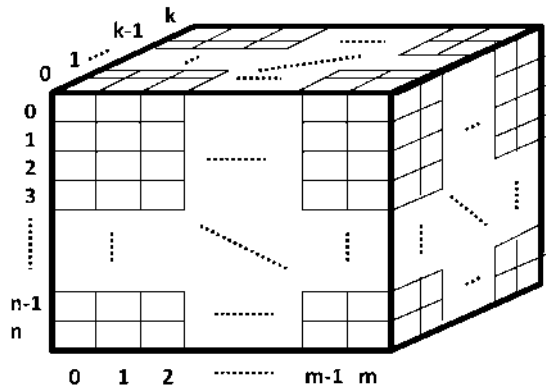

Figura 1.3: Estructura de una imagen en 3D bajo la representación espacial.

Por tanto, como una imagen bajo la representación espacial está compuesta por un arreglo de píxeles (o vóxels en el caso 3D), entonces sólo falta definir el concepto de resolución de una imagen. Por simplicidad, pero sin perdida de generalidad, solamente se hará uso de imágenes en dos dimensiones.

La resolución de una imagen se define como el número de píxeles la compone, dado que se está bajo la la representación espacial y aclarando que sólo se usan píxeles de tipo rectangular (pues existen otras geometrías para los píxeles, se muestra en la 1.4), entonces en este caso las imágenes tienen estructura rectangular, y por tanto si la imagen tiene n píxeles horizontalmente y m píxeles verticalmente, entonces la resolución de la imagen será de $n \times m$, pues se tienen nm píxeles en la imagen.

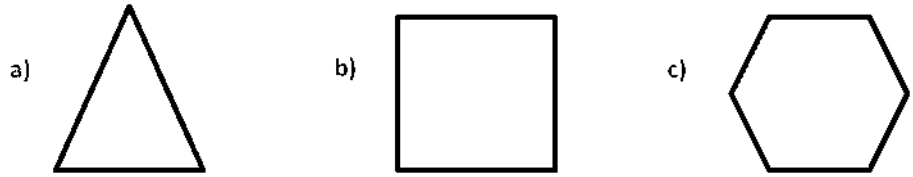

Figura 1.4: Tipos de geometrias de píxeles.

Finalmente, dado que en este trabajo se restringió a imágenes de dos dimensiones, se concluye que una imagen vista desde la perspectiva computacional se debe visualizar como un arreglo matricial de datos, donde cada entrada es el valor de color de un píxel en la escala de grises.

1.1.3. Función de intensidad de una imagen

Bajo la representación espacial, también es posible entender a una imagen desde el punto de vista matemático, es decir, se puede entender una imagen como una función I tal que $I : \Omega \subset \mathbb{R}^2 \to \mathbb{R}$, donde la función tiene la forma $I(x, y)$ y (x, y) denota la posición del un píxel.

Figura 1.5: Imagen de Lenna Gray, clásica en el procesamiento de imágenes.

A la función I se le conoce como **función de intensidad** y es propia de cada imagen, pues la función de intensidad asocia a la posición de cada píxel de la imagen con un valor numérico que representa el color del píxel.

En la presente tesis se utilizó la escala de grises en las imágenes para construir las funciones de intensidad. La función de intensidad fue fundamental en el presente trabajo, pues dada una imagen, se obtuvo su respectiva función de intensidad I, la cual nos muestra características importantes de la información del color de la imagen, en otras palabras, la función de intensidad es una forma de modelación matemática del color de una imagen, por lo que se pueden utilizar conceptos matemáticos para analizar una imagen.

La figura 1.5 nos muestra la imagen de Lenna Gray, esta misma imagen pero en escala de grises se muestra en la figura 1.6. Para dicha imagen en escala de grises, las figuras 1.7, 1.8 y 1.9 muestran distintas perspectivas de la función de intensidad de la imagen 1.6.

Figura 1.6: Imagen de Lenna Gray en escala de grises.

La función de intensidad de una imagen es de gran utilidad, pues a simple vista se puede ver la variación de color en la imagen. Esto último ha tenido muchas aplicaciones, en los próximos capítulos se muestra cómo se usó la función de intensidad en los propósitos de este trabajo. Es importante sealar que se puede analizar matemáticamente la función de intensidad usando algunos otros conceptos matemáticos, como los son las curvas de nivel, en las figuras 1.10 y 1.11 se muestra un ejemplo de distintas perspectivas de curvas de nivel de la función de intensidad de la imagen 1.6.

La función de intensidad sirve de inspiración para el surgimiento de nuevos métodos, pues la idea intuitiva principal que surge al observar su gráfica para detectar los bordes en una imagen es que los mismos están localizados en las discontinuidades más notables de la función de intensidad.

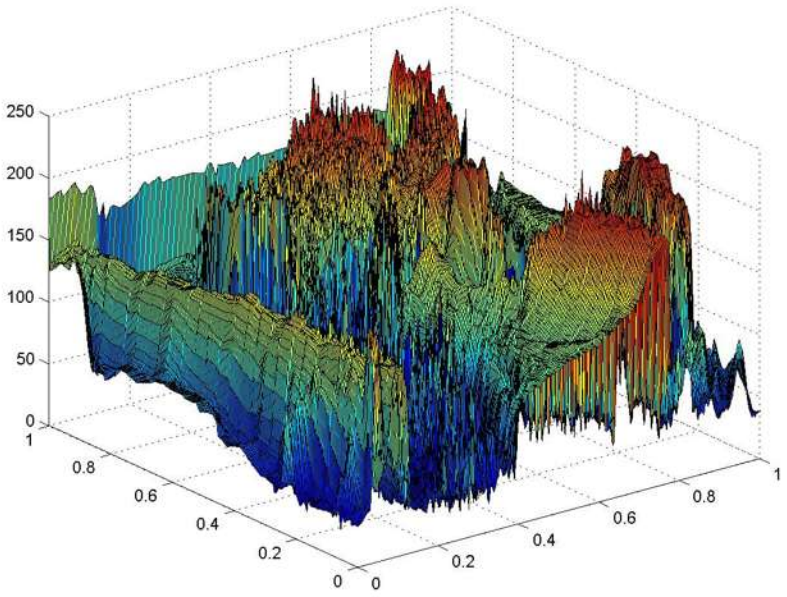

Figura 1.7: Función de intensidad de la imagen de Lenna Gray.

1.2. Filtro de Perona-Malik

1.2.1. Planteamiento

Existen diversas maneras de detectar bordes en una imagen, en la presente tesis se usó un método basado en una EDP, en el cual se emplearon diferencias finitas generalizadas. Este método es el Filtro de Perona-Malik, el cual está basado en la ecuación diferencial de difusión anisotrópica. Este método fue desarrollado por Pietro Perona y Jitendra Malik en 1987 [11], la aplicación de las diferencias finitas se da en una región Ω, en la cual se va a aproximar la solución de la EDP, por lo que la región Ω es el dominio rectangular de la función de intensidad de la imagen.

$$u_t = \nabla \cdot (c(x, y, t)\nabla u(x, y, t)). \tag{1.1}$$

La ecuación de difusión anisotrópica está dada por la ecuación diferencial parcial no lineal (1.1), donde la función $c(x, y, t)$ es llamada coeficiente de difusión. Si $c(x, y, t)$ es una función constante para toda $(x, y) \in \Omega$, entonces la ecuación 1.1 se reduce a la ecuación de difusión isotrópica.

Figura 1.8: Función de intensidad de la imagen de Lenna Gray vista desde otra perpectiva.

Las imágenes tienen distintos tipos de regiones, los bordes y el complemento de los bordes, en donde la función de intensidad es constante o casi constante, estas regiones serán llamadas regiones homogéneas o casi homogéneas de la imagen, pues la variación del color en este tipo de regiones es muy baja. Puesto que el objetivo de este modelo es lograr la detección de bordes y el suavizamiento en el resto de la imagen, entonces se busca una función $c(x, y)$ adecuada que tenga las siguientes características:

- La función $c(x, y, t)$ tenga un efecto nulo en los bordes.
- La funcion $c(x, y, t)$ tenga un efecto difusivo en el resto de las regiones de la imagen para lograr un suavizamiento y donde el efecto difusivo sea máximo en las regiones homogéneas.

Por tanto, teniendo en cuenta las características mencionadas, se busca una función $c(x, y, t)$ tal que:

- La función $c(x, y, t)$ tenga valor 0 en los bordes.
- La función $c(x, y, t)$ tenga valor 1 en el resto de las regiones de la imagen.

Dado que en principio no se sabe donde están los bordes en una imagen, entonces se necesita primero encontrar un tipo de medida E que sea usada como

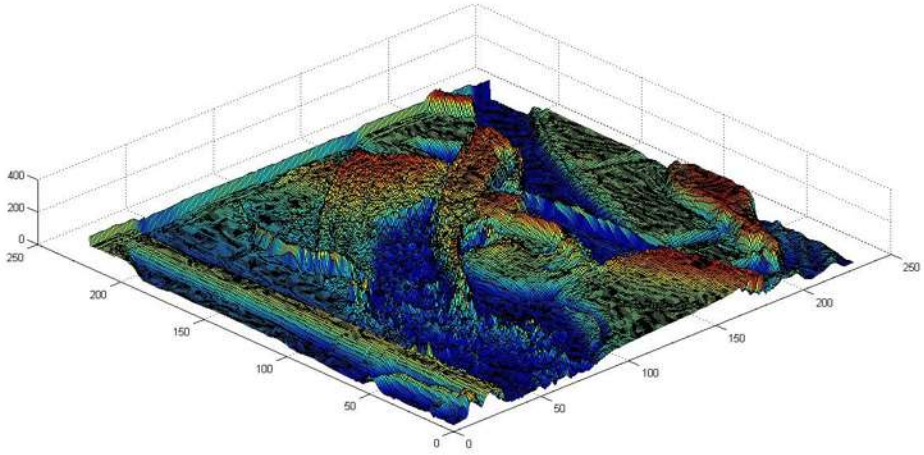

Figura 1.9: Una interpolación de la función de intensidad de la imagen de Lenna Gray en escala de grises.

parámetro para la función $c(x, y, t)$, es decir, una medida E para denotar la función de difusión tal que $c(E)$. Se necesita que la medida E esté en términos de las variables espaciales x e y, y que además ayude a saber si un píxel de la imagen forma parte de un borde o no, dichas ideas se examinan con más detalle en [3] y [15].

Observando la función de intensidad, se produce una idea intuitiva, esta idea consiste en que los bordes se encuentran en las discontinuidades más notables, donde se tienen variaciones locales mayores de color, es decir, esta idea intuitiva consiste en que los bordes de la imagen serán aquellos donde se tenga mayor variación de color, en otras palabras, un gradiente mayor. Es por esto que se propuso que $E = \| \nabla I \|$, esta propuesta tuvo buenos resultados en los primeros experimentos realizados por Perona y Malik.

Por tanto, si se establece $E = \| \nabla I \|$ y compactando las ideas establecidas y para que se cumplan las características buscadas de la función de difusión $c(x, y)$, se tiene que:

- Si en (x, y) se tiene que $\| \nabla I \| \to \infty$ entonces es más probable que se trate de un borde, y por tanto se debe cumplir que $c(\nabla I) \to 0$, , para así tener un efecto difusivo menor en (x, y), pues así se tendrá un efecto de conservación y por tanto detección de bordes al final del proceso.

- Si en (x, y) se tiene que $\| \nabla I \| \to 0$ entonces es muy probable que no se trate de un borde sino de un punto que pertenece a una región homogénea

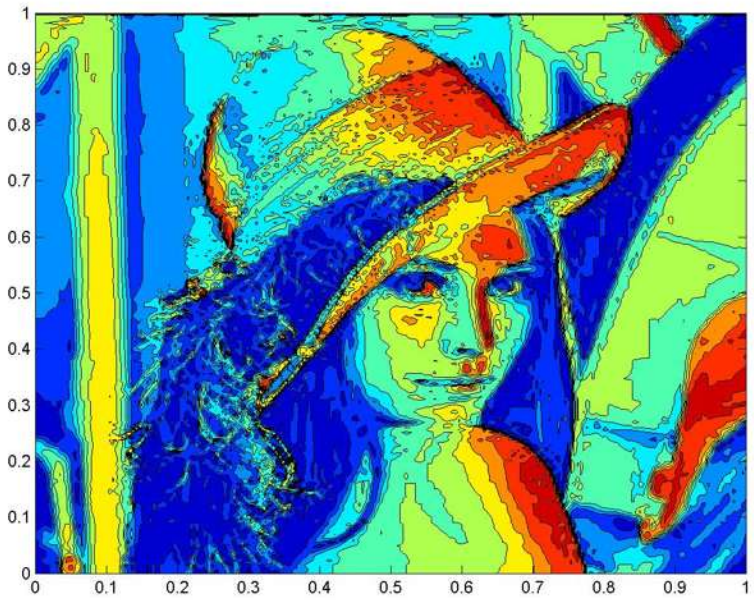

Figura 1.10: Curvas de nivel de la función 1.7.

o casi homogénea, y por tanto se debe cumplir que $c(\nabla I) \to 1$, para así tener un efecto difusivo mayor en (x, y) y por tanto un suavizamiento.

Puesto que se busca la función c que cumpla con las características anteriores, entonces esto implica que la forma de la funición c tenga una forma peculiar, la cual se muestra en la figura 1.12, pues las ultimas dos características mencionadas implican que si la función c esta en términos del gradiente de I, esta debe tener un comportamiento monótonamente decreciente. Esto ultimo restringe a la función c a un tipo más específico de función, Perona y Malik propusieron la función c dada por

$$c(\nabla I) = \frac{1}{1 + \frac{\|\nabla I\|^2}{\lambda^2}}. \tag{1.2}$$

Analizando esta expresión se puede ver que cumple con las características buscadas, teniendo un efecto difusivo mayor en regiones homogéneas o casi homogéneas de la imagen. La función de difusión dada por la expresión (1.2) logró muy buenos resultados en la detección de bordes y suavizamiento.

Es muy importante aclarar que la función u que aparece en la ecuación (1.1), no puede ser cualquier función en la aplicación a imágenes, pues para la detección de

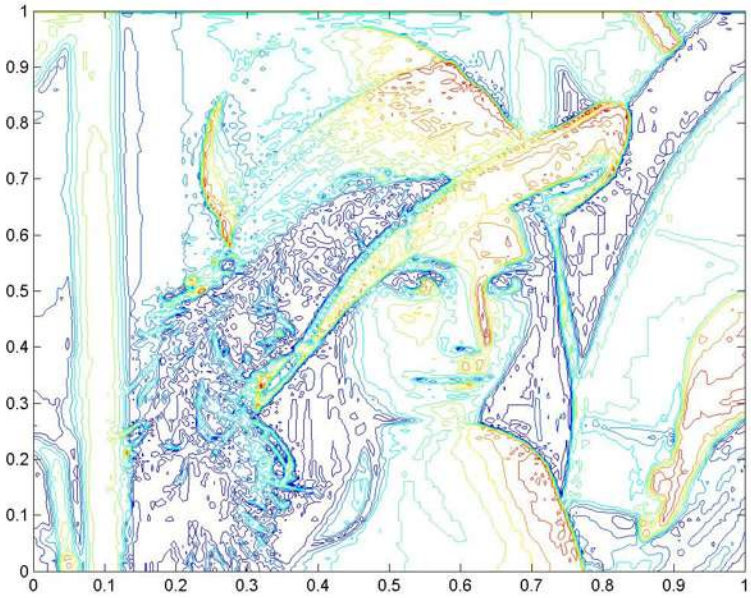

Figura 1.11: Otra perspectiva las curvas de nivel de la función 1.7.

bordes, la función u en la ecuación de difusión anisotrópica debe ser precisamente la función de intensidad de la imagen I. Por tanto, la ecuación de difusión anisotrópica, para la detección de bordes, está en términos de la función de intensidad I, por lo que el filtro de Perona-Malik está dado por la siguiente ecuación

$$I_t = \nabla \cdot (c(x, y, t)\nabla I), \tag{1.3}$$

donde el coeficiente de difusión $c(x, y)$ está dado por la expresión (1.2), nótese que ∇I tambien varía con el tiempo. Hay que observar que la función c está en términos de las variables espaciales x e y, pues ∇I está en términos de x e y.

1.3. Filtro de Canny

1.3.1. Planteamiento

El filtro de Canny fue diseñado para la detección de bordes en una imagen [1], es uno de los más utilizados. Los criterios en los que se basa son los siguientes:

- Criterio de detección: lograr que la probabilidad de obtener píxeles que no son parte de un borde sea muy baja.

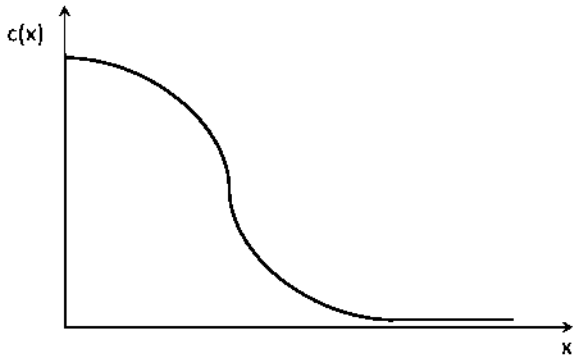

Figura 1.12: Forma general de la función $c(\nabla I)$ para cumplir las caraterísticas para la detección de bordes.

- Criterio de localización: los píxeles que sean determinados como parte de un borde deben de estar lo mas cerca posible del centro del borde real.
- Criterio de una sola respuesta: obtener sólo un posible borde por cada borde real.

En lo que respecta a la modelación de los criterios de detección y localización, en el caso de una dimensión se tiene lo siguiente: sea $f(x)$ la respuesta impulsiva del filtro, $n(x)$ el ruido y sea $G(x)$ una función que denota al borde. Sin perdida de generalidad se puede suponer que el borde está centrado en $x = 0$. La respuesta del filtro en el centro del borde denotado por H_G se obtiene mediante la integral de convolución siguiente

$$H_G = \int_{-W}^{+W} G(-x)f(x)dx,$$

donde se asume que el filtro tiene una respuesta impulsiva acotada por $[-W, W]$. La respuesta media cuadrática al ruido es

$$H_n = n_0\sqrt{\int_{-W}^{W} f^2(x)dx},$$

donde ${n_0}^2$ es la amplitud de ruido medio cuadrático por unidad de longitud.

Por tanto Canny uso la razón de salida de señal a ruido SNR, por sus siglas en inglés, como

$$SNR = \frac{\left|\int_{-W}^{+W} G(-x)f(x)dx\right|}{n_0\sqrt{\int_{-W}^{W} f^2(x)dx}}.$$

Para el criterio de localización se desea encontrar una cantidad o medida que incremente si la localización incrementa, Canny usó el recíproco de la distancia media cuadrática del posible borde al centro del verdadero borde. En el análisis unidimensional, Canny consideró que los posibles bordes estarían en un máximo local del operador $f(x)$. Al realizar una análisis unidimensional, Canny determinó que la localización L estaba dada por

$$L = \frac{H_G}{H_n} = \frac{|\int_{-W}^{+W} G(-x)f^{'}(x)dx|}{n_0\sqrt{\int_{-W}^{W} f^{'2}(x)dx}}.$$

Los criterios de detección y localización se ven reflejados matemáticamente por SRN y L,respectivamente, por lo que el problema se reduce a maximizar SNR y L simultaneamente. Para lograr una maximización simultánea se maximiza el producto de L y D, esto se hace empleando la desigualdad de Schwarz para integrales, pues SRN está acotada por

$$n_0^{-1}\sqrt{\int_{-W}^{W} G^2(x)dx},$$

y L está acotada por

$$n_0^{-1}\sqrt{\int_{-W}^{W} G^{'2}(x)dx},$$

por lo que el máximo del producto se alcanza si $f(x) = G(-x)$ en $[-W, W]$. Lo único que resta por hacer es abordar el tercer criterio para evitar multiples respuestas, Canny abordó este problema agregando algunas reestricciones al problema. La base del filtro de Canny es la aplicacíon de los modelos de los tres criterios.

1.4. Filtro de Sobel

1.4.1. Planteamiento

Este algoritmo fue utilizado en el sistema de visión del laboratorio de inteligencia artificial de la universidad de Stanford en 1968 [12] para la detección de un punto que pertenece a un borde. Desde entonces, el operador desarrollado por Sobel ha sido referenciado y usado en multiples trabajos de investigación hasta la actualidad.

Existe una descipción un poco más ámplia de la implementación de este operador en la literatura. La principal motivación de Sobel para desarrollar este operador fue que quería desarrollar un método para obtener de manera eficiente

un estimador del gradiente que fuera computacionalmente viable, a este operador se le llama operador gradiente isotrópico de 3×3 en imágenes.

El estimador de gradiente pensado por Sobel pretendía ser más isotrópico que el operador que se usaba en aquel momento, el cual era el operador de cruz de Robert. Para lograr su objetivo, Sobel ideó un operador que estimara el gradiente de una imagen digital en un punto mediante la suma de las cuatro posibles estimaciones del gradiente central en una vecindad de 3×3 de la imagen.

Sobel propuso que si la función de intensidad era muy uniforme, entonces como es de esperarse, las cuatro estimaciones del gradiente en la vecindad tendrían el mismo valor y en el caso de presentar una diferencia, era consecuencia de la naturaleza (no uniforme) local de la función de intensidad en la vecindad. Sobel tenía la intención de obtener la mejor dirección sin tanto rigor.

Es por esto que si se tiene una vecindad de 3×3, entonces el píxel central tiene ocho vecinos, como se muestra en la figura 1.13, entonces se define el modulo del vector de la derivada direccional g de esa vecindad tal que

$$|g| = \frac{n}{m},$$

donde n es la diferencia de densidad y m es la distancia a la vecindad.

a	b	c
d	e	f
g	h	i

Figura 1.13: Valores de la función de intensidad en una vecindad de 3×3.

La dirección de g está dada por un vector unitario apropiado para la vecindad, se puede observar que los distintos pares formados en la vecindad son: (a, i), (b, h), $(c,)g$ y (f, d). Por lo que se definió la estimación del gradiente G como

$$G = ([(c - g - a + i)/4 + (f - d)/2], [(c - g + a - i)/4 + (b - h)/2]).$$

Sobel consideró que si esta estimación del gradiente era métricamente correcta, entonces se debía dividir entre cuatro, para así obtener el gradiente promedio; además, para aumentar el nivel de precisión, por lo que decidió escalar a 4 el vector anterior y así evitar la perdida de precisión al realizar la división, por tanto, se definió $G^{'}$ tal que

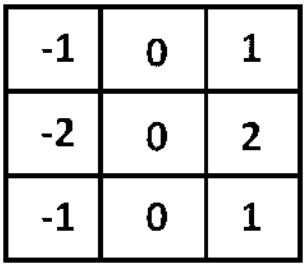

Figura 1.14: Pesos de la componente en x asignados a la vecindad 1.13.

$$G^{'} = 4G = ([(c - g - a + i) + 2(f - d)], [(c - g + a - i) + 2(b - h)]).$$

Para expresar la definición de la ultima expresión de una manera más practica, a la vecindad se le asignan los pesos mostrados en la figuras 1.14 y 1.15, de las componentes x e y, respectivamente.

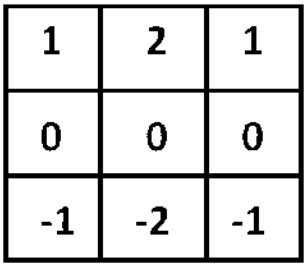

Figura 1.15: Pesos de la componente en y asignados a la vecindad 1.13.

Utilizando este algoritmo, se consideraba que un punto está en un borde si y sólo si

$$2|G^{'}| > T$$

,
donde T es un valor umbral dado.

Capítulo 2

Generación variacional de mallas adaptativas

2.1. Generación de mallas

2.1.1. Planteamiento general

La generación de mallas tiene como objetivo construir una partición por medio de un conjunto de puntos, llamados nodos de una región $\Omega \in \mathbb{R}^n$ en un número finito m de subregiones α_i tal que

- $\bigcup_{i=1}^{m} \alpha_i = \Omega$
- $\alpha_i \bigcap_{i \leq j} \alpha_j = \emptyset$.

la región Ω es llamada región física, la cual debe ser usualmente convexa, aunque en el caso de que no sea así, se divide dicha región en subregiones simplemente convexas y se hace la partición en cada subregión.

Se dice que una malla es estructurada si existe una estructura de orden sobre los nodos de la malla, es decir, una malla es estructurada si se sabe que nodos están en la frontera y cuales nodos son interiores, además de que cada clase de nodos tenga propiedades propias, como una estructura.

Por lo general, la aplicación principal de estas mallas es la solución de ecuaciones diferenciales, como la EPD usada en el filtro de Perona-Malik. Para aproximar la solución de una EDP empleando diferencias finitas, se necesita una malla para calcular las aproximaciones. La generación de mallas se vuelve un problema más complejo si la región Ω tiene una frontera muy irregular, afortunadamente en la presente tesis se trabaja en imágenes, por lo que las mallas tienen simpre una frontera en forma de rectángulo. La figura 2.1 muestra un ejemplo de una malla como las que se usaron para generar mallas adaptativas en imágenes, por

otra parte la figura 2.2 muestra una malla en una región con una frontera muy irregular.

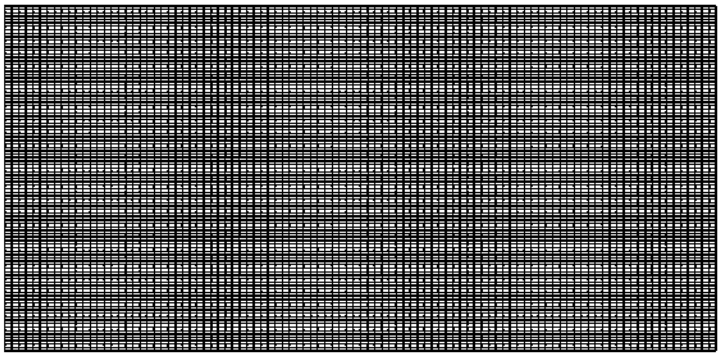

Figura 2.1: Malla rectangular inicial.

Por tanto, se plantea el problema de la generación de mallas de la siguiente manera: sea Ω la región física simplemente convexa y acotada donde se quiere generar una malla estructurada. Se propone utilizar una función $\bar{\mathbf{X}}$ (también llamada tranformación o mapeo) tal que $\bar{\mathbf{X}} : \mathbf{B} \to \Omega$, donde $\mathbf{B}$ es el cuadrado unitario. El cuadrado unitario es la región del plano definida por el producto cartesiano $[0, 1] \times [0, 1]$. El dominio de $\bar{\mathbf{X}}$, que en este caso es el cuadrado unitario, es llamado región lógica, la figura 2.3 ilustra la idea de un mapeo $\bar{\mathbf{X}}$ general.

Por tanto, en el contexto continuo, se dice que una malla sobre la región física Ω es la función continua (o mapeo) $\bar{\mathbf{X}}$ tal que

$$\bar{\mathbf{X}} : \mathbf{B} \to \Omega,$$

siendo $\mathbf{B}$ el cuadrado unitario, y donde

$$\bar{\mathbf{X}}(\xi, \eta) = (x(\xi, \eta), y(\xi, \eta)).$$

Para que el mapeo continuo $\bar{\mathbf{X}}$ sea una malla, se debe cumplir que:

- $\bar{\mathbf{X}}$ sea biyectivo.
- $\bar{\mathbf{X}}(\partial B) = \partial\Omega$.

Nótese que si se genera una cuadricula en B, entonces el mapeo debe generar una malla sobre Ω tal que las líneas en Ω no deben entrelazarse. Las condiciones anteriores establecen que la función continua $\bar{\mathbf{X}}$ será una malla si el es

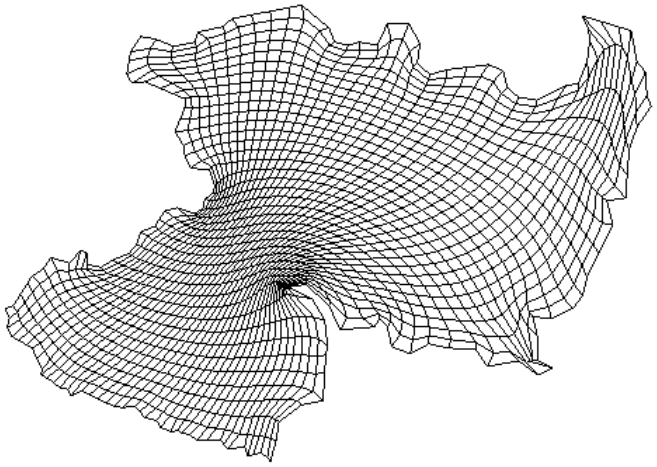

Figura 2.2: Malla irregular compuesta por cuadriláteros.

uno a uno y las fronteras corresponden entre sí, es decir, un punto de la frontera del cuadrado unitario bajo el mapeo $\bar{\mathbf{X}}$ corresponde a un punto frontera de Ω.

El que el mapeo $\bar{\mathbf{X}}$ sea continuo, en especial en la frontera del cuadrado unitario, garantiza que el orden que resulta al considerar el cuadrado unitario como la unión de cuatro segmentos sea también preservado en la región física.

Usualmente, surge la idea de dividir la frontera del $\mathbf{B}$ en cuatro segmentos, y así generar la malla por medio de la aplicación del mapeo a las líneas que unan las fronteras opuestas del cuadrado unitario, por la continuidad del mapeo $\bar{\mathbf{X}}$ las líneas en $\mathbf{B}$ generarán líneas continuas en Ω, que da como resultado una malla en la región fisica Ω. La ilustración de esta idea se muestra en la figura 2.4.

Es importante señalar que general las mallas están compuestas por triángulos o por cuadriláteros. Las figuras 2.5 y 2.2 muestran mallas compuestas por triángulos y cuadriláteros respectivamente.

Los elementos que definen la malla, ya sean triángulos o cuadriláteros, son llamados celdas. En el presente trabajo solamente se usaron mallas compuestas por cuadriláteros.

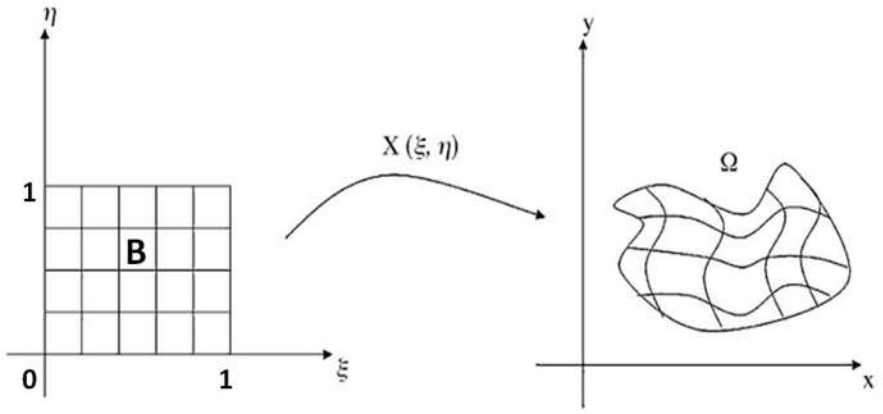

Figura 2.3: Mapeo $\bar{\mathbf{X}}$ definido del cuadrado unitario B a la región física Ω

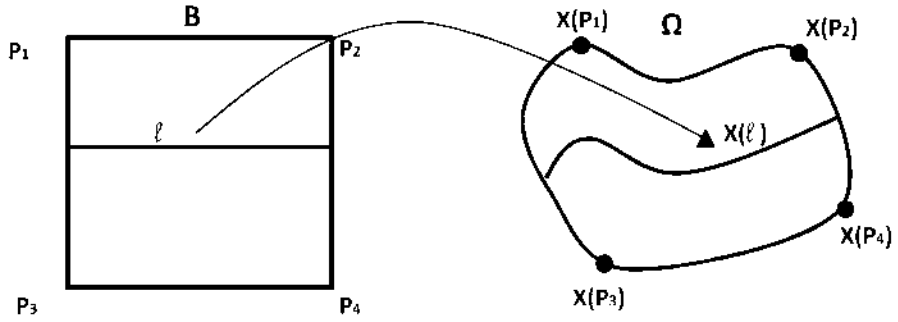

Figura 2.4: Frontera y líneas interiores bajo el mapeo

2.2. Generación variacional de mallas

2.2.1. Planteamiento

Para generar mallas en una región física Ω existen diversos métodos, sin embargo, en este trabajo se empleó el método variacional de generación de mallas (capítulo 2 de [2]), esto es, generar mallas utilizando ideas del cálcuo de variaciones, tales como los funcionales.

Puesto que las mallas tienen características geométricas específicas interesantes, se propusierón formas de optimizar dichas características geométricas por medio de la minimización un funcional, siendo así el surgimiento de la generación va-

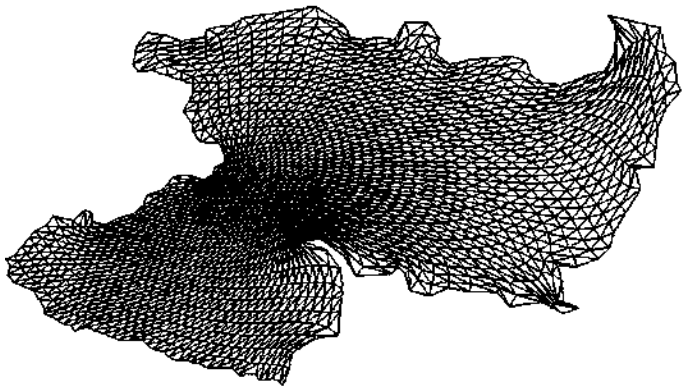

Figura 2.5: Malla compuesta por triángulos.

riacional de mallas. Las características geométricas que se quieren obtener por medio de el proceso de optimización pueden ser:

- La uniformidad de las áreas de las celdas de la malla.
- Ortogonalidad entre las líneas de la malla.
- Suavidad de las líneas de la malla.
- La uniformidad de la longitud de los lados de las celdas que componen la malla.

Pero además se pueden contruir mallas en donde se busque optimizar dos características simultaneamente, es decir, Si F_1 y F_2 son dos funcionales, en donde cada uno optimiza una característica de la malla, entonces es posible crear un funcional F^* tal que

$$F^* = \alpha_1 F_1 + \alpha_2 F_2,$$

donde $\alpha_1 + \alpha_2 = 1$ y F^* es un funcional que optimiza las carecterísticas de F_1 y F_2 en menor o mayor grado dependiendo de los valores de α_1 y α_2, pues si $\alpha_1 < \alpha_2$, entonces el funcional F_1 tendrá mayor peso en la optimización. Esto

permite crear mallas donde haya un balance entre dos características geométricas, por ejemplo, es posible generar mallas donde se combinen suavidad y área o suavidad y ortogonalidad simultaneamente.

El planteamiento del problema de generación de variacional de mallas es básicamente el mismo que el planteamiento general, pero a diferencia de la formulación de la sección anterior, el mapeo buscado en la generación variacional debe cumplir condiciones adicionales; no sólo se busca un mapeo que genere una malla convexa en la región física, sino que además optimice una propiedad geométrica de la malla (suavidad, ortogonalidad, área o longitud), por tanto de entre todos los mapeos continuos y biyectivos posibles del cuadrado unitario a la región física, se busca un mapeo que minimice un funcional de la forma

$$I[\bar{\mathbf{X}}] = \int_B f(\bar{\mathbf{X}}_\xi, \bar{\mathbf{X}}_\eta) d\xi d\eta$$

2.3. Funcional continuo de suavidad

2.3.1. Planteamiento

El funcional de suavidad en ocasiones es llamado **funcional armónico**, pues satisface la ecuación de Laplace, las ideas de esta sección pueden revisarse en el capítulo 2 de [4]. Los inicios de este funcional fue la formulación del sistema elíptico más simple que se puede formular a una región física, este problema propició las bases para el funcional de suavidad, el cual ha tenido cambios en el contexto continuo y discreto.

Sea Ω la región física y sea su frontera la unión de los cuatro segmentos l_i, l_d, l_s y l_{in} como se muestra en la figura 2.6.

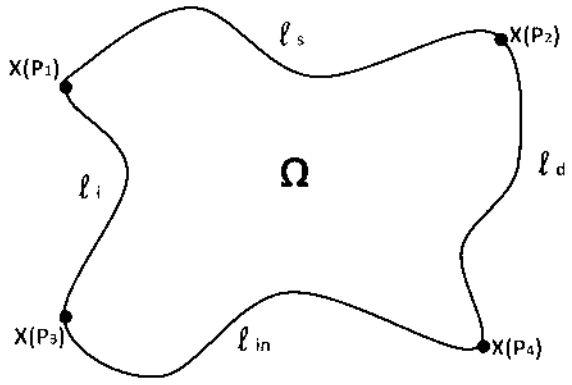

Figura 2.6: División de la frontera de la región física Ω en cuatro segmentos.

Se planteó la construcción de un mapeo directo $\tilde{\mathbf{X}}$ del cuadrado unitario B a Ω, como el que se ilustra en la figura 2.3, y donde además se buscaba que las componentes del mapeo 2.1 satisfacieran la ecuación de Laplace, es decir, si

$$\tilde{\mathbf{X}} = \tilde{\mathbf{X}}(\xi, \eta) = \begin{pmatrix} x(\xi, \eta) \\ y(\xi, \eta) \end{pmatrix}, \tag{2.1}$$

entonces

$$\tilde{\mathbf{X}}_{\xi\xi} = \begin{pmatrix} x_{\xi\xi}(\xi, \eta) \\ y_{\xi\xi}(\xi, \eta) \end{pmatrix}, \qquad \tilde{\mathbf{X}}_{\eta\eta} = \begin{pmatrix} x_{\eta\eta}(\xi, \eta) \\ y_{\eta\eta}(\xi, \eta) \end{pmatrix},$$

y por tanto se debía satisfacer que

$$x_{\xi\xi} + x_{\eta\eta} = 0 \tag{2.2}$$

$$y_{\xi\xi} + y_{\eta\eta} = 0. \tag{2.3}$$

Al establecer que las fronteras correspondan, se debían cumplir las siguientes expresiones

$$\tilde{\mathbf{X}}(\xi, 0) = \begin{pmatrix} x(\xi, 0) \\ y(\xi, 0) \end{pmatrix} = x_{in}(\xi) \tag{2.4}$$

$$\tilde{\mathbf{X}}(\xi, 1) = \begin{pmatrix} x(\xi, 1) \\ y(\xi, 1) \end{pmatrix} = x_s(\xi) \tag{2.5}$$

$$\tilde{\mathbf{X}}(0, \eta) = \begin{pmatrix} x(0, \eta) \\ y(0, \eta) \end{pmatrix} = x_i(\eta) \tag{2.6}$$

$$\tilde{\mathbf{X}}(1, \eta) = \begin{pmatrix} x(1, \eta) \\ y(1, \eta) \end{pmatrix} = x_d(\eta), \tag{2.7}$$

donde x_i, x_d, x_s y x_{in} son parametrizaciones para los segmentos l_i, l_d, l_s y l_{in} de $\partial\Omega$, respectivamente.

En este sistema elíptico se supone que $\tilde{\mathbf{X}}$ es suficientemente suave, por lo que existe una solución única infinitamente diferenciable al interior de Ω.

El problema, es que aunque el mapeo directo $\tilde{\mathbf{X}}$ sea suave en Ω, en regiones no convexas la malla resultante puede tener celdas no convexas. El mapeo (2.8) propuesto por Ivanenko muestra este problema.

$$\tilde{\mathbf{X}}(\xi, \eta) = \begin{pmatrix} \frac{1}{2}(\xi^2 - \eta^2) - \frac{2}{3}\xi \\ \xi\eta + \frac{1}{2}\xi - \frac{1}{2}\eta \end{pmatrix}, \tag{2.8}$$

en el cual las componentes satisfacen la ecuación de Laplace. Sin embargo, en la figura 2.7 se muestra como la malla resultante en una región se producen celdas no convexas. La causa de este problema es que si

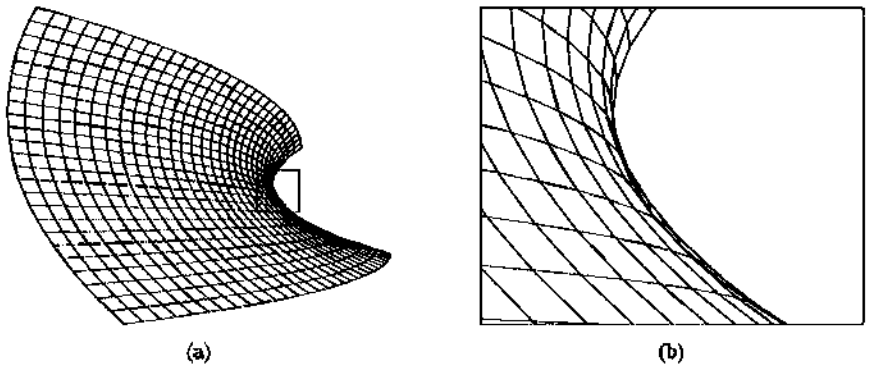

Figura 2.7: **a)** Malla resultante del mapeo propuesto por Ivanenko que cumple con las condiciones establecidas,**b)** se tienen celdas no convexas.

$$\eta = 0, \quad \frac{1}{3} < \xi < \frac{2}{3},$$

entonces el Jacobiano $J(\xi, \eta)$ es negatívo, pues se tiene que

$$J(\xi, \eta) = x_\xi y_\eta - x_\eta y_\xi = (\xi - \frac{2}{3})(\xi - \frac{1}{3}) + \eta(\eta + \frac{1}{2}).$$

Esto implica que algunas líneas de la malla en el cuadrado unitario $\mathbf{B}$, al ser mapeadas, produzcan líneas que se doblan. También se tendrá el mismo problema si se tiene una región física muy irregular, en particular, se generará este problema en donde haya picos en la frontera de la región física.

A pesar de este detalle, el mapeo de Ivanenko tiene la ventaja de ser muy simple y generar líneas suaves en la malla, pero este fue un hecho que motivó la incursión en el estudio de generación de mallas desde una perspectiva variacional, pues las ecuaciones de Laplace (2.2) y (2.3) son las ecuaciones de Euler-Lagrange del funcional

$$I[\tilde{\mathbf{X}}] = \int_0^1 \int_0^1 (x_\xi^2 + x_\eta^2 + y_\xi^2 + y_\eta^2) d\xi d\eta, \tag{2.9}$$

en donde las condiciones de frontera siguen siendo las parametrizaciones establecidas en 2.4, 2.5, 2.6 y 2.7.

Aunque la suavidad en las líneas de la malla es importante, también lo es la convexidad de las celdas que componen la malla. La convexidad de la malla está directamente relacionada con el Jacobiano del mapeo, por lo que Winslow hizo una versión un poco distinta del funcional de suavidad, dado por la siguiente expresión

$$I[\tilde{\mathbf{X}}] = \int_0^1 \int_0^1 \frac{(x_\xi^2 + x_\eta^2 + y_\xi^2 + y_\eta^2)}{x_\xi^t J x_\eta} d\xi d\eta, \tag{2.10}$$

donde J es el Jacobiano y esta dado por

$$J(\xi, \eta) = x_\xi y_\eta - x_\eta y_\xi.$$

2.4. Funcional Discreto de suavidad

2.4.1. Planteamiento

En esta sección se describe la discretización del funcional de suavidad de Winslow (2.10), partiendo del funcional continuo, pues es más práctico resolver un problema con el funcional discreto de suavidad, que resolver el sistema de ecuaciones diferenciales parciales de Euler-Lagrange resultantes de la minimización del funcional continuo.

Para plantear la discretización se hace uso de mapeos bilineales, además en esta sección se aborda la teoría involucrada con la convexidad de la malla (seccion 2,2 de [10]), pues como se ha mencionado antes, no sólo se busca suavidad en las mallas, sino que también se buscan mallas convexas, es decir, mallas con todas sus celdas convexas.

Por tanto, supongase que en la región física Ω se plantea generar la malla y sea P_Ω el conjunto de puntos en $\partial\Omega$ dado por

$$P_\Omega = \{p_1, p_2, p_3, p_4, ..., p_k\}, \tag{2.11}$$

donde el conjunto P_Ω forma el polígono que compone Ω; un ejemplo de esta idea se muestra en la figura 2.8.

Se puede definir a una malla G de dimensión $n \times m$ sobre la región Ω como el conjunto G tal que

$$G = \{p_{i,j} : 1 < i < n, 1 < j < m\}. \tag{2.12}$$

Nótese que al considerar una malla sobre Ω ya no sólo se consideran los nodos P_Ω que están en su frontera, sino que se están considerando nm nodos. En una malla de dimensión $n \times m$, $(2n + 2m - 4)$ nodos estarán en la frontera, y $(n-2)(m-2)$ estarán en el interior, por tanto es importante notar que $P_\Omega \subset G$ y que además $n, m \in \mathbb{N} \setminus \{1, 2\}$.

Entonces, es posible visualizar a una malla G como un conjunto estructurado de nodos (puntos en el plano), pues al manejar un orden en los nodos por medio

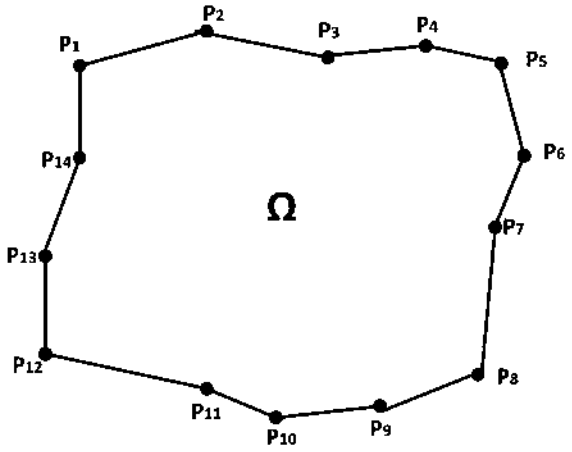

Figura 2.8: Poligono compuesto por P_Ω que define la región Ω.

de los subíndices i, j, como se hace en una matriz, se tiene una forma natural de estructurar los nodos. Se puede observar en la figura 2.9 un ejemplo de una malla estructurada; a los nodos que están sobre la frontera se les llama nodos frontera y a los nodos en el interior se les llama nodos interiores.

En la sección anterior se mencionó que se debe considerar la frontera de la regón física como la unión de cuatro segmentos, así como establecer una orientación en estos segmentos, los cuales se denotan como segmento izquierdo, segmento derecho, segmento inferior y segmento superior. Por tanto, dado el conjunto G que define la malla, se desprenden automáticamente los siguientes cuatro conjuntos que definen la orientación de la malla

$$G_{iz} = \{p_{i,j} \in G : i = 1, 2, 3, 4, ..., n, j = 1\} \tag{2.13}$$

$$G_d = \{p_{i,j} : i = 1, 2, 3, 4, ..., n, j = m\} \tag{2.14}$$

$$G_i = \{p_{i,j} : i = n, j = 1, 2, 3, 4, ..., m\} \tag{2.15}$$

$$G_s = \{p_{i,j} : i = n, j = 1, 2, 3, 4, ..., m\} \tag{2.16}$$

Es importante hacer algunas observaciones, cada nodo interior tiene ocho nodos vecinos, además en la malla habrá únicamente cuatro nodos esquina, los nodos esquina son $p_{1,1}$, $p_{1,m}$, $p_{n,1}$ y $p_{n,m}$ (ver figuras 2.9 y 2.10).

Lo comentado en esta sección sirve para describir lo que significa una malla estructurada en el contexto discreto, es decir, se debe entender una malla tanto

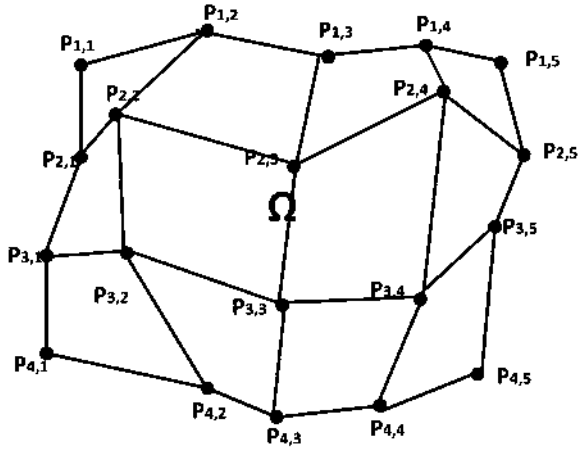

Figura 2.9: La estructura de la malla está definida por los índices de los nodos.

como una estructura de nodos y como un mapeo que cumple ciertas condiciones específicas. Estas ideas ayudan a poder visualizar de forma geométrica una malla lógicamente estructurada.

Pero no hay que olvidar que para generar una malla variacionalmente se debe encontrar un mapeo que satisfaga las condiciones mencionadas en la sección anterior. En la siguiente sección se abordan los mapeos bilineales, los cuales sirven para hacer la discretización del funcional de suavidad de Winslow.

2.4.2. Mapeos bilineales

Los mapeos bilineales son los mapeos más simples de formular, una descripción alternativa de este tipo de mapeos puede verse en la sección 3,3 de [2], pues hasta ahora sólo se han estudiado mapeos que van del cuadrado unitario a $\mathbf{B}$ a una región física Ω, pero a diferencia de esos mapeos, los mapeos bilineales van de $\mathbf{B}$ a un cuadrilátero arbitrario.

Para encontrar el mapeo adecuado de B a Ω se utilizan los mapeos bilineales, pues para encontrar el mapeo se construye una cuadrícula uniforme en B para formular mapeos bilineales, donde cada mapeo bilineal tiene como dominio una celda de la cuadrícula, en cierto sentido se construye el mapeo principal basándose en la filosofía de divide y venceras usando mapeos bilineales.

Puesto que la región más sencilla para construir una malla es un cuadrilátero, es por esto que surge el concepto de mapeo bilineal. A continuación se describe la formulación de un mapeo bilineal $\bar{\mathbf{x}}$ de $\mathbf{B}$ a un cuadrilátero arbitrario $\square PQRS$, es decir, $\bar{\mathbf{x}} : \mathbf{B} \longrightarrow \square PQRS$, la figura 2.11 ilustra el mapeo bilineal $\bar{\mathbf{x}}$.

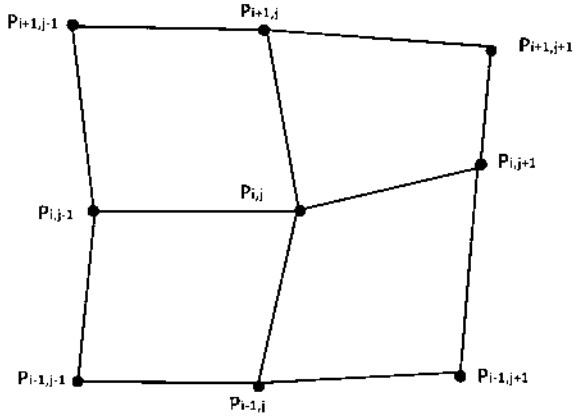

Figura 2.10: Nodos vecinos del nodo $\mathbf{P}_{i,j}$.

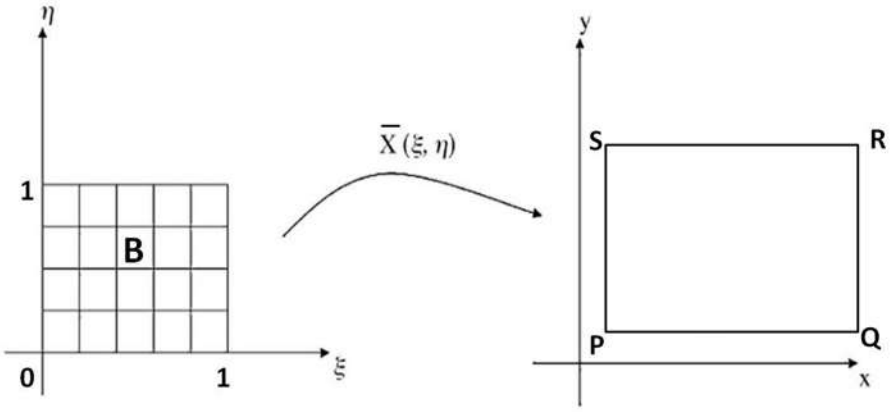

Figura 2.11: Mapeo bilineal $\bar{\mathbf{x}}$

El mapeo bilineal $\bar{\mathbf{x}}$ que se describe en esta sección tiene la forma

$$\bar{\mathbf{x}} = \alpha + \beta\xi + \gamma\eta + \delta\xi\eta,$$

donde

$$\bar{\mathbf{x}} = \begin{pmatrix} x(\xi,\eta) \\ y(\xi,\eta) \end{pmatrix}$$

$$\alpha = \begin{pmatrix} \alpha_1 \\ \alpha_2 \end{pmatrix} \quad \beta = \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix} \quad \gamma = \begin{pmatrix} \gamma_1 \\ \gamma_2 \end{pmatrix} \quad \delta = \begin{pmatrix} \delta_1 \\ \delta_2 \end{pmatrix}.$$

Para definirlo, se debe encontrar α, β, γ y δ, esto se hace usando la condición de que las fronteras deben corresponder, por tanto los vértices de $\mathbf{B}$ bajo el mapeo deben coicidir a los vértices de $\square PQRS$ en un orden dado.

Sin pérdida de generalidad, supongase que las esquinas corresponden de la siguiente manera (ver figura 2.11)

$$\bar{\mathbf{x}}(0,0) = P \quad \bar{\mathbf{x}}(1,0) = Q \quad \bar{\mathbf{x}}(1,1) = R \quad \bar{\mathbf{x}}(0,1) = S. \tag{2.17}$$

Entonces, evaluando $\bar{\mathbf{x}}$ en las esquinas de B y utilizando las ecuaciones anteriores, se tiene que

$$\begin{aligned}
\bar{\mathbf{x}}(0,0) &= \alpha + \beta(0) + \gamma(0) + \delta(0)(0) = \alpha \\
&\Rightarrow \alpha = P && (2.18)\\
\bar{\mathbf{x}}(1,0) &= \alpha + \beta(1) + \gamma(0) + \delta(1)(0) = \alpha + \beta \\
&\Rightarrow \alpha + \beta = Q \\
&\Rightarrow \beta = Q - \alpha \\
&\Rightarrow \beta = Q - P && (2.19)\\
\bar{\mathbf{x}}(0,1) &= \alpha + \beta(0) + \gamma(1) + \delta(0)(1) = \alpha + \gamma \\
&\Rightarrow \alpha + \gamma = S \\
&\Rightarrow \gamma = S - \alpha \\
&\Rightarrow \gamma = S - P && (2.20)\\
\bar{\mathbf{x}}(1,1) &= \alpha + \beta(1) + \gamma(1) + \delta(1)(1) = \alpha + \beta + \gamma + \delta \\
&\Rightarrow \alpha + \beta + \gamma + \delta = R \\
&\Rightarrow \delta = R - \alpha - \beta - \gamma \\
&\Rightarrow \gamma = R - P - S + P - Q + P = R + P - S - Q. && (2.21)
\end{aligned}$$

Por tanto, el mapeo bilineal buscado está definido por la la siguiente ecuación

$$\bar{\mathbf{x}} = P + (Q - P)\xi + (S - P)\eta + (R + P - S - Q)\xi\eta. \tag{2.22}$$

Se puede ver que efectivamente las fronteras corresponden, es decir, un punto en la frontera de $\mathbf{B}$ va a un punto en la frontera de $\square PQRS$, por ejemplo, sea m_0 un punto en la frontera de $\mathbf{B}$ tal que $m_0 = (\xi, 1)$ y donde $0 \leq \xi \leq 1$, entonces

$$\begin{aligned}
\bar{\mathbf{x}}(m_0) &= P + (Q - P)\xi + (S - P)(1) + (R + P - S - Q)\xi(1) \\
&\Rightarrow \bar{\mathbf{x}}(m_0) = P + (Q - P)\xi + S - P + (R + P - S - Q)\xi \\
&\Rightarrow \bar{\mathbf{x}}(m_0) = S + (Q - P + R + P - S - Q)\xi \\
&\Rightarrow \bar{\mathbf{x}}(m_0) = S + (R - S)\xi. && (2.23)
\end{aligned}$$

Se ve que efectivamente la frontera superior de B corresponde a el segmento de frontera entre S y R de $\square PQRS$, pues como $0 \leq \xi \leq 1$ y $(R - S)$ es el vector que va de S a R, análogamente se puede comprobar para las otras.

Existe una relación directa entre el Jacobiano y la convexidad de una celda de la malla, lo que nos permite saber si una celda es convexa o no. Si se denota al mapeo $\bar{\mathbf{x}}$ como

$$\bar{\mathbf{x}}(\xi, \eta) = \begin{pmatrix} x(\xi, \eta) \\ y(\xi, \eta) \end{pmatrix},$$

entonces es claro que

$$\bar{\mathbf{x}}_\xi(\xi, \eta) = \begin{pmatrix} x_\xi(\xi, \eta) \\ y_\xi(\xi, \eta) \end{pmatrix},$$

$$\bar{\mathbf{x}}_\eta(\xi, \eta) = \begin{pmatrix} x_\eta(\xi, \eta) \\ y_\eta(\xi, \eta) \end{pmatrix}.$$

El Jacobiano de $\bar{\mathbf{x}}$ está dado por la siguiente expresión

$$J(\xi, \eta) = x_\xi y_\eta - x_\eta y_\xi, \tag{2.24}$$

calculando las parciales $\bar{\mathbf{x}}_\xi$ y $\bar{\mathbf{x}}_\eta$ de (2.22), se tiene que

$$\bar{\mathbf{x}}_\xi = Q - P + (R + P - S - Q)\eta$$

$$\bar{\mathbf{x}}_\eta = S - P + (R + P - S - Q)\xi,$$

por tanto

$$x_\xi = Q_x - P_x + (R_x + P_x - S_x - Q_x)\eta,$$

$$x_\eta = S_x - P_x + (R_x + P_x - S_x - Q_x)\xi,$$

$$y_\xi = Q_y - P_y + (R_y + P_y - S_y - Q_y)\eta,$$

$$y_\eta = S_y - P_y + (R_y + P_y - S_y - Q_y)\xi,$$

sustituyendo las expresiones anteriores en (2.24), se tiene lo siguiente

$$\begin{aligned} J(\xi, \eta) = [Q_x - P_x + (R_x + P_x - S_x - Q_x)\eta][S_y - P_y + (R_y + P_y - S_y - Q_y)\xi] - \\ -[S_x - P_x + (R_x + P_x - S_x - Q_x)\xi][Q_y - P_y + (R_y + P_y - S_y - Q_y)\eta], \end{aligned}$$

expandiendo y agrupando términos se tiene la siguiente expresión del Jacobiano

$$\begin{aligned} J(\xi,\eta) = (Q_x - P_x)(S_y - P_y) &- (S_x - P_x)(Q_y - P_y) + \\ +[(Q_x - P_x)(R_y + P_y - S_y - Q_y) &- (Q_y - P_y)(R_x + P_x - S_x - Q_x)]\xi + \\ +[(S_y - P_y)(R_x + P_x - S_x - Q_x) &- (S_x - P_x)(R_y + P_y - S_y - Q_y)]\eta. \end{aligned} \tag{2.25}$$

Nótese que si se evalua el Jacobiano en una esquina de **B**, por ejemplo en la esquina $(0,0)$, esto implica que

$$J(0,0) = (Q_x - P_x)(S_y - P_y) - (S_x - P_x)(Q_y - P_y),$$

sustituyendo esta última expresión en (2.25) se obtiene la siguiente expresión del Jacobiano

$$\begin{aligned} J(\xi,\eta) = J(0,0) + & \\ +[(Q_x - P_x)(R_y + P_y - S_y - Q_y) - (Q_y - P_y)(R_x + P_x - S_x - Q_x)]\xi + & \\ +[(S_y - P_y)(R_x + P_x - S_x - Q_x) - (S_x - P_x)(R_y + P_y - S_y - Q_y)]\eta. & \end{aligned} \tag{2.26}$$

Evaluando $J(1,0)$ y $J(0,1)$, se obtiene lo siguiente

$$\begin{aligned} J(1,0) &= J(0,0) + [(Q_x - P_x)(R_y + P_y - S_y - Q_y) - (Q_y - P_y)(R_x + P_x - S_x - Q_x)] \\ &\Rightarrow [(Q_x - P_x)(R_y + P_y - S_y - Q_y) - (Q_y - P_y)(R_x + P_x - S_x - Q_x)] = \\ &= J(1,0) - J(0,0), \end{aligned}$$

$$\begin{aligned} J(0,1) &= J(0,0) + [(S_y - P_y)(R_x + P_x - S_x - Q_x) - (S_x - P_x)(R_y + P_y - S_y - Q_y)] \\ &\Rightarrow [(S_y - P_y)(R_x + P_x - S_x - Q_x) - (S_x - P_x)(R_y + P_y - S_y - Q_y)] = \\ &= J(0,1) - J(0,0), \end{aligned}$$

por lo que si también se sustituyen estas últimas dos expresiones en (2.26), entonces se reduce a la siguiente expresion

$$\Rightarrow J(\xi,\eta) = J(0,0) + [J(1,0) - J(0,0)]\xi + [J(0,1) - J(0,0)]\eta. \tag{2.27}$$

Finalmente, al evaluar $J(1,1)$ se tiene que

$$J(1,1) = J(1,0) + J(0,1) - J(0,0).$$

La importancia del trabajo algebraico anterior es que se encontró que el Jacobiano de un mapeo bilineal es lineal, además del hecho de que tal Jacobiano

está expresado en términos de $J(0,0)$, $J(1,0)$ y $J(0,1)$, los cuales son las evalucaiones del Jacobiano en esquinas de $\mathbf{B}$.

Respecto a las celdas de una malla G (en Ω), Ivanenko y Barrera propusieron trabajar con los triángulos que componen las celdas de la malla. Barrera propuso considerar a la malla como un conjunto de triángulos, estos triángulos son resultado de visualizar las dos diagonales de cada celda de la malla y que por tanto se tienen cuatro posibles triángulos por cada celda. Sin embargo, Barrera no sólo consideró a la malla como un conjunto de triángulos, sino aun más, consideró que los triángulos tienen una orientación.

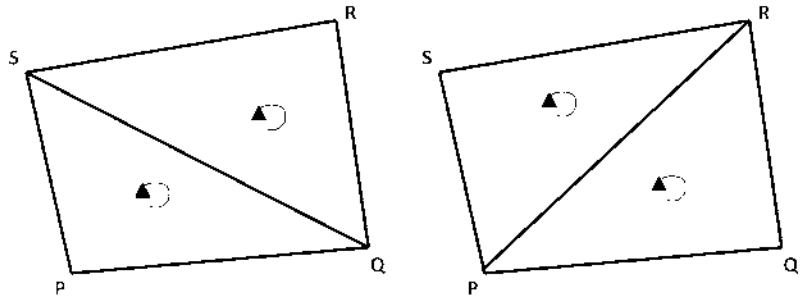

Figura 2.12: Los cuatro triángulos en los que se puede dividir una celda usando sus dos posibles diagonales

La figura 2.12, nos muestra la idea de Barrera [10], es decir, en una celda de la malla considerar los cuatro triángulos posibles que se obtienen al emplear las dos diagonales de la celda. El la misma figura se muestra también la orientación de los triángulos, la cual consiste en que un triángulo está orientado positivamente si sus vertices estan ordenados en el sentido contrario a las manecillas del reloj, y por tanto, si los vertices del triangulo están descritos de manera que tienen un orden en el sentido de las manecillas del reloj, entonces el triangulo está orintado negativamente.

En la figura 2.12 se puede ver los cuatro triángulos que se forman en una celda al considerar las dos posibles diagonales, tales triángulos son:

- $\triangle PQS$.
- $\triangle QRP$.
- $\triangle RSQ$.
- $\triangle SPR$.

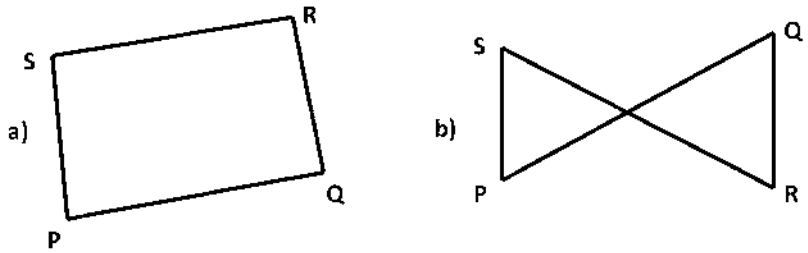

Figura 2.13: **a)**Celda convexa **b)**Celda no convexa

Se considera que un triángulo estará orientado positivamente si el orden en que se denotan los vertices están en sentido contrario a las manecillas del reloj, por ejemplo, el triángulo $\triangle PQS$ de la figura 2.12 está orientado positivamente, por otro lado, el triángulo $\triangle PSQ$ tendrá una orientación negativa.

Se define la función α tal que

$$\alpha(\triangle ABC) = \frac{1}{2} det(\vec{B} - \vec{A}, \vec{C} - \vec{A}),$$

donde la función α calcula el área del triángulo $\triangle ABC$.

La formula de área anterior se puede expresar de la siguiente manera

$$\alpha(\triangle ABC) = \frac{1}{2} \begin{vmatrix} x_b - x_a & x_c - x_a \\ y_b - y_a & y_c - y_a \end{vmatrix} = \frac{1}{2} (\vec{B} - \vec{A})^t J_2 (\vec{C} - \vec{A}), \tag{2.28}$$

donde

$$J_2 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}$$

Nótese que (2.28) tiene implícita la orientación establecida, es decir, si un triángulo $\triangle ABC$ está orientado positivamente, entonces $\alpha(\triangle ABC) > 0$, en cambio si está orientado negativamente, entonces $\alpha(\triangle ABC) < 0$.

Ivanenko propuso una idea muy ingeniosa para saber si una celda es o no convexa, pues si los cuatro triángulos de una celda están orientados positivamente, entonces bajo la formula (2.28) los cuatro triángulos tendrán áreas positivas, y si los cuatro triángulos tienen área positiva, entonces la celda es convexa. La figura 2.13 muestra un ejemplo de una celda convexa y una no convexa.

Ya se establecieron por separado los argumentos respecto al Jacobiano y a la convexidad de una celda, ahora falta establecer que relación hay entre el Jacobiano y la convexidad. Supongase que de nuevo que se tiene un mapeo bilineal $\bar{\mathbf{x}}$ como el de la figura 2.11, se encontró que

$$J(0,0) = (Q_x - P_x)(S_y - P_y) - (S_x - P_x)(Q_y - P_y)$$

acomodando los términos del lado derecho de la igualdad anterior en forma de determinante se tiene que

$$\begin{aligned} J(0,0) &= (Q_x - P_x)(S_y - P_y) - (S_x - P_x)(Q_y - P_y) = \\ &= \begin{vmatrix} (Q_x - P_x) & (S_x - P_x) \\ (Q_y - Py) & (S_y - P_y) \end{vmatrix} = \\ &= (Q-P)^t J_2 (S-P) = 2\alpha(\triangle PQS) = 2\alpha(\triangle SPQ), \end{aligned}$$

por tanto, $J(0,0) >$ si $\alpha(\triangle PQS) > 0$, es decir, $J(0,0) >$ si y sólo si $\alpha(\triangle SPQ) > 0$.

para $J(1,0)$ se tenia que

$$J(1,0) = J(0,0) + (Q_x - P_x)(R_y + P_y - S_y - Q_y) - (Q_y - P_y)(R_x + P_x - S_x - Q_x)$$

por tanto

$$\begin{aligned} J(1,0) &= (Q_x - P_x)(S_y - P_y) - (S_x - P_x)(Q_y - P_y) + \\ &+ (Q_x - P_x)(R_y + P_y - S_y - Q_y) - (Q_y - P_y)(R_x + P_x - S_x - Q_x) = \\ &= (R_x - Q_x)(P_y - Q_y) - (R_y - Q_y)(P_x - Q_y) = \\ &= \begin{vmatrix} (R_x - Q_x) & (P_x - Q_x) \\ (R_y - Q_y) & (P_y - Q_y) \end{vmatrix} = \\ &= 2\alpha(\triangle QRP) = 2\alpha(\triangle PQR), \end{aligned}$$

análogamente se tendrá que

$$J(0,1) = \alpha(\triangle RSP),$$

$$J(1,1) = \alpha(\triangle QRS),$$

por lo que se puede concluir que $J(0,0)$, $J(1,0)$, $J(0,1)$ y $J(1,1)$ son positivos si y sólo si $\alpha(\triangle SPQ), \alpha(\triangle PQR), \alpha(\triangle RSP), \alpha(\triangle QRS) > 0$, es decir, $J(0,0)$, $J(1,0)$, $J(0,1)$ y $J(1,1)$ son positivos si y sólo si el área de cada uno de los cuatro triángulos de la figura 2.12 son positivas.

Se encontró que el Jacobiano de un mapeo bilineal es lineal, por lo que el lugar geométrico del Jacobiano es un plano, por lo que el valor mínimo del Jacobiano estará en una de sus esquinas del cuadrado unitario. Por tanto, se concluye que el Jacobiano es positivo si y solo si la celda $\square PQRS$ es convexa, esta es precisamente la relación entre el Jacobiano y la convexidad de una Celda.

2.4.3. Discretización del funcional de suavidad

A continuación se muestra la discretización del funcional de suavidad de Winslow (sección 2,3 de [4], esto se hace para poder trabajar en un problema discreto que aproxíma el problema continuo. El funcional de Wislow está dado por 2.10, y en su forma matricial esta dado por

$$I[\bar{\mathbf{X}}] = \int_0^1 \int_0^1 \frac{\| \bar{\mathbf{X}}_\xi \|^2 + \| \bar{\mathbf{X}}_\eta \|^2}{\bar{\mathbf{X}}_\xi^t J_2 \bar{\mathbf{X}}_\eta} d\xi d\eta,$$

donde

$$J_2 = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix}.$$

Haciendo uso de los mapeos bilineales se puede formular el funcional $I[\bar{\mathbf{X}}]$ como un funcional compuesto de un conjunto finito de mapeos bilineales (filosofía de divide y vencerás). Sea G_B una cuadrícula uniforme estructurada en el cuadrado unitario $\mathbf{B}$, bajo el mapeo esta malla sería transformada a una malla G_Ω en Ω. Si $B_{i,j}$ es una celda de la malla G_B que es mapeada a una celda $C_{i,j}$ de la malla en Ω, entonces bajo el mapeo $\bar{\mathbf{X}}$ se debe cumplir la condición

$$\bar{\mathbf{X}}(B_{i,j}) = C_{i,j}. \tag{2.29}$$

Usando también el hecho que de cada mapeo bilineal que va de una celda de B a una celda $C_{i,j}$ de la malla en Ω, entonces se puede expresar el funcional de Winslow como la siguiiente sumatoria

$$I[\bar{\mathbf{X}}] = \sum_{k=1}^{N} \int_{B_{i,j}} \frac{\| \bar{\mathbf{X}}_\xi \|^2 + \| \bar{\mathbf{X}}_\eta \|^2}{\bar{\mathbf{X}}_\xi^t J_2 \bar{\mathbf{X}}_\eta} d\xi d\eta, \tag{2.30}$$

en donde N es el número de celdas de G_B, por tanto, si se discretiza cada evaluación del funcional en cada mapeo bilineal, entonces se conseguirá una distcretización del funcional en el mapeo principal $\bar{\mathbf{X}}$.

Por tanto, para hacer la discretización del funcional de Winslow es conveniente primero realizar una discretización de una integral de (2.30). Sea $\bar{x}_{i,j}$ un mapeo bilineal cualquiera entre una celda $B_{i,j}$ de $\mathbf{B}$ y una celda $C_{i,j}$ de la malla por construir en Ω, la figura 2.14 muestra una ilustración del mapeo $\bar{x}_{i,j}$.

Es por esto que el funcional de Winslow se aproxima de la siguiente manera

$$I[\bar{\mathbf{X}}] \approx \sum_{k=1}^{N} \int_{B_{i,j}} \frac{\| \frac{\partial \bar{x}_{(i,j)}}{\partial \xi} \|^2 + \| \frac{\partial \bar{x}_{(i,j)}}{\partial \eta} \|^2}{\frac{\partial \bar{x}_{(i,j)}}{\partial \xi}^t J_2 \frac{\partial \bar{x}_{(i,j)}}{\partial \eta}} d\xi d\eta.$$

Dado que la malla es de G_B es de dimensiones $n \times m$, entonces $N = (n-1)(m-1)$.

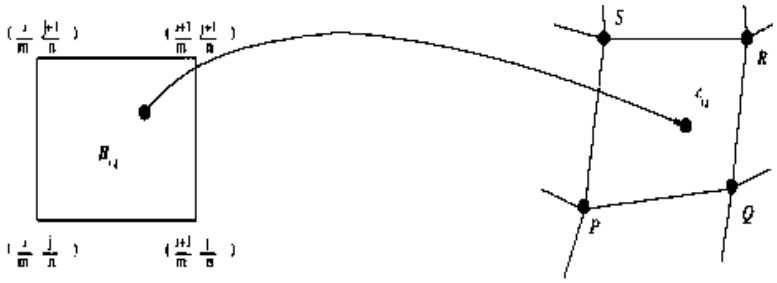

Figura 2.14: Mapeo bilineal $\bar{x}_{i,j}$ establecido entre la celda $B_{i,j}$ y el cuadrilátero $C_{i,j}$.

La forma general de un mapeo bilineal como $\bar{x}_{i,j}$ es la siguiente

$$\bar{x}_{i,j}(\xi,\eta) = \vec{A} + \vec{B}\left(\xi - \frac{i}{m}\right) + \vec{C}\left(\eta - \frac{j}{n}\right) + \vec{D}\left(\xi - \frac{i}{m}\right)\left(\eta - \frac{j}{n}\right), \quad (2.31)$$

Se establece el siguiente orden en la correspondencia de las esquinas en el mapeo bilineal (ver figura 2.14)

$$\bar{x}_{i,j}\left(\frac{i}{m},\frac{j}{n}\right) = \vec{P}, \qquad \bar{x}_{i,j}\left(\frac{i+1}{m},\frac{j}{n}\right) = \vec{Q},$$

$$\bar{x}_{i,j}\left(\frac{i+1}{m},\frac{j+1}{n}\right) = \vec{R}, \qquad \bar{x}_{i,j}\left(\frac{i}{m},\frac{j+1}{n}\right) = \vec{S},$$

nótese que se se está usando notación vectorial. Al evaluar en el mapeo (2.31), se tiene que

$$\begin{aligned}
\bar{x}_{i,j}\left(\frac{i}{m},\frac{j}{n}\right) &= \vec{A} \\
&\Rightarrow \quad \vec{A} = \vec{P}
\end{aligned}$$

$$\begin{aligned}
\bar{x}_{i,j}\left(\frac{i+1}{m},\frac{j}{n}\right) &= \vec{A} + \frac{\vec{B}}{m} \\
&\Rightarrow \quad \vec{A} + \frac{\vec{B}}{m} = \vec{Q}
\end{aligned}$$

$$\begin{aligned}
\bar{x}_{i,j}\left(\frac{i+1}{m},\frac{j+1}{n}\right) &= \vec{A} + \frac{\vec{B}}{m} + \frac{\vec{C}}{n} + \frac{\vec{D}}{mn} \\
&\Rightarrow \quad \vec{A} + \frac{\vec{B}}{m} + \frac{\vec{C}}{n} + \frac{\vec{D}}{mn} = \vec{R}
\end{aligned}$$

$$\begin{aligned}\bar{x}_{i,j}\left(\frac{i}{m},\frac{j+1}{n}\right) &= \vec{A}+\frac{\vec{C}}{n}\\ &\Rightarrow \quad \vec{A}+\frac{\vec{C}}{n}=\vec{S}\end{aligned}$$

resolviendo el sistema anterior, se tiene que los coeficientes de (2.31) están dados por

$$\vec{A}=\vec{P}, \qquad \vec{B}=m(\vec{Q}-\vec{P})$$

$$\vec{C}=n(\vec{S}-\vec{P}), \qquad \vec{D}=mn(\vec{R}+\vec{P}-\vec{Q}-\vec{S}),$$

sustituyendo en (2.31), entonces el mapeo tienen la forma siguiente

$$\begin{aligned}\bar{x}_{i,j}(\xi,\eta) &= \vec{P}+m(\vec{Q}-\vec{P})\left(\xi-\frac{i}{m}\right)+n(\vec{S}-\vec{P})\left(\eta-\frac{j}{n}\right)+\\ &\quad +mn(\vec{R}+\vec{P}-\vec{Q}-\vec{S})\left(\xi-\frac{i}{m}\right)\left(\eta-\frac{j}{n}\right). \qquad (2.32)\end{aligned}$$

Esto implica que las derivadas parciales del mapeo bilineal están dadas por las siguientes expresiones

$$\frac{\partial\bar{x}_{i,j}(\xi,\eta)}{\partial\xi}=m(\vec{Q}-\vec{P})+mn(\vec{R}+\vec{P}-\vec{Q}-\vec{S})(\eta-\frac{j}{n}), \qquad (2.33)$$

$$\frac{\partial\bar{x}_{i,j}(\xi,\eta)}{\partial\eta}=n(\vec{S}-\vec{P})+mn(\vec{R}+\vec{P}-\vec{Q}-\vec{S})(\xi-\frac{i}{m}). \qquad (2.34)$$

Evaluando las derivadas parciales en las esquinas de la celda $B_{i,j}$, se tienen las siguientes expresiones

$$\frac{\partial\bar{x}_{i,j}(\frac{i}{m},\frac{j}{n})}{\partial\xi}=m(\vec{Q}-\vec{P}), \qquad \frac{\partial\bar{x}_{i,j}(\frac{i+1}{m},\frac{j}{n})}{\partial\xi}=m(\vec{Q}-\vec{P})$$

$$\frac{\partial\bar{x}_{i,j}(\frac{i+1}{m},\frac{j+1}{n})}{\partial\xi}=m(\vec{R}-\vec{S}), \qquad \frac{\partial\bar{x}_{i,j}(\frac{i}{m},\frac{j+1}{n})}{\partial\xi}=m(\vec{R}-\vec{S})$$

$$\frac{\partial\bar{x}_{i,j}(\frac{i}{m},\frac{j}{n})}{\partial\eta}=n(\vec{S}-\vec{P}), \qquad \frac{\partial\bar{x}_{i,j}(\frac{i+1}{m},\frac{j}{n})}{\partial\eta}=n(\vec{R}-\vec{Q})$$

$$\frac{\partial\bar{x}_{i,j}(\frac{i+1}{m},\frac{j+1}{n})}{\partial\eta}=n(\vec{R}-\vec{Q}), \qquad \frac{\partial\bar{x}_{i,j}(\frac{i}{m},\frac{j+1}{n})}{\partial\eta}=n(\vec{S}-\vec{P}).$$

Ahora, haciendo uso de una cuadratura para aproximar la integral en $B_{i,j}$, entonces se tiene que

$$\int_{B_{i,j}} \frac{\| \frac{\partial \bar{x}_{i,j}}{\partial \xi} \|^2 + \| \frac{\partial \bar{x}_{i,j}}{\partial \eta} \|^2}{\frac{\partial \bar{x}_{i,j}}{\partial \xi}^t J_2 \frac{\partial \bar{x}_{i,j}}{\partial \eta}} d\xi d\eta \approx \frac{1}{4}\left(\frac{\| \frac{\partial \bar{x}_{i,j}(\frac{i}{m},\frac{j}{n})}{\partial \xi} \|^2 + \| \frac{\partial \bar{x}_{i,j}(\frac{i}{m},\frac{j}{n})}{\partial \eta} \|^2}{\frac{\partial \bar{x}_{i,j}(\frac{i}{m},\frac{j}{n})}{\partial \xi}^t J_2 \frac{\partial \bar{x}_{i,j}(\frac{i}{m},\frac{j}{n})}{\partial \eta}} \right) +$$
$$+\frac{1}{4}\left(\frac{\| \frac{\partial \bar{x}_{i,j}(\frac{i+1}{m},\frac{j}{n})}{\partial \xi} \|^2 + \| \frac{\partial \bar{x}_{i,j}(\frac{i+1}{m},\frac{j}{n})}{\partial \eta} \|^2}{\frac{\partial \bar{x}_{i,j}(\frac{i+1}{m},\frac{j}{n})}{\partial \xi}^t J_2 \frac{\partial \bar{x}_{i,j}(\frac{i+1}{m},\frac{j}{n})}{\partial \eta}} \right) +$$
$$+\frac{1}{4}\left(\frac{\| \frac{\partial \bar{x}_{i,j}(\frac{i+1}{m},\frac{j+1}{n})}{\partial \xi} \|^2 + \| \frac{\partial \bar{x}_{i,j}(\frac{i+1}{m},\frac{j+1}{n})}{\partial \eta} \|^2}{\frac{\partial \bar{x}_{i,j}(\frac{i+1}{m},\frac{j+1}{n})}{\partial \xi}^t J_2 \frac{\partial \bar{x}_{i,j}(\frac{i+1}{m},\frac{j+1}{n})}{\partial \eta}} \right) +$$
$$+\frac{1}{4}\left(\frac{\| \frac{\partial \bar{x}_{i,j}(\frac{i}{m},\frac{j+1}{n})}{\partial \xi} \|^2 + \| \frac{\partial \bar{x}_{i,j}(\frac{i}{m},\frac{j+1}{n})}{\partial \eta} \|^2}{\frac{\partial \bar{x}_{i,j}(\frac{i}{m},\frac{j+1}{n})}{\partial \xi}^t J_2 \frac{\partial \bar{x}_{i,j}(\frac{i}{m},\frac{j+1}{n})}{\partial \eta}} \right),$$

sustituyendo las expresiones de las derivadas parciales de $\bar{x}_{i,j}$ evaluadas en las esquinas, entonces

$$\int_{B_{i,j}} \frac{\| \frac{\partial \bar{x}_{(i,j)}}{\partial \xi} \|^2 + \| \frac{\partial \bar{x}_{(i,j)}}{\partial \eta} \|^2}{\frac{\partial \bar{x}_{(i,j)}}{\partial \xi}^t J_2 \frac{\partial \bar{x}_{(i,j)}}{\partial \eta}} d\xi d\eta \approx \frac{1}{4}\left(\frac{m^2 \| \vec{Q}-\vec{P} \|^2 + n^2 \| \vec{S}-\vec{P} \|^2}{mn(\vec{Q}-\vec{P})J_2(\vec{S}-\vec{P})} \right) +$$
$$+\frac{1}{4}\left(\frac{m^2 \| \vec{Q}-\vec{P} \|^2 + n^2 \| \vec{R}-\vec{Q} \|^2}{mn(\vec{Q}-\vec{P})J_2(\vec{R}-\vec{Q})} \right) + \frac{1}{4}\left(\frac{m^2 \| \vec{R}-\vec{S} \|^2 + n^2 \| \vec{R}-\vec{Q} \|^2}{mn(\vec{R}-\vec{S})J_2(\vec{R}-\vec{Q})} \right) +$$
$$+\frac{1}{4}\left(\frac{m^2 \| \vec{R}-\vec{S} \|^2 + n^2 \| \vec{S}-\vec{P} \|^2}{mn(\vec{R}-\vec{S})J_2(\vec{S}-\vec{P})} \right).$$

Se puede observar que en la expresión anterior cada sumando de la aproximación depende únicamente de tres puntos del cuadrilatero $C_{i,j}$ y no de los cuatro puntos, esto hace que se comience en pensar en los triángulos que conforman la celda.

Nótese que los denominadores de la aproximación de la integral en $B_{i,j}$ se pueden expresar en términos de la función de área (2.28), pues por ejemplo, para el denominador del primer término se cumple que

$$(\vec{Q}-\vec{P})J_2(\vec{S}-\vec{P}) = 2\alpha(\triangle PQS).$$

Además, si se observan los numeradores, puede notarse que se están calculando la suma de los cuadrados de las longitudes de los lados del triángulo que no son una diagonal, por tanto se define la función $\lambda(\triangle PQS)$ dada por

$$\lambda(\triangle PQS) = \| \vec{Q}-\vec{P} \|^2 + \| \vec{S}-\vec{P} \|^2 . \tag{2.35}$$

Por tanto, para compactar la notación, se define la función f como

$$f(\triangle PQS) = \frac{\lambda(\triangle PQS)}{2Area(\triangle PQS)},$$

aplicando esta notación en la aproximación de la integral sobre $B_{i,j}$, se tiene que

$$\int_{B_{i,j}} \frac{\| \frac{\partial \bar{x}_{(i,j)}}{\partial \xi} \|^2 + \| \frac{\partial \bar{x}_{(i,j)}}{\partial \eta} \|^2}{\frac{\partial \bar{x}_{(i,j)}}{\partial \xi}^t J_2 \frac{\partial \bar{x}_{(i,j)}}{\partial \eta}} d\xi d\eta \approx$$

$$\approx \frac{1}{4}\left[\frac{\lambda(\triangle PQS)}{2Area(\triangle PQS)} + \frac{\lambda(\triangle QRP)}{2Area(\triangle QPR)} + \frac{\lambda(\triangle RSQ)}{2Area(\triangle RSQ)} + \frac{\lambda(\triangle SPR)}{2Area(\triangle SPR)}\right] =$$

$$= \frac{1}{4}\left[f(\triangle PQS) + f(\triangle QPR) + f(\triangle RSQ) + f(\triangle SPR)\right] = \frac{1}{4}\sum_{k=1}^{4} f(\triangle^k),$$

donde la variable k sirve para denotar cada uno de los cuatro triángulos que componen la celda. Finalmente se tiene la aproximación del funcional de Winslow dada por

$$I[\mathbf{X}] \approx \frac{1}{4}\sum_{i=1}^{m-1}\sum_{j=1}^{n-1}\sum_{k=1}^{4} f(\triangle_{i,j}^k), \tag{2.36}$$

donde los índices i, j denotan la celda y k uno de los triángulos de la celda, es decir, la sumatoria anterior es sobre todos los triangulos que componen la malla.

2.5. Generación de mallas adaptativas

2.5.1. Planteamiento

Para los propósitos de este trabajo sólo se tuvo interés en funcional de suavidad por una razón muy especial, dado que se planteó desde el inicio del presente trabajo, el uso de mallas adaptativas, entonces se empleó la teoría para generar dichas mallas [9], pero para generar este tipo de mallas sólo se dispone de teoría para el funcional de suavidad, esta es la razon por la que solamente se describió la teoría del funcional de suavidad.

Anteriormente se discutió la teoría de generación variacional de mallas, esto implicaba encontrar un mapeo que debia cumplir con características especiales. La teoría para generar mallas adaptativas variacionalmente es casi la misma, sólo que entra en juego un nuevo factor, el cual es una función $g : \mathbb{R}^2 \longrightarrow \mathbb{R}$ que es precisamente la función a la que la malla se adapta, de aquí el nombre

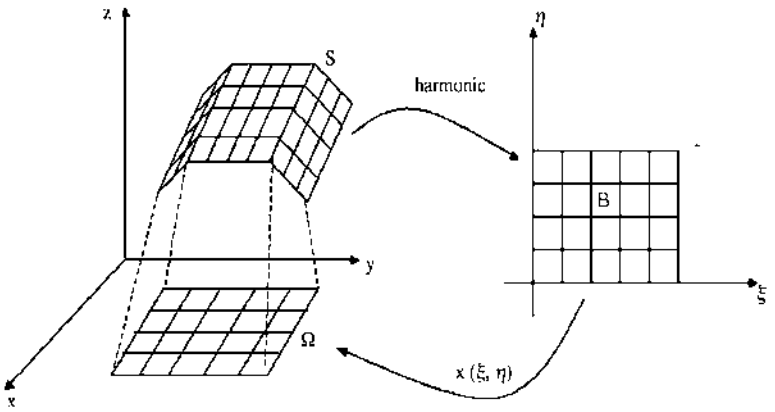

Figura 2.15: Mapeos involucrados en la generación de mallas adaptativas

de mallas adaptativas.

De igual manera, se considera que la región física Ω y el cuadrado unitario B son regiones cuyas fronteras son poligonales cerradas orientadas positivamente. La imagen 2.15 muestra en que consiste este nuevo enfoque para la generación de mallas adaptativas, en el cual están involucrados dos mapeos en lugar de sólo uno.

Simplificando ideas, el problema en el el caso de mallas adaptativas es encontrar un mapeo $\mathbf{x} : \mathbf{B} \longrightarrow \Omega$ (2.3) tal que:

- $\mathbf{x}$ sea uno a uno.
- $\mathbf{x}$ sea diferenciable.
- $\mathbf{x}$ satisfaga que el Jacobiano sea positivo en B.
- $\partial\Omega = \mathbf{x}(\partial B)$.

En la figura 2.15 se aprecia como además de tener un mapeo del cuadrado unitario $\mathbf{B}$ a la region fisica Ω, también se tiene una superficie S, la cual se obtiene al evaluar la función g en la región Ω.

En la sección anterior se discretizó el funcional de suavidad de Winslow, obteniendo así un funcional discreto, el cual se denotará por la siguiente expresión

$$H(G) = \sum_{n}^{i=1} \sum_{m}^{j=1} \sum_{4}^{k=1} \frac{\lambda(\triangle_k^{i,j})}{\alpha(\triangle_k^{i,j})}, \tag{2.37}$$

donde para compactar la notación, se definió la función f tal que

$$f(\triangle_k^{i,j}) = \frac{\lambda(\triangle_k^{i,j})}{\alpha(\triangle_k^{i,j})},$$

donde G es una malla de dimensión de $n \times m$ y las funciones λ y α están dadas por (2.35) y (2.28), respectivamente.

La malla G será convexa si $min\{\alpha(\triangle_q)\} > 0$, donde $\triangle_q$ denota un triángulo cualquiera de la malla.

Por tanto, en lugar encontrar el óptimo del funcional continuo (2.10), el problema ahora consiste en resolver un problema de optimización, en el cual se debe encontrar la malla G^* tal que

$$G^* = argmin_G \sum_{q=1}^{N} H(\triangle_q),$$

donde N es el número total de triángulos de la malla, como la malla es de dimensión $n \times m$, entonces $N = 4(n-1)(m-1)$.

Sin embargo existe en general una dificultad, pues para el problema de optimización dado por la expresión anterior necesita una malla convexa inicial, pero existen variaciones del funcional discreto que no requieren una malla inicial convexa. Afortunadamente, este no es un problema en la presente tesis, pues como el propósito es generar mallas adaptativas en imagenes, entonces la malla inicial estará dada por una malla rectangular estructurada con frontera siempre rectangular.

Pero para la optimización necesaria en la generación de mallas adaptativas en imágenes no se usó el funcional H, sino el funcional H_ω, el cual es una variación de H. Este funcional fue producto del trabajo de Barrera-Sánchez, Domínguez-Mota, Ivanenko y Tinoco-Ruiz, donde H_ω es llamado funcional de suavidad discreto quasi-armónico que depende del parámetro ω.

La diferencia entre H y H_ω es de solamente cambiar el término $1/\alpha(\triangle_q)$ por la función $\phi_\omega(\alpha(\triangle_q))$, la cual está dada por la siguiente expresión

$$\phi_\omega(\alpha(\triangle_q)) = \begin{cases} \frac{2\omega - \alpha(\triangle_q)}{\omega 2}, & si \quad \alpha(\triangle_q) < \omega \\ \\ \frac{1}{\alpha(\triangle_q)} & si \quad \alpha(\triangle_q) \geq \omega. \end{cases} \tag{2.38}$$

El propósito principal de modificar el funcional H por el funcional H_ω es que la función (2.38) penaliza con mayor rigor las áreas negativas que se puedan llegar a tener por la no convexidad de algunos triángulos de la malla durante el proceso de optimización. Por tanto, el funcional H_ω está dado por la siguiente ecuación

$$H_\omega(\triangle_q) = \lambda(\triangle_q)\phi_\omega(\triangle_q).$$

Sin embargo, el funcional H_ω debe incorporar una función g a la cual la malla se adaptará. La idea principal para incorporar la función g en las mallas adaptativas es el incorporar un mapeo armónico de la superficie S al cuadrado unitario, donde S es la superficie de la función $g(x, y)$ en Ω, la figura 2.15 muestra esta idea.

Se formuló el funcional continuo I_{Ad}

$$I_{Ad} = \int_0^1 \int_0^1 \frac{(x_\xi^2 + x_\eta^2)(1+g_x)^2 + 2g_x g_y(x_\xi y_\xi + x_\eta y_\eta) + (y_\xi^2 + y_\eta^2)(1+g_y)^2}{(x_\xi y_\eta - x_\eta y_\xi)\sqrt{1+g_x^2+g_y^2}} d\xi d\eta, \tag{2.39}$$

En forma matricial está dado por

$$I_{Ad} = \int_0^1 \int_0^1 \frac{\mathbf{x}_\xi^t \mathbf{A} \mathbf{x}_\xi + \mathbf{x}_\eta^t \mathbf{A} \mathbf{x}_\eta}{(x_\xi y_\eta - x_\eta y_\xi)\sqrt{1+g_x^2+g_y^2}} d\xi d\eta, \tag{2.40}$$

donde

$$\mathbf{x}(\xi,\eta) = \begin{pmatrix} x(\xi,\eta) \\ y(\xi,\eta) \end{pmatrix}, \qquad \mathbf{A} = \mathbf{A}(P) = \begin{pmatrix} 1+g_x^2 & g_x g_y \\ g_x g_y & 1+g_y^2 \end{pmatrix};$$

nótese que la definición de la matriz $\mathbf{A}$ está en función de un punto P del plano, en el cual se obtienen las parciales de g para definir $\mathbf{A}$.

El funcional adaptativo I_{Ad} es una forma general del funcional de suavidad, pues si se considera una función constante g, entonces las derivadas parciales con respecto a x e y son cero, por lo que I_{Ad} se reduce al funcional de suavidad propuesto por Brackbill y Saltman

$$\int_0^1 \int_0^1 \frac{(x_\xi^2 + x_\eta^2)(1+0)^2 + 2(0)(0)(x_\xi y_\xi + x_\eta y_\eta) + (y_\xi^2 + y_\eta^2)(1+0)}{(x_\xi y_\eta - x_\eta y_\xi)\sqrt{1+0+0}} d\xi d\eta =$$

$$= \int_0^1 \int_0^1 \frac{(x_\xi^2 + x_\eta^2 + y_\xi^2 + y_\eta^2)}{x_\xi y_\eta - x_\eta y_\xi} d\xi d\eta = \int_0^1 \int_0^1 \frac{(x_\xi^2 + x_\eta^2 + y_\xi^2 + y_\eta^2)}{J} d\xi d\eta = I_S.$$

Finalmente, al discretizar (2.39) de manera similar a como se hizo con el funcional de suavidad de Winslow, se obtiene H_{Ad}, el cual está dado por la siguiente expresión

$$H_{Ad}(G) = \sum_{i=1}^{n} \sum_{j=1}^{m} \sum_{k=1}^{4} \lambda_{Ad}(\triangle_{i,j}^k)\phi(\triangle_{i,j}^k), \tag{2.41}$$

donde el único cambio es el uso de la función λ_{Ad}, que está definida por

$$\lambda_{Ad}(\triangle ABC) = \frac{(A-B)^t \mathbf{A}(B)(A-B) + (C-B)^t \mathbf{A}(B)(C-B)}{\sqrt{det(\mathbf{A}(B))}}.$$

La figura 2.16 se muestra un ejemplo de una malla generada con el funcional de suavidad de Winslow. La función g definida de la siguiente manera se empleó para mostrar un ejemplo de generación de una malla adaptativa, empleando una función continua

$$g(x,y) = e^{-2(4(-1-2x)^2+9(2y-1)^2)^2-1}.$$

La función g se muestra en la figura 2.17, se puede observar que tiene una forma peculiar, pues tiene zonas donde el gradiente cambia radicalmente.

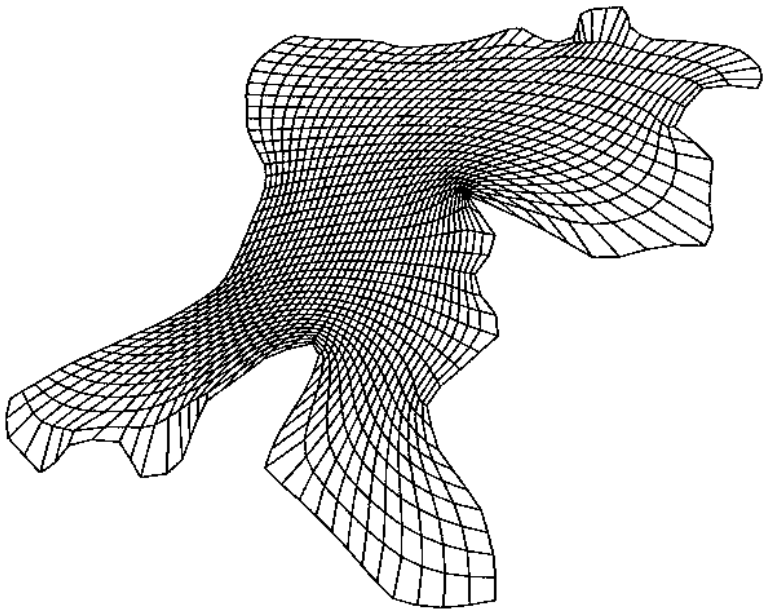

Figura 2.16: Malla generada usando el funcional de suavidad de Winslow en la región de la Habana.

La razón de usar mallas adptativas es que se producen celdas más elongadas en la malla sobre las regiones donde el gradiente de la función $g(x,y)$ es más alto, como se muestra en la figura 2.18, en la práctica se ha encontrado que en celdas elongadas se tienen mejores aproximaciones usando diferencias finitas.

Además hay que recordar que en el capítulo 1 se vio que por la definición de la función de difusión $c(\nabla I)$, los bordes son conservados en las regiones de la

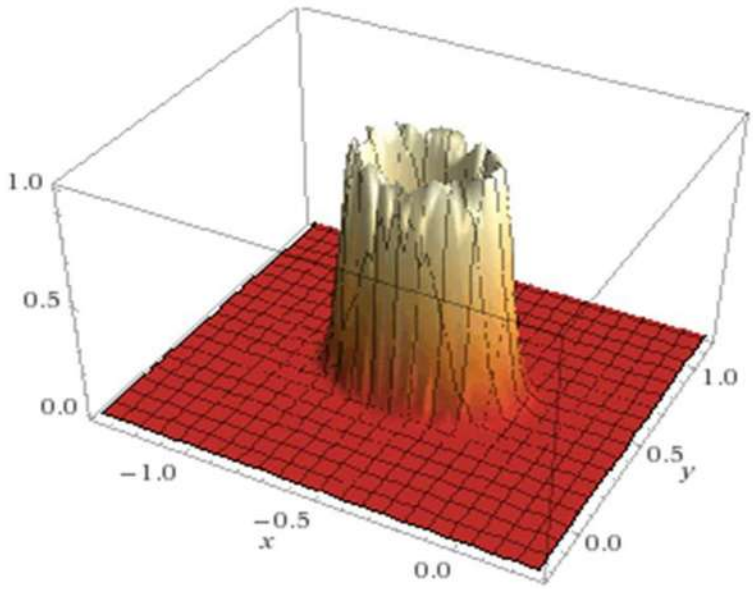

Figura 2.17: Grafica de la función $g(x, y)$

imagen donde se tiene un gradiente alto en la función de intensidad, entonces si se usa la función de intensidad de las imágenes como la función $g(x, y)$, el resultado seía que se obtendrán celdas elongadas las regiones cercanas a los bordes de la imagen. Esta es la principal razón de porque se decidió emplear específicamente mallas adaptativas en este trabajo. En el siguiente capítulo se muestran ejemplos de mallas adaptativas generadas en imágenes.

2.5.2. Problema de optimización a gran escala

Durante la elaboración de esta tesis se generaron mallas adaptativas a partir de imágenes digitales de dos dimensiones con una resolución definida. En un principio se tenia pensado usar imágenes cuya resolución oscilará entre 250×250 y 600×600 píxeles, pero debido a que hoy en dia la resolución de las imágenes cada vez es más alta, entonces se intentó enfocar el presente trabajo para imágenes de resoluciones más altas. Esto implica que el problema de busqueda de una malla adaptativa de una imagen optimizando el funcional (2.41) se convierta en un problema de optimización a gran escala, las nociones básicas de esta clase de problemas puede consultarse en el apendice $textbfB$ de [4], pues suponiendo que dada una imagen de resolución de $n \times m$ píxeles y que por lo menos se genere una malla adaptativa de igual dimensiones, entonces el número de variables a optimizar es muy grande, más adelante se analíza esta problemática con más detalle.

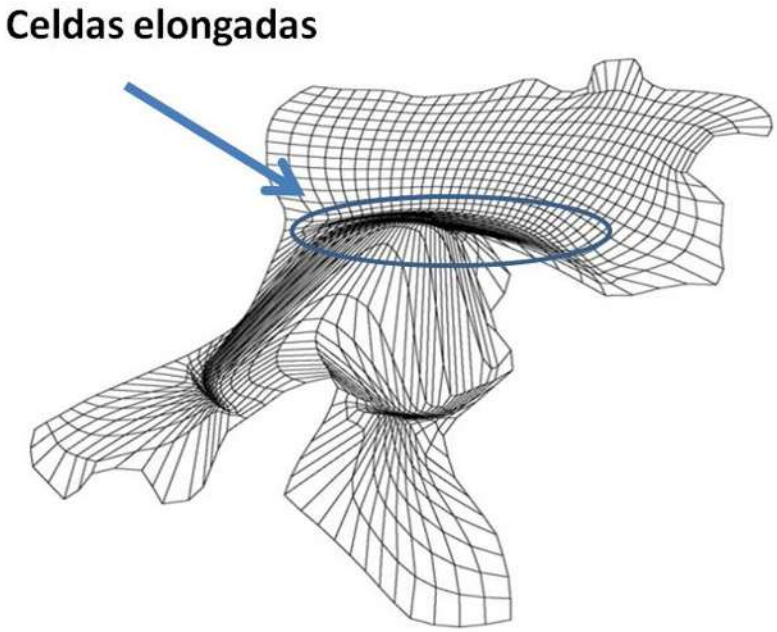

Figura 2.18: Malla adaptativa generada sobre la malla de la habana y la función $g(x, y)$, donde se producen celdas elongadas.

Un problema de optimización en general es un problema en el que básicamente se busca una solución que minimice (o maximice) el problema incicial. Este tipo de problemas es común en matemáticas, la industria e ingeniería, por mencionar algunas áreas. Existe toda una teoría de métodos de optimización, sin embargo, la optimización es una rama de las matemáticas que sigue en desarrollo, sobre todo la optimización discreta, pues se busca mejorar la precisión y rapidez de los algoritmos ya existentes. Actaulmente existen problemas de optimización discreta que no se han podido resolver.

Los métodos de optimización generan una sucesión $\{x_k\}$, con $k \in 0 \cup \mathbb{N}$, donde el término x_0 es el término inicial. En estos métodos se realiza un número finito de iteraciones y donde en cada iteración se obtiene un término más óptimo o igual al anterior. Los métodos de optimización terminan cuando no se cumple un criterio establecido.

Supongase que se tiene una función $f : \mathbb{R}^n \longrightarrow \mathbb{R}^n$, en la cual se tiene interés en encontrar el mínimo local $x^* \in \mathbb{R}^n$, por tanto, se debe emplear un método de optimización de tipo descenso, es decir, se emplea un método de optimización que genera una sucesión $\{x_k\}$ donde se cumpla que

$$f(x_{k+1}) < f(x_k),$$

es decir, se busca que en cada iteración se obtenga un término que produzca una reducción en la función f. La función f es llamada **función objetivo** y una de las características principales de los métodos de descenso es que en cada iteración se debe encontrar un vector p llamado **vector de descenso**, pues el vector p nos indica en que dirección la funcion f tiene un máximo decrecimiento.

Por tanto, en la iteración k el vector p_k denotará el vector de descenso en x_k siempre y cuando la derivada direccional de la función objetivo en la dirección p_k es negativa, es decir, p_k será el vector de descenso de la iteración k si

$$p_k \bigtriangledown f(x_k) < 0.$$

En los cursos de optimización de nivel licenciatura se estudian métodos de optimización para minimizar funciones objetivo de variable real, en otras palabras, la función objetivo f es tal que $f : \mathbb{R} \longrightarrow \mathbb{R}$.

Pero si la función objetivo f tiene como dominio $\mathbb{R}^n$, donde n es grande, es decir, si se tiene un número muy grande de variables en la optimización, entonces tenemos un **problema de optimización a gran escala**.

Este tipo de problemas tienen una teoría matemática bien planteada, capítulo 7 de [8] para ser resueltos, sin embargo, estos problemas son dificiles de implementar, pues los algoritmos de la teoría sólo indican el procedimiento, pero no se menciona el costo computacional que tienen en un problema en específico.

Dado que el número de componentes del vector x_k (en cada iteración) es muy grande, esto implica el uso de una cantidad muy grande de memoria tan sólo almacenar los datos, sin mencionar que el número de operaciones para realizar una iteración es muy grande, esto repercute en que tiempo ejecución también sea muy alto.

Es por esto que a mediados del siglo pasado se desarrollaron algoritmos que solucionaban esta problemática, pues en 1950 el fisíco estaunidense Davidon, que trabajaba en laboratorio nacional Argonne, estaba empleando un método de descenso para realizar un proceso de optimización a gran escala. Dado que en ese tiempo las computadoras estaban en sus inicios, el poder de computo era muy limitado.

Debido a que los cálculos que intentaba hacer Davidon no llegaban a terminar, pues el proceso siempre era interrumpido por las limitaciones computacionales, entonces Davidon cansado y frustrado decidió idear una forma de mejorar el proceso de optimización.

Este fue el nacimiento de los métodos de optimización Quasi-Newton, los cuales se han convertido en los métodos mas creativos de la optimización no lineal. Fletcher y Powell demostraron después que este nuevo tipo de métodos eran más rapidos y confiables que los demás métodos de esa época. Desde entonces han sido propuestas distintas variantes de los métodos Quasi-Newton.

Tan sólo como dato histórico y curioso, el articulo original que Davidon elaboró, en el cual planteó sus ideas para la creación de los métodos Quasi-Newton no fue aceptado para publicarse.

2.5.3. Solución al problema de optimización a gran escala para la generación de mallas adaptativas

Puesto que se desea generar mallas adaptativas a partir de imágenes, entonces en el proceso de optimización a gran escala se utilizó como malla inicial un malla rectangular estructurada y uniforme con frontera rectangular, como la que se muestra en la figura 2.19.

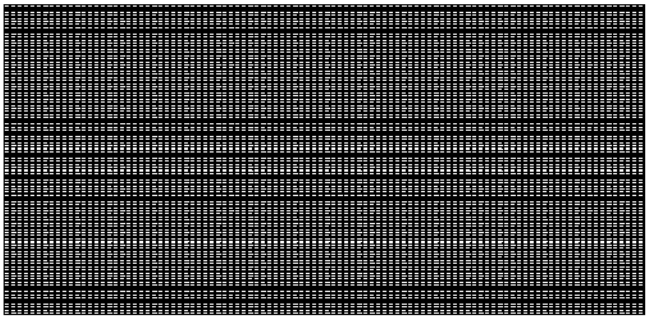

Figura 2.19: Malla inicial empleada para generar una malla adaptativa en una imagen digital.

El objetivo del problema de optimización a gran escala es encontrar una malla convexa G^* que minimice el funcional de suavidad armónico $H_{Ad}(G)$ dado por (2.41) , es decir, se busca una malla convexa $G*$ tal que

$$G^* = Argmin_{G \in M(\Omega)} H_{Ad}(G),$$

donde $M(\Omega)$ es el conjunto de mallas convexas en la región fisíca Ω.

Es importante establecer las ideas que se plantean hasta el momento, por ejemplo, en el problema de optimización, la función objetivo es el funcional $H_{Ad}(G)$ y la malla $G*$ juega el papel del vector x_k , pues el argumento del funcional es

una malla. La figura 2.20 muestra la forma en que se enumeraron los nodos de la malla, dado que la malla adaptativa óptima que se desea obtener es de una imagen y la frontera se mantiene fija, entonces solamente los nodos interiores participan en el proceso de optimización.

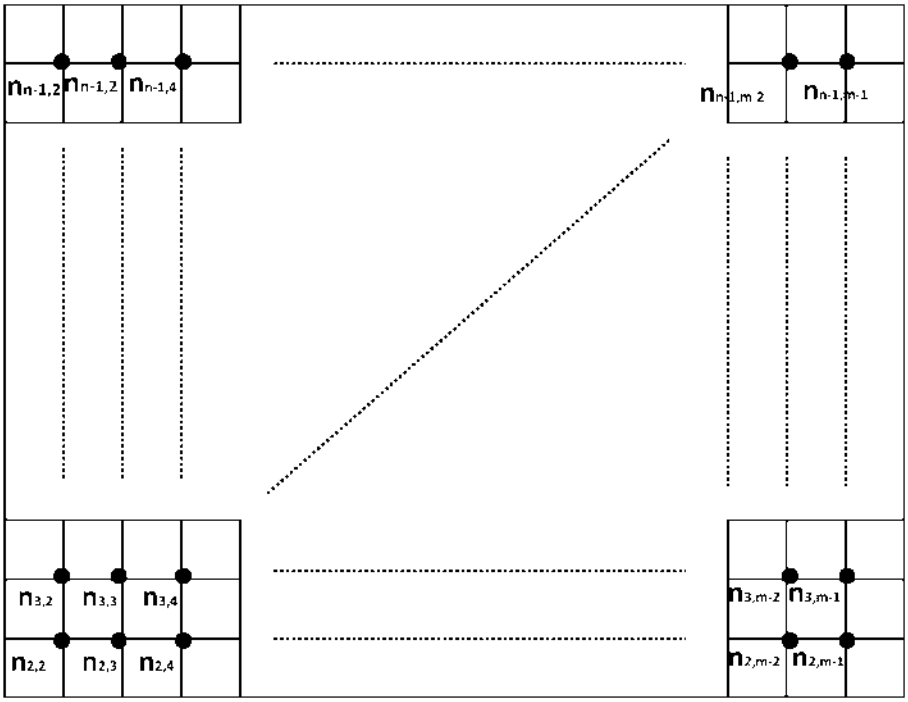

Figura 2.20: Nodos interiores de la malla.

Si la malla es de dimensiones $n \times m$, entonces se tendrán $(n-2)(m-2)$ nodos interiores, donde cada nodo tiene dos componentes, pues no hay que olvidar que los nodos en el presente problema son puntos del plano xy. Dado que es una malla quien juega el papel de vector x_k en cada iteración del proceso de optimización, entonces se debe encontrar una manera de representar una malla (como la que se muestra en la figura 2.19) en forma de un vector de $\mathbb{R}^n$. Puesto que los nodos de la malla fueron enumerados de la forma en que se muestra en 2.20 y cada nodo es un punto en el plano, es decir, cada nodo es tal que

$$n_{i,j} = \begin{pmatrix} x_{i,j} \\ y_{i,j} \end{pmatrix}. \tag{2.42}$$

Por tanto, si se concatenan los nodos siguiendo el orden mostrado en 2.20, entonces se podra representar la malla inicial en forma de un vector. Si la malla inicial representada como vector la definimos como x_0, entonces siguiendo las ideas planteadas para representar una malla como vector, se tiene que x_0

está dado por

$$x_0 = \begin{pmatrix} n_{2,2} \\ n_{2,3} \\ . \\ . \\ . \\ n_{2,m-1} \\ n_{3,1} \\ n_{3,2} \\ . \\ . \\ . \\ n_{3,m-1} \\ n_{4,1} \\ n_{4,2} \\ . \\ . \\ . \\ n_{n-2,m-1} \\ n_{n-1,m-1} \end{pmatrix} = \begin{pmatrix} x_{2,2} \\ y_{2,2} \\ x_{2,3} \\ y_{2,3} \\ . \\ . \\ . \\ x_{2,m-1} \\ y_{2,m-1} \\ x_{3,1} \\ y_{3,1} \\ x_{3,2} \\ y_{3,2} \\ . \\ . \\ . \\ x_{3,m-1} \\ y_{3,m-1} \\ x_{4,1} \\ y_{4,1} \\ x_{4,2} \\ y_{4,2} \\ . \\ . \\ . \\ x_{n-2,m-1} \\ y_{n-2,m-1} \\ x_{n-1,m-1} \\ y_{n-1,m-1} \end{pmatrix}. \tag{2.43}$$

Por tanto el vector x_0 tiene $2(n-2)(m-2)$ componentes, y son estos los valores que variarán para encontrar la malla óptima convexa G^*. En cada iteración del proceso de optimización se obtendrá un vector x_k que en realidad es una malla, cuando el proceso de optimización finalice y sea obtenido un vector x^*, entonces este vector representa en forma vectorial la malla optima G^*.

La forma inversa para obtener la malla G^* a partir del vector x_* es simplemente deshacer la representacion de malla a vector que se describió. Para llevar a cabo la optimización en la presente tesis, se utilizó en un inicio el método de Newton truncado con busqueda lineal, pero fue descartado despues de la implementación por la cantidad de recursos computacionales requeridos por este método. En la próxima sección se describen los métodos de optimización que fueron considerados y algunos implementados.

Si se tiene por ejemplo una imagen de resolución de 640X740 pixéles y se deseea obtener una malla adaptativa de 640X480 , esto implíca que se estará optimizando vectores que constan de $2(640-2)(480-2) = 609928$ variables, es decir, en cada iteración del proceso de optimización tan sólo para calcular el vector gradiente, se tendrá que calcular un vector de 609 928 componentes y aproximar una matriz Hessiana de dimensión de 609 928X609 928, lo cual repercute, entre otras cosas, en una cantidad considerable de memoria necesaria para el almacenamiento.

2.6. Métodos de optimización a gran escala

2.6.1. Busqueda lineal, condición de armijo y condiciones fuertes de Wolfe

En los últimos 40 años se han desarrollado algoritmos muy poderosos para problemas de optimización a gran escala. Estos algoritmos necesitan un punto inicial x_0 para comenzar la optimización, el cual debe ser seleccionado con cuidado dependiendo de la naturaleza del problema, los conceptos abordados en esta sección se muestran a profundidad en [8] y de una manera más básica en el apendice **B** de [4].

Como se mencionó anteriormente, a partir del punto inicial x_0 los algoritmos de descenso generan una secesión de puntos monotonamente decrecientes, donde el proceso termina al cumplirse un criterio establecido en base de una tolerancia, de igual manera, la condición y la tolerancia dependen del problema.

Pero hay un aspecto importante, pues una de las primeras preguntas que surgen al plantear un algoritmo de optimización es cómo moverse de la iteración inicial x_0 a la siguiente iteración x_1 a través de la función objetivo f. Existen dos estrategias hacer esto, estas dos estrategias son la **busqueda lineal** y **región de confianza**.

La busqueda lineal consiste en calcular un vector de descenso p_k en la iteración x_k, el cual es llamado vector de descenso, pues tiene la dirección máxima de decrecimiento de la función objetivo. Ya calculado el vector de descenso p_k en la iteración x_k, entonces ahora será en esta dirección en la cual se buscará la nueva iteración x_{k+1}.

Sin embargo, no se puede establecer que en general la nueva iteración sea $x_{k+1} = x_k + p_k$, sino que se debe calcular también un valor α_k mayor que cero llamado longitud de paso. La longitud de paso es necesaria pues para encontrar la nueva iteración x_{k+1}, en realidad se debe resolver el problema de minimización siguiente

$$\alpha_k = min_{\alpha>0} f(x_k + \alpha p_k),$$

Al encontrar α_k es como finalmente se encuentra la nueva iteración definida por

$$x_{k+1} = x_k + \alpha_k p_k.$$

Buscar α_k de manera casi continua para resolver el problema de minización anterior es muy costoso e innecesario. Por lo que al buscar α_k, se hace de manera más simple, esta manera consiste en buscar α_k con los valores $\alpha = 1$, $\alpha = 1/2$, $\alpha = 1/4$, ..., etcetera, hasta encontrar α_k.

Sucesivamente, se hace lo mismo con la nueva iteración para encontrar la siguiente. En cambio la estrategia de región de confianza es muy distinta, pues se usa información de la función objetivo f para formular una función modelo m_k, la cual tiene un comportamiento especial, pues el comportamiento de m_k cerca de la iteración x_k es muy similar al comportamiento de la función objetivo. Dado que la función m_k puede ser muy distinta a f mientras más lejos se esté del punto x_k, entonces se restringe la busqueda en una región para minimizar m_k.

Por tanto, en los métodos de busqueda lineal se calculará en cada iteración una dirección de descenso p_k y se decidirá que tando moverse en esa dirección mediante la longitud de paso α_k.

Por lo que la eficiencia de esta clase de métodos depende de la efectividad en que se calculen p_k y α_k. Se debe tener en cuenta de que en general no todos los métodos optimización de busqueda lineal requieren que p_k sea una dirección de descenso, sin embargo en los métodos que se abordan en este trabajo si debe ser de descenso.

La manera en que se calcula la dirección de descenso depende directamente de la clase de método que se esté usando, pues para los métodos de clase Newton p_k está dado por

$$p_k = -H(f)^{-1} \nabla f_k, \tag{2.44}$$

donde $H(f)$ es la matriz Hessiana exacta de f. Mientras que en los métodos Quasi-Newton no se utiliza H_k, sino una aproximación de la Hessiana denotada como B_k, la cual es una matriz simétrica y no singular que además es actualizada en cada iteración, por lo que el vector de descenso esta dado por

$$p_k = -B_k^{-1} \nabla f_k. \tag{2.45}$$

Se puede comprobar que si el vector de descenso se calcula por medio de 2.45 y si B_k es una matriz definida positíva, entonces estaŕa garantizado un descenso de la función f, en la dirección p_k, pues se tiene que

$$p_k^t \nabla f = -\nabla f^t (B_k^{-1})^t \nabla f,$$

dado que B_k es simétrica y definida positiva, entonces

$$p_k^t \nabla f = -\nabla f^t B_k^{-1} \nabla f < 0,$$

por lo que p_k es un vector de dirección de descenso.

Ahora, en cuanto a α_k, se busca (como en la mayoría de los métodos numéricos) poder obtener un α_k de manera eficiente y que tenga un efecto máximo en la reducción de la función objetivo f, lo cual se logrará si se encuentra el mínimo global de la siguiente función

$$\Phi(\alpha) = f(x_k + \alpha p_k) \quad \alpha > 0, \tag{2.46}$$

pero desde luego que esto en general es muy costoso y nada fácil de lograr. Por lo que la mejor opción es tratar de enfocarse en encontrar mínimos locales, trantando de evitar tantas evaluaciones de f como sean posibles, pues no hay que olvidar que en este trabajo se está tratando con problemas de optimización a gran escala.

Por tanto, para encontrar α_k se plantea lo siguiente, sea la función dada por 2.46, dado que en general es muy costoso encontrar de manera exacta α_k, entonces se opta por buscar la longitud de paso con estrategias inexactas, no olvidando que el objetivo es encontrar la longitud de paso que tenga un efecto de reducción significativo en f, por lo que lo usual es proponer valores de la longitud de paso hasta que satisfagan algunas condiciones que garanticen que la longitud de paso tenga un decrecimiento significativo en la función objetivo, es decir, que se tenga que

$$f(x_k + \alpha_k p_k) < f(x_k).$$

La expresión anterior es una primera y clara condición que deben cumplir α_k y p_k, sin embargo no es una condición suficiente, pues se podría dar el caso de que el decrecimiento sea muy lento e incluso que no haya convergencia, a continuación se muestra en la figura 2.21 una ilustración de este posible problema con una función objetivo de una variable.

Por tanto son necesarias mas condiciones que nos garanticen el que un valor de α_k es el adecuado como longitud de paso. Una condición muy común para la busqueda lineal de forma inexacta es la llamada condición de Armijo, la cual está dada por la siguiente desigualdad

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^t p_k. \tag{2.47}$$

Puede verse que la estrategia de busqueda lineal es más simple que la región de confianza, pero no es por esto que se elijió la estrategia de busqueda lineal, la razón por la que se optó por la busqueda lineal fue porque se emplearon métodos basados en la busqueda lineal.

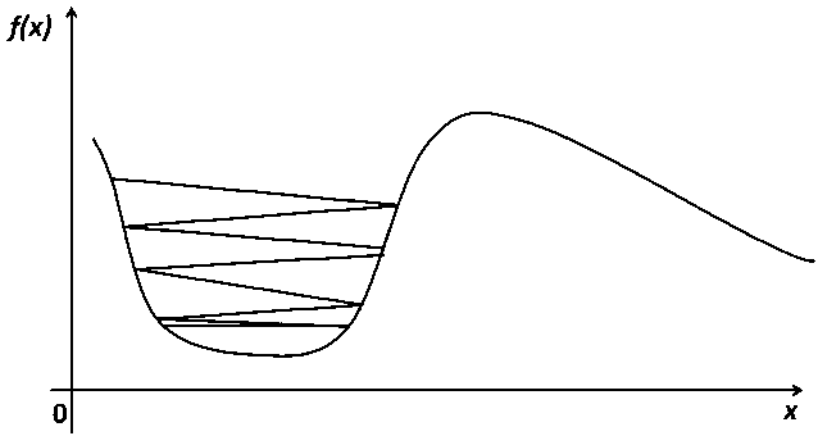

Figura 2.21: Caso en el que se puede presentar problemas con la convergencia.

La interpretación gráfica de la condición de Armijo se muestra la figura 2.22, donde pueden verse los intervalos que satisfacen cierta condición, donde $l(\alpha)$ denota la recta dada por el lado derecho de (2.47), es decir, las regiones que cumplen con la condición de Armijo son los intervalos donde la función $\Phi(\alpha) = f(x_k + \alpha p_k)$ queda dabajo de la recta

$$l(\alpha) = f(x_k) + c_1 \alpha \nabla f^t p_k.$$

Sin embargo, la condición de Armijo no es suficiente para asegurar una convergencia relativamente rápida, es por esto que para lidiar con esta problemática se considera la condición de curvatura junto con la condición de Armijo, a este par de desigualdades se le conoce como **condiciones de Wolfe**, la condición de curvatura está dada por

$$\nabla f(x_k + \alpha_k p_k)^t p_k \geq c_2 \nabla f_k^t p_k,$$

donde en la práctica $c_2 \in (c_1, 1)$, de igual manera, la figura 2.23 muestra la idea geométrica de la condición de curvatura. Al conjunto de desigualdades compuesto por la condición de Armijo y una versión modificada de la condición de curvatura, se les conoce como **las condiciones fuertes de Wolfe**, la longitud de paso α_k que cumpla estas condiciones generan un mayor decrecimiento, el siguiente par de expresiones muestra dichas condiciones

$$f(x_k + \alpha_k p_k) \leq f(x_k) + c_1 \alpha_k \nabla f_k^t p_k,$$

$$|\nabla f(x_k + \alpha_k p_k)^t p_k| \geq c_2 |\nabla f_k^t p_k|.$$

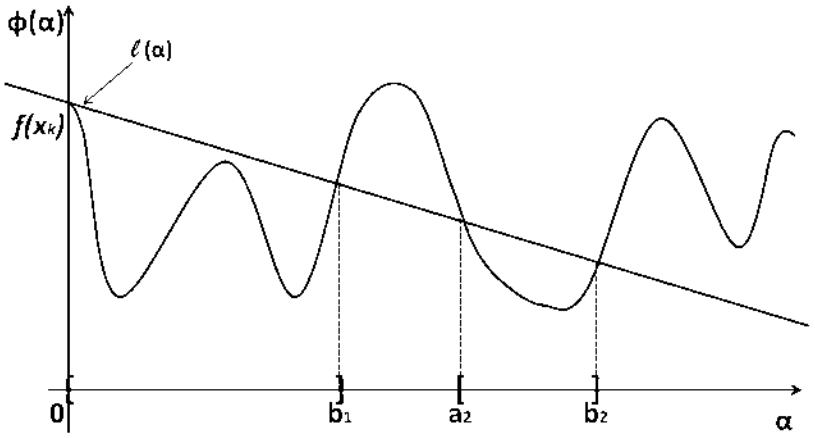

Figura 2.22: Intervalos adecuados que cumplen con la condición de Armijo.

2.6.2. Método BFGS

El método BFGS debe su nombre a las iniciales de sus creadores Broyden, Fletcher, Goldfarb y Shanno, siendo este método uno de los más populares de la clase de métodos Quasi-Newton, revisar el capítulo 6 de [8]. Este método consiste en lo siguiente, en cada iteración se formúla la función $m_k(p)$ dada por

$$m_k(p) = f_k + \nabla f_k^t p + \frac{1}{2} p^t B_k p.$$

Nótese que si $p = 0$ entonces $\nabla m_k(0) = \nabla f_k$. Hay que señalar que B_k es una aproximación de la matriz Hessiana, la cual es simétrica y positiva definida, que además es actualizada en cada iteración. El vector de descenso es el mínimo local de la función convexa cuadrática, por lo que se puede calcular como

$$p_k = -B_k^{-1} \nabla f_k,$$

por tanto, después de calcular la longitud de paso usando búsqueda lineal inexacta y las condiciones fuertes de Wolfe, se puede calcular la nueva iteración por medio de

$$x_{k+1} = x_k + \alpha_k p_k.$$

Al desarrollar la función $m_{k+1}(p)$, teniendo en cuenta que si $p = 0$ entonces $\nabla m_{k+1}(0) = \nabla f_{k+1}$ y calculando $\nabla m_{k+1}(\alpha_k p_k)$ se tiene que

$$\nabla m_{k+1}(\alpha_k p_k) = \nabla f_k + \alpha_k B_{k+1} p_k = \nabla f_{k+1},$$

por tanto

$$B_{k+1} \alpha_k p_k = \nabla f_{k+1} - \nabla f_k,$$

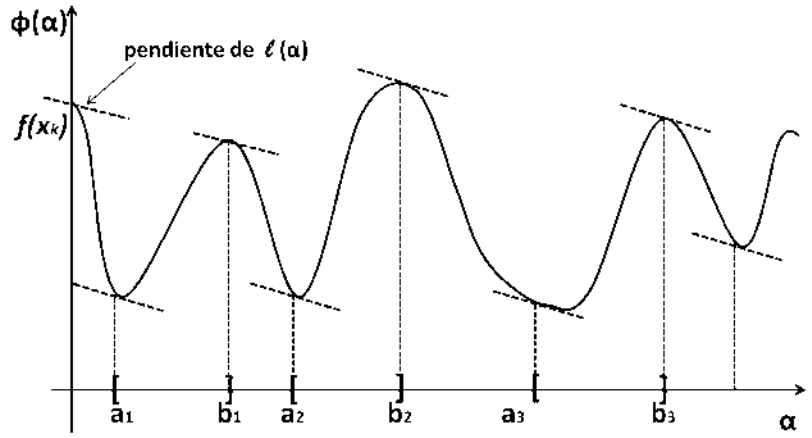

Figura 2.23: La condición de curvatura.

para simplificar se definen dos variables que serán muy útiles más adelante, las cuales son s_k y y_k y que están dadas por

$$s_k = x_{k+1} - x_k, \qquad y_k = \nabla f_{k+1} - \nabla f_k, \tag{2.48}$$

sustituyendo se tiene que

$$B_{k+1}s_k = y_k.$$

Esta última expresión es llamada ecuación secante y que al multiplicar por la izquierda por s_k^t, entonces

$$s_k^t B_{k+1} s_k = s_k^t y_k,$$

como B_{k+1} es definida positiva, esto implica que

$$s_k^t y_k > 0. \tag{2.49}$$

La cual es la condición de curvatura, esta condición se puede visualizar en la figura 2.23. Para actualizar la matriz B_k en cada iteración, se hace por medio de la siguiente formula

$$B_{k+1} = (I - \rho_k y_k s_k^t) B_k (I - \rho_k s_k y_k^t) + \rho_k y_k y_k^t, \tag{2.50}$$

donde

$$\rho_k = \frac{1}{y_k^t s_k}. \tag{2.51}$$

Puesto que para calcular el vector de descenso p_k se necesita la aproximación de la inversa de Hessiana, esta aproximación se puede calcular como

$$B_{k+1}^{-1} = B_k^{-1} - \frac{B_k^{-1} y_k y_k^t B_k^{-1}}{y_k^t B_k^{-1} y_k} + \frac{s_k s_k^t}{y_k^t s_k}, \tag{2.52}$$

por lo que se tiene la siguiente formula

$$B_{k+1}^{-1} = (I - \rho_k s_k y_k^t) B_k^t (I - \rho_k y_k s_k^t) + \rho_k s_k s_k^t, \tag{2.53}$$

donde ρ_k esta dada por (2.51), la figura 2.24 muestra el pseudocódigo del método BFGS.La aproximación inicial de la inversa de la Hessiana se puede calcular por medio de

$$B_0^{-1} = \frac{y_k^t s_k}{y_k^t y_k} I.$$

Dado un punto inicial x_0, luna aproximación inicial de la matriz Hessiana H_0, además de una tolerancia E tal que E>0, entonces

Mientras $\|g_k\|>E$
- **Calcular p_k tal que**
  - **$p_k=-H_k*g_k$**
- **Establecer a=1**
- **Establecer $x_{k+1}=x_k+a*p_k$**
- **Si no se cumplen las condiciones de Wolfe, entonces**
  - **a=a/2**
  - **Mientras no se cumplan las condiciones de Wolfe**
    - **Establecer $x_{k+1}=x_k+a*p_k$**
    - **a=a/2;**
    - **Calcular g_{k+1}**
    - **Calcular f_{k+1}**
- **Una vez que x_{k+1} cumple las condiciones de Wolfe, entonces calcular g_k y s_k.**
- **Actualizar la aproximaci\'on de la matriz Hessiana H_{k+1}**

Figura 2.24: Pseudocodigo del método BFGS.

En la expresión (2.52) se puede ver la ventaja que tiene el método BFGS, pues en lugar de calcular desde cero la aproximación de la inversa de la Hessiana, solamente se modifica la aproximación ya existente.

El método BFGS no fue conveniente usarlo, pues se tuvieron limitantes computacionales como pasó con el método de Newton Truncado.

Si se optimizará una malla de 400×400, entonces los vectores obtenidos en cada iteración, así como el gradiente serían de dimensión 316808×1 y la matriz Hessiana (o su aproximación) tendría dimensión de 316808×316808, si se toma en cuenta de que en Matlab los números son de tipo double y que cada uno ocupa 64 bits, entonces, el principal problema está en el almacenamiento

de la Hessiana, pues se necesitan $6{,}4235^{12}$ bits para poder almacenar la Hessiana.

Teniendo en cuenta de que

$$1bit = 1{,}25^{-10}GB,$$

entonces

$$(6{,}4235^{12}bit)\left(\frac{1{,}25^{-10}GB}{1bit}\right) = 802{,}9375GB.$$

Lo que significa que tan sólo para almacenar la Hessiana en una iteración se necesita 802.9375 GB de memoria, por lo que dada la naturaleza del problema de la presente tesis, no es posible usar los métodos de clase Newton ni el método BFGS. Pero no todo está perdido, pues en la siguiente sección se muestra el método LBFGS, el cual es un método que si fue posible usar, pues es el método BFGS modificado para usar un mínimo de memoria, este método también es llamado BFGS de memoria limitada.

2.6.3. Método LBFGS

Este método consiste en el método BFGS limitando el espacio de almacenamiento, de ahí sus siglas, revisar sección 7,2 de [8]. Como se vio en la parte final de la sección anterior, no fue conveniente usar el método BFGS por las exigencias de recursos computacionales que hacían imposible su implementación.

Esto no hace que haya sido inútil la examinación del método BFGS, pues es precisamente el método BFGS el antecesor del método LBFGS, es decir, las bases del método LBFGS son las mismas que las del método BFGS. El método LBFGS es ideal para problemas de optimización a gran escala donde el cálculo de la aproximación de la Hessiana es muy costoso computacionalmente, pues corresponde a la clase de métodos Quasi-Newton de memoria limitada.

El método LBFGS no almacena aproximaciones anteriores de la Hessiana, en lugar de eso sólo usa un número m de vectores de iteraciones anteriores que contienen información de la curvatura, la cual es información importante y representativa matriz Hessiana exacta, a los m vectores que se almacenan y que cambian en cada iteración se les llama **parejas de corrección**.

La idea principal del LBFGS es usar las parejas de corrección para actualizar la aproximación de la matriz inversa de la Hessiana exacta, esta idea logra reducir sustancialmente el espacio de almacenamiento. Sea Bl_k la aproximación de la matriz de la Hessiana, a diferencia del método BFGS, en el LBFGS se usa una aproximación de la inversa de Bl_k, es decir, el método usa Bl_k^{-1} en lugar de Bl_k.

En el método LBFGS, el vector de descenso se calcula de la siguiente manera

$$p_k = -Bl_k^{-1}\nabla f_k. \tag{2.54}$$

Al igual que en el método BFGS, en el método LBFGS se emplea busqueda lineal en el vector de descenso por medio de una longitud paso, en otras palabras, una vez obtenido el vector de descenso p_k y la longitud de paso óptima α_k, la siguiente iteración está dada por

$$x_{k+1} = x_k + \alpha_k p_k.$$

La fórmula para calcular Bl_k^{-1} es la siguiente

$$Bl_{k+1}^{-1} = (I - \rho_k s_k y_k^t) Bl_k^{-1} (I - \rho_k y_k s_k^t) + \rho_k s_k s_k^t, \tag{2.55}$$

donde ρ_k y las parejas de corrección s_k y y_k estan dados por (2.51) y (2.48), respectivamente. En la práctica, el número m de parejas de corrección que se almacenan oscila entre 3 y 19. Se denota a cada pareja de corrección como$\{s_i, y_i\}$, la actualización de Bl_{k+1}^{-1} en cada iteración haciendo uso de las m parejas de correción sólo requiere del cálculo de algunos productos punto y suma de vectores, además de usar ∇f_k. Sin embargo, la primera interrogante que surge es acerca de como seleccionar las parejas de correción y que par de vectores formarian la primera pareja de corrección en la iteración inicial.

La respuesta es la siguiente, en la iteración inicial no se tiene ninguna pareja de corrección, pues por ejemplo, no se dispone del gradiente ∇f_{k+1} para el caso de y_k. Despues realizar la primera iteración es que por fin se conoce la primera pareja de corrección, al terminar la siguiente iteración se almacen la segunda pareja de corrección, esto se hace hasta acumular m parejas de corrección. Una vez obtenidas las m parejas de corrección, en cada iteración siguiente se almacena la nueva pareja de corrección y se descarta la pareja más antigua, manteniendo así un número fijo m de parejas de corrección.

Es importanate agregar que la actualización de Bl_{k+1}^{-1} en cada iteración, se hace usando las parejas de corrección disponibles. El LBFGS requiere de una aproximación inicial Bl_{k+1}^{-1}, en la presente tesis se usó como aproximación inicial, la matriz $Bl_{0^{-1}}$ que consiste sólo de los elementos de la diagonal de matriz Hessiana exacta.

Para calcular el vector de descenso p_k, se usa un proceso recursivo basado en (2.55) y (2.54), este procedimiento usa las parejas de correción disponibles. La figura 2.25 se muestra el pseudocódigo del algoritmo recursivo para calcular el vector de descenso.

En la figura 2.25 se muestra el pseudocodigo para calcular el vector p_k de manera recursiva, usando las parejas de corrección disponibles.

La figura 2.26 muestra el pseudocódigo del método LBFGS. Puesto que las mallas adaptativas se escalan al cuadrado unitario, entonces entre mayor sea la

resolución de la imagen, menos se podrán distinguir las celdas elongadas que aparecen cerca o en los bordes. Es por esto que se hace uso de la función de distorsión, la cual es aplicada en cada celda de la malla y el valor resultante se asigna al centroide de la celda, la función de distorsion está dada por la siguiente expresión

$$F(\lambda) = \lambda \frac{L_M}{l_m} + (1-\lambda)\frac{D_M}{d_m}, \qquad \lambda \in [0,1], \tag{2.56}$$

donde L_M, l_m, D_M y d_m son los valores de las longitudes del lado mayor, lado menor, diagonal mayor y diagonal menor, respectivamente, en cada celda.

Es importante mencionar que en la implementación del método LBFGS fue necesario agregar una condición adicional a la longitud de paso α_k, pues en cada iteración del proceso de optimización se generan p_k y α_k, donde en el nuevo vector $x_{k+1} = x_k + \alpha_k p_k$ a pesar que α_k cumpla con las condiciones fuertes de Wolfe, puede pasar que algunos nodos se salgan del cuadrado unitario. Hay que recordar que la malla adaptativa generada se escala al cuadrado unitario, por lo que si durante el proceso de optimización existen nodos que se salen del cuadrado unitario, es muy probable que el proceso para generar la malla adaptativa falle, por lo que la condición adicional que se agrego en la presente tesis fue que α_k sea tal que $x_{k+1} = x_k + \alpha_k p_k$ esté acotado por el cuedrado unitario. esta condicion es una clase de acotamiento a los nodos de la malla adaptativa.

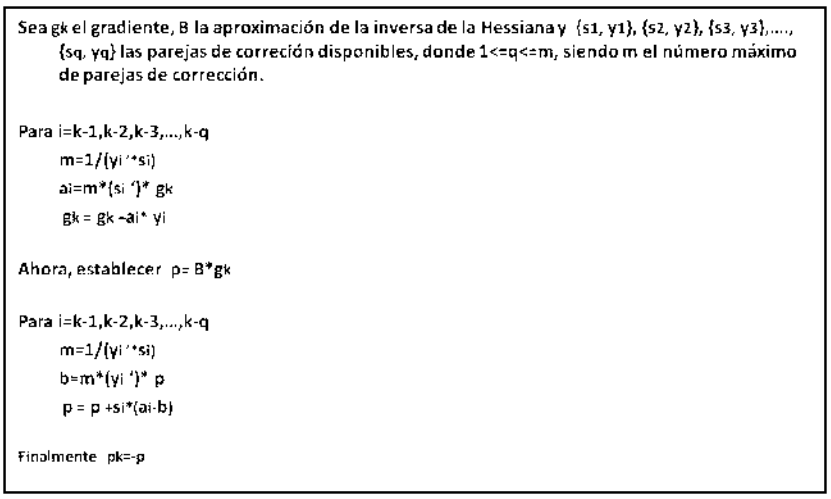

Figura 2.25: Pseudocodigo para calcular el producto $-Bl_k^{-1}\nabla f_k$ y así obtener p_k.

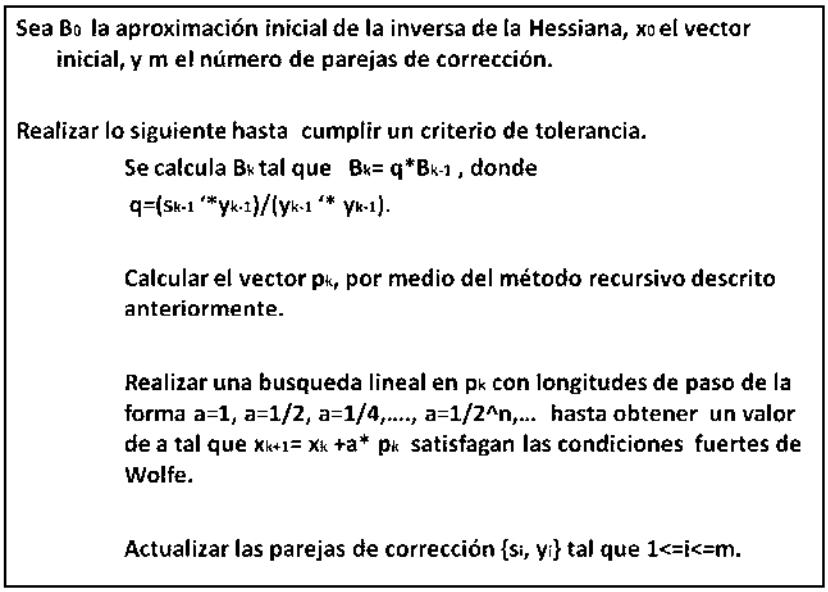

Figura 2.26: Pseudocodigo del método LBFGS.

2.6.4. Método LBFGS-B

El método LBFGS-B es una variación del método LBFGS, pues el método LBFGS-B esta diseñado para la solución de problemas no lineales de optimización a gran escala con reestricciones. Este método desarrollado por Jorge Nocedal y su equipo [7], está basado en el método de proyección del gradiente y LBFGS para la aproximación de la Hessiana, además de otras modificaciones para hacer el algoritmo más eficiente como lo son el empleo de matrices compactas. Este método tambien se empleo por la problematica de recursos computacionales y para compararlo con el método LBFGS en cuanto a tiempo de ejecución se refiere.

Sea el problema no lineal de optimización a gran escala con reestricciones siguiente

$$\begin{aligned} min \quad & f(\vec{x}) \\ sujeto \quad a \quad & \vec{l} \leq \vec{x} \leq \vec{u}, \end{aligned}$$

donde f es una función no lineal tal que $f : \mathbb{R}^n \to \mathbb{R}$, de la cual se conoce en principio su gradiente g, las cotas inferior y superior, $\vec{l}$ y $\vec{u}$, respectivamente, tal que $\vec{l}, \vec{u} \in \mathbb{R}^n$. Una característica importante del LBFGS-B es que no se necesita conocer información de las segundas derivadas o información de la las propiedades de la función, como lo es su estructura.

En el método LBFGS-B al igual que en BFGS, se actualiza la aproximación de la matriz Hessiana que se denotará por Blb_k, además se emplea búsqueda lineal. En el LBFGS-B también se emplean las parejas de corrección, como se hizo en el LBFGS, pero a diferencia del LBFGS, las parejas de correción se estructuran en matrices y son sometidas a cumplir con ciertas condiciones para ser almacenadas. Hay que recordar que las parejas de corrección de la $k-esima$ iteración son

$$s_k = x_{k+1} - x_k, \qquad y_k = g_{k+1} - g_k.$$

Las parejas de corrección se emplean para la construcción de las llamadas matrices BFGS de memoria limitada. Sea m el número de parejas de corrección a considerar, recordando que en la práctica $3 \leq m \leq 20$, entonces surge una interrogante importante, la cual consiste en como representar las matrices BFGS de memoria limitada usando las parejas de corrección sin formarlas y almacenarlas, Nocedal, Byrd y Schnabel optaron por usar la forma compacta de representación para las matrices BFGS de memoria limitada, por tanto se define el par de matrices S_k y Y_k tal que

$$\begin{aligned} S_k &= [s_{k-m}|s_{k-(m-1)}|s_{k-(m-2)}|, ..., |s_{k-1}], \\ Y_k &= [y_{k-m}|y_{k-(m-1)}|y_{k-(m-2)}|, ..., |y_{k-1}], \end{aligned} \tag{2.57}$$

donde las parejas de corrección $\{s_i, y_i\}$ satisfacen que $s_i^t y_k > 0$, por tanto usando una variación de la formula del método BFGS y las matrices 2.57, se tiene que la actualización de la aproximación de la matriz Hessiana está dada por

$$Blb_k = aI - W_k M_k W_k^t, \tag{2.58}$$

donde $a \in \mathbb{R}$, $a > 0$ y

$$W_k = [Y_k|aS_k], \tag{2.59}$$

$$M_k = \begin{bmatrix} -D_k & L_k^t \\ L_k & aS_k^t S_k \end{bmatrix}^{-1}, \tag{2.60}$$

y finalmente las matrices L_k y D_k de dimensiones $m \times m$ son tal que

$$(L_k)_{i,j} = \begin{cases} (s_{k-m-1+i})^t (y_{k-m-1+i}) & si \quad i > j \\ 0 & en \quad otro \quad caso \end{cases} \tag{2.61}$$

$$D_k = diag([s_{k-m}^t y_{k-m}, s_{k-(m-1)}^t y_{k-(m-1)}, s_{k-(m-2)}^t y_{k-(m-2)}, ..., s_{k-1}^t y_{k-1}]). \tag{2.62}$$

Nótese que las matrices L_k y D_k son matrices de $m \times m$, por lo que M_k es de $2m \times 2m$ y dado que W_k es de $n \times 2m$, entonces efectivamente se puede comprobar que Blb_k es de $n \times n$. Hay que recalcar que m es un natural pequeo,

entonces el costo de calcular M_k^{-1} es considerado despreciable.

Ahora, en cada iteración se dispone del gradiente g_k, de la aproximación de la Hessiana Blb_k y un punto $\vec{x_k}$, pues en la primera iteración como no se tiene ninguna pareja de corrección, entonces se utiliza $Blb = aI$ para calcular la aproximación. El trasfondo del método LBFGS-B es el siguiente, primero se forma el modelo dado por la función m_k tal que

$$m_k(\vec{x}) = f(\vec{x}_k) + g_k^t(\vec{x} - \vec{x}_k) + \frac{1}{2}(\vec{x} - \vec{x}_k)^t Blb_k(\vec{x} - \vec{x}_k). \tag{2.63}$$

La meta es minimizar 2.63 tomando en cuenta las restricciones $\vec{l}$ y $\vec{u}$ del problema inicial, esta minimización se hace por medio del método de proyección del gradiente, para así encontrar las cotas activas, posteriormente la minimización de m_k se hace tratando a las cotas activas como restricciones dadas por igualdades. Por tanto con la dirección de máximo descenso en la región factible se obtiene la trayectoria lineal a trozos dada por la siguiente función

$$x(t) = P(\vec{x}_k - tg_k, l, u), \tag{2.64}$$

donde

$$P(\vec{x}, l, u)_i = \begin{cases} l_i & si \quad \vec{x}_i < l_i \\ u_i & si \quad \vec{x}_i > u_i \\ \vec{x}_i & si \quad \vec{x}_i \in [l_i, u_i] \end{cases} . \tag{2.65}$$

Es por esto que se calcula el punto generalizado de Cauchy $\vec{x}^c$, el cual está definido como el mínimo local de 2.63 usando 2.64, es decir, el punto generalizado de Cauchy se define como el mínimo local de la función q_k definida por la siguiente expresión

$$q_k(t) = m_k(x(t)).$$

Se define el conjunto $A(\vec{x}^c)$ como el conjunto que tiene como elementos los indices de las variables cuyo valor en el punto de Cauchy generalizado es igual a una cota inferior o superior. A las variables que no son activas se llaman variables libres. Por tanto, ahora sólo se considera el problema de minimización (cuadrático) de las variables libres, el cual está definido por el siguiente problema de optimización

$$\begin{aligned} &min\{m_k(\vec{x}) : x_i = x_i^c, \quad \forall i \in A(x^c)\} \\ &sujeto \quad a \quad l_i \leq x_i \leq u_i \quad i\forall \notin A(x^c). \end{aligned} \tag{2.66}$$

El problema 2.66 se resuleve empleando gradiente conjugado o por medio del métod primal directo. Al optar por un método iterativo como gradiente conjugado, se utiliza x^c como punto inicial. Al encontrar una solución $\vec{x}_{p,k+1}$ del problema 2.66, entonces se calcula el vector de descenso p_k dado por

$$p_k = \vec{x}_{p,k+1} - \vec{x}_k,$$

donde finalmente sólo queda hacer una búsqueda lineal a través de p_k para encontrar $\vec{x}_{k+1}$, teniendo en cuenta que $\vec{x}_{k+1}$ debe satisfacer las siguientes condiciones

$$\begin{aligned} f(\vec{x}_{k+1}) &\leq f(\vec{x}_k) + c_1\alpha_k g_k^t p_k \\ |g_{k+1}^t p_k| &\leq c_2 |g_k^t p_k|, \end{aligned} \tag{2.67}$$

donde α_k es la longitud de paso resultante de la búsqueda lineal hecha con x_k y p_k. Se usan valores $c_1 = 10^{-4}$ y $c_2 = 0{,}9$, y al igual que en el método BFGS, se debe tener que Blb_k sea positiva definida para que p_k sea una dirección de descenso, pues hay que recordar que el punto de Cauchy es tal que

$$m_k(\vec{x}_k) > m_k(x^c),$$

dado que

$$m_k(\vec{x}_k) = f(\vec{x}_k),$$

entonces

$$f(\vec{x}_k) > m_k(x^c).$$

Como la solución aproximada $\vec{x}_{p,k+1}$ se calcula en una trayectoria que parte de x^c y en la cual la funcion m_k decrece, entonces

$$m_k(x^c) \leq m_k(\vec{x}_{p,k+1}),$$

por lo que las últimas dos expresiones implican que

$$f(\vec{x}_k) > m_k(\vec{x}_{p,k+1}),$$

puesto que

$$\begin{aligned} m_k(\vec{x}_{p,k+1}) &= f(\vec{x}_k) + g_k^t(\vec{x}_{p,k+1} - \vec{x}_k) + \frac{1}{2}(\vec{x}_{p,k+1} - \vec{x}_k)^t Blb_k(\vec{x}_{p,k+1} - \vec{x}_k) \\ m_k(\vec{x}_{p,k+1}) &= f(\vec{x}_k) + g_k^t p_k + \frac{1}{2} p_k^t Blb_k p_k, \end{aligned} \tag{2.68}$$

por tanto, al observar las ultimas dos expresiones, se tiene que

$$\begin{aligned} f(\vec{x}_k) &> f(\vec{x}_k) + g_k^t p_k + \tfrac{1}{2} p_k^t Blb_k p_k \\ \Rightarrow 0 &> g_k^t p_k + \tfrac{1}{2} p_k^t Blb_k p_k \\ \Rightarrow -\tfrac{1}{2} p_k^t Blb_k p_k &> g_k^t p_k, \end{aligned}$$

si Blb_k es definida positiva se tiene finalmente que

$$g_k^t p_k < 0,$$

comprobandose así que p_k es dirección de descenso.

Capítulo 3

Diferencias finitas generalizadas

3.1. Introducción

En general, en muchos aplicaciones, sólo se puede aspirar a encontrar una aproximación de la solución de la ecuaión diferencial (ED) que las modela, pues al tratar con un problema con todas las condiciones reales que están involucradas, el problema deja de estár planteado de forma idealizada.

Es por esto que surgen los métodos numéricos, un camino para obtener una solución aproximada. Afortunadamente, en la actualidad se dispone de muchos métodos numéricos en la literatura, lo cual se ve reflejado en la tecnología actual.

Sin embargo, también es importante mencionar que al resolver un problema numéricamente, no sólo se debe lidiar con la complejidad del problema, sino que tambiém surgen problemas adicionales, tales como las limitaciones fisícas de los equipos de computo, pues aunque actualmente las computadoras tienen gran potencial, en ocasiones no es suficiente.

En esta capítulo se aborda cómo aproximar la solución de una ecuación diferencial por medio de diferencias finitas, las cuales son muy usadas, en esta tesis se decidió emplearlas por su sencillez en su formulación y aplicación. Un aspecto importante de las diferencias finitas es que se deben establecer ciertas propiedades en la frontera del dominio donde se pretende aproximar la EDP.

Las propiedades que se establecen en la frontera Ω son llamadas **condiciones de frontera**. Las condiciones en la frontera juegan un papel complementario a la **condición inicial** en un problema de evolución, es decir, son fundamentales para obtener las constantes involucradas en el problema de aproximar una EDP, así como para establecer restricciones en la frontera.

Por último, es importante señalar que las diferencias finitas tienen como trasfondo el pasar de un problema continuo de una EDP a un problema discreto, aproximando la solución de la EDP en un número finito de puntos, que desde luego pertenecen al dominio donde se pretende obtener la solución aproximada. Es por esto que el primer paso para emplear diferencias finitas consiste en discretizar el dominio; a la discretización del dominio se le denomina **establecer una geometría en el dominio**.

3.2. Mallado del dominio

3.2.1. Planteamiento

La forma en que se discretiza el dominio para aproximar una EDP es un aspecto crucial para la precisión de las diferencias finitas. Es por esto que se ha desarrollado toda una teoría de generación de mallas, en la cual se han diseñado distintos métodos para la obtención de mallas. Aunque en el capítulo anterior se describió la teoría de generación de mallas, en esta sección se abordará un poco el tema desde una perspectíva más geométrica.

Una malla creada en un dominio (la región donde se aproxima la EDP) no es más que un conjunto finito de puntos del dominio, a los que les llama **nodos**, los nodos son obtenidos al discretizar el dominio, este conjunto pueden tener o no una estructura especifica. Las mallas pueden ser generadas con distintas características y es en los nodos y aristas donde se aproxima la EDP.

Existen distintos métodos para la generación de mallas, en la sección 1.,4,1 de [2] se muestra una descripcción de los siguientes métodos:

- Métodos algebraicos.
- Métodos elípticos.
- Métodos hiperbólicos.
- Métodos variacionales.

3.2.2. Mallas estructuradas y no estructuradas

Las mallas se pueden clasificar en dos grupos, las mallas estructuradas y las no estructuradas. Una malla puede estar constituida por elementos geometricos facilmente manipulables. La mallas que son estructuradas son llamadas así porque se establece que los nodos tienen toda una estructura lógicamente rectángular.

Para entender un poco mejor que es la estructura de una malla, sirve de ayuda establecer una analogía: las matrices de dimensiones de $n \times m$ con entradas reales tienen una estructura rectangular, pues una matriz está compuesta por un conjunto de valores numéricos ordenados rectangularmente por renglones o por columnas, estos elementos son llamados entradas de la matriz. La estructura de la matriz está dada por un par de índices, que son utilizados para indicar en que renglón y en que columna está una entrada específica, dandole así una forma rectangular al ordenar las entradas por medio de los índices.

En el caso de una malla estructurada y rectangular en 2D, la estructura consiste en que los nodos se dividen en varios grupos, es decir, la malla tiene nodos que están en la frontera y nodos que están en el interior del dominio, los nodos frontera tienen cinco nodos vecinos, pero entre los nodos frontera están los nodos esquina, que a diferencia de los anteriores, sólo tienen tres nodos vecinos. En cambio todo nodo interior tienen ocho nodos vecinos; las figuras 3.1 y 3.2 muestran una malla rectangular estructurada y una malla no estructurada, respectivamente, se puede apreciar que en el caso de la malla no estructurada, no se puede establecer, por ejemplo, que existe alguna clase de nodos con un número fijo de nodos vecinos.

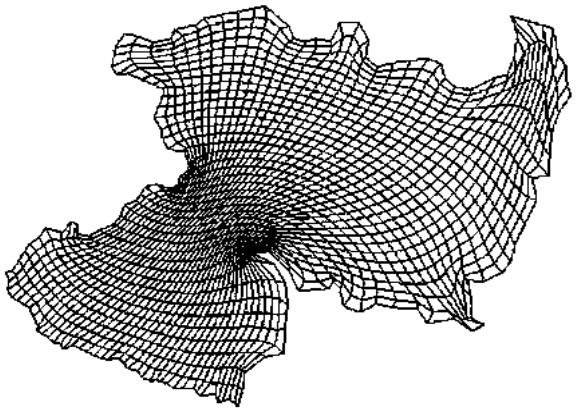

Figura 3.1: Una malla estructurada compuesta por cuadriláteros.

Aunque formalmente una malla se define como una función, en la práctica y para este capítulo se puede entender una malla como un conjunto de nodos que discretizan una región. En este trabajo sólo se usaron mallas lógicamente estructuradas rectangulares. Se pueden generar mallas estructuradas en una dimensión. Por ejemplo, considérese el intervalo $[a, b]$, haciendo una partición arbitraria del intervalo en n subintervalos, se obtien una sucesión de $n+1$ nodos $x_0, x_1, x_2, x_3, \ldots, x_n$, donde $a = x_0$ y $b = x_n$; la figura 3.3 muestra una ilustración de este caso.

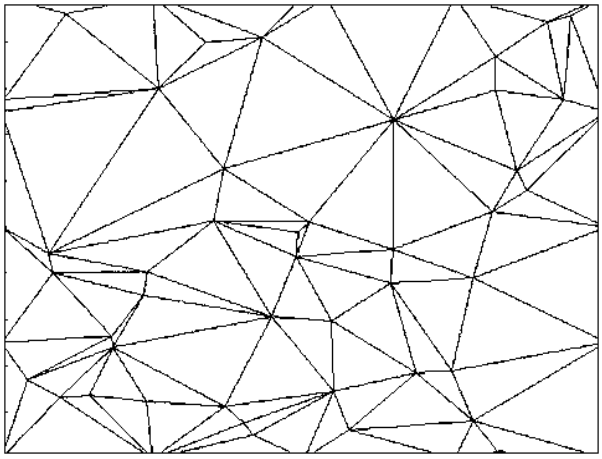

Figura 3.2: Una malla no estructurada, en este caso una triangulación.

Nótese que no es necesario que la distancia entre los nodos sea constante. La estructura en este ejemplo está dada por la indexación entre los nodos, es decir, $x_0 \leq x_1 \leq x_2 \leq x_3 \leq \ldots \leq x_n$, además los dos nodos frontera x_0 y x_n sólo tienen un nodo vecino, el resto de los nodos tienen dos vecinos.

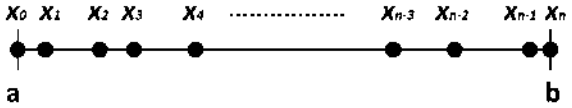

Figura 3.3: Una malla estructurada en 1D.

En el caso de mallas estructuradas rectangulares en 2D surge la problemática de que la frontera del dominio puede ser muy irregular, lo que produce mallas con cuadriláteros muy irregulares, en el sentido de que se producen cuadriláteros distorsionados, las figuras 3.4, 3.5, 3.6 muestran mallas en donde cada dominio es cada vez más irregular.

3.2.3. Diferencias finitas en 1D y 2D

En esta sección se describe la deducción de las diferencias finitas [14] en una y dos dimensiones, pues es útil saber el origen de esta herramienta matemática, la cual se distingue por su sencillez y elegancia.

En el caso de una dimensión, sea $f(x)$ una función de una variable real tal que $f : \mathbb{R} \to \mathbb{R}$, si f es diferenciable en x_0, entonces la derivada de f en x_0

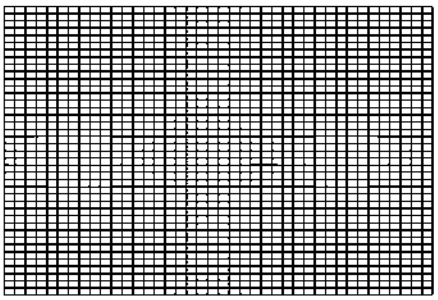

Figura 3.4: Una malla rectangular estructurada en una región con frontera rectangular.

está definida por

$$f^{'}(x_0) = \lim_{\Delta x \to 0} \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x},$$

la derivada de f en x_0 es el valor de la pendiente de la recta tangente a la gráfica de la función $f(x)$ en x_0.

Las diferencias finitas tienen su origen en el Teorema de Taylor: si f es una función continua en $[a, b]$ y diferenciable n veces en (a, b) y $x_0, x \in (a, b)$, donde $x = x_0 + \Delta x$, entonces por el teorema de Taylor se tiene que

$$f(x) = f(x_0) + \Delta x f^{'}(x_0) + \frac{\Delta x^2 f^{''}(x_0)}{2!} + \ldots + \frac{\Delta x^n f^{(n)}(x_0)}{n!} + R_n(f), \quad (3.1)$$

donde $R_n(f)$ es el residuo de f.

Por tanto, del teorema de Taylor y tomando en cuenta que $x = x_0 + \Delta x$, es posible expresar la ecuación 3.1 truncando a partir de la segunda derivada usando la notación $O(\cdot)$, entonces

$$\begin{aligned} f(x_0 + \Delta x) &= f(x_0) + \Delta x f^{'}(x_0) + O(\Delta x^2) \\ \Rightarrow \quad f(x_0 + \Delta x) - f(x_0) &= \Delta x f^{'}(x_0) + O(\Delta x^2) \\ \Rightarrow \quad \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} &= f^{'}(x_0) + O(\Delta x). \end{aligned} \quad (3.2)$$

Ahora, reacomodando la expresión anterior se tiene

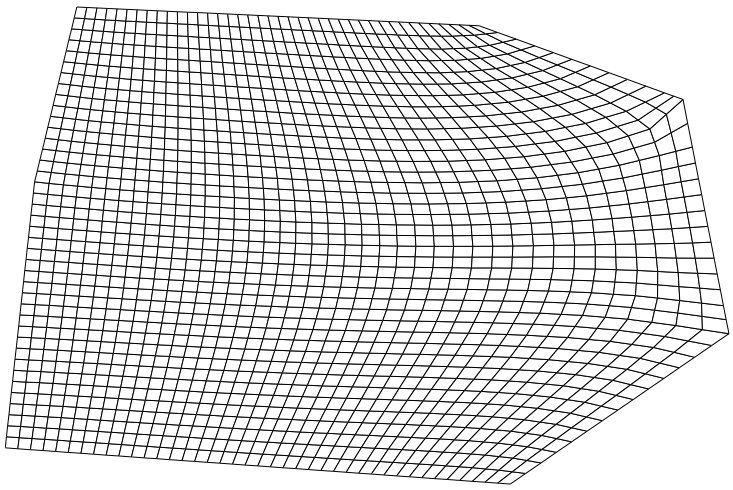

Figura 3.5: Una malla rectangular estructurada en una región un poco irregular.

$$f^{'}(x_0) \approx \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}.$$

Esta última expresión es precisamente la aproximación por diferencias finitas de la derivada de f hacia adelante (por la derecha), pues entre menor sea Δx, entonces x se acerca mas a x_0 por la derecha y se obtiene una mejor aproximación a la derivada como se puede ver en la figura 3.7, por tanto si se denota la aproximación en diferencias finitas de f por la derecha como f_d, entonces f_d está dada por

$$f_d(x_0) = \frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x}. \tag{3.3}$$

De manera análoga se deduce la aproximación en diferencias finitas por la izquierda, solamente tomando ahora $x = x_0 - \Delta x$. La aproximación de la derivada de f en x_0 por la izquierda denotada por f_i está dada por la expresión

$$f_i(x_0) = \frac{f(x_0) - f(x_0 - \Delta x)}{\Delta x}. \tag{3.4}$$

No solamente están las aproximaciones por la derecha o por la izquierda, también se puede formular una aproximacón centrada en x_0; esta aproximación se denota por f_c y está definida por el promedio de las aproximaciones por la derecha y por la izquierda dadas por. Por tanto, f_c está dada por la siguiente expresión

$$f_c(x_0) = \frac{1}{2}\left(\frac{f(x_0 + \Delta x) - f(x_0)}{\Delta x} + \frac{f(x_0 - \Delta x) - f(x_0)}{\Delta x}\right),$$

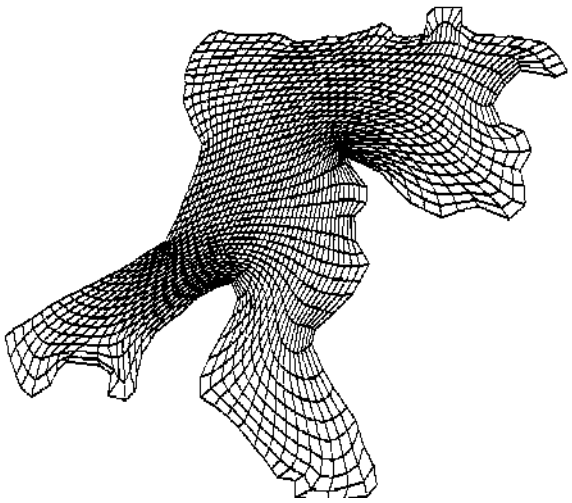

Figura 3.6: Una malla rectangular estructurada en una región irregular.

$$\Rightarrow f_c(x_0) = \frac{f(x_0 + \Delta x) - f(x_0 - \Delta x)}{2\Delta x}. \tag{3.5}$$

La imagen 3.8 muestra las respectivas aproximaciones por diferencias finitas que se han descrito hasta ahora, es decir, se muestra las aproximaciones de la derivada de una función f por la derecha, por la izquierda y centrada.

Para deducir formulas para la segunda derivada (ya sea por la derecha, izquierda o centrada) basta con elegir con que términos se desea obtener la formula, por ejemplo, a continuación se mostrará como obtener la formula de la segunda derivada de f en diferencias finitas centradas, esta aproximación se denotará como f_c^2

$$f_c^2(x_0) = af(x_0 + \Delta x) + bf(x_0) + cf(x_0 - \Delta x), \tag{3.6}$$

entonces, usando el teorema de Taylor para expandir $f(x_0 + \Delta x)$ y $f(x_0 - \Delta x)$, se tiene lo siguiente

$$\begin{aligned} f_c^2(x_0) = \quad & a\left[f(x_0) + \Delta x f^{'}(x_0) + \tfrac{\Delta x^2 f^{''}(x_0)}{2!} + \ldots\right] + bf(x_0) + \\ & + c\left[(x_0) - \Delta x f^{'}(x_0) + \tfrac{\Delta x^2 f^{''}(x_0)}{2!} + \ldots\right], \end{aligned}$$

agrupando términos

$$\Rightarrow f_c^2(x_0) = (a + b + c)f(x_0) + (a - c)\Delta x f^{'}(x_0) + (a + c)\Delta x^2 f^{''}(x_0) + \ldots.$$

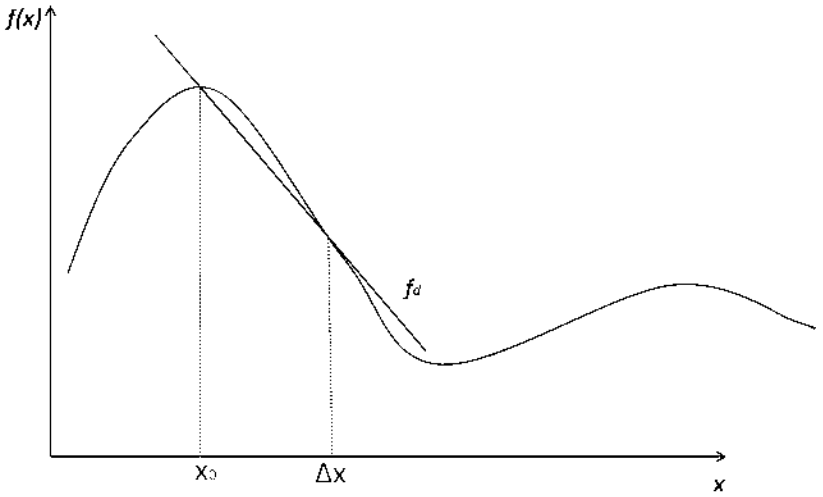

Figura 3.7: Aproximación de la derivada de f en x_0 por la derecha.

Dado que f_c^2 es la segunda derivada, entonces se debe cumplir que f_c^2 sea del mismo orden que $f^{''}(x_0)$, por tanto a,b y c deben cumplir que

$$a + b + c = 0$$

$$a - c = 0$$

$$a + c = \frac{2}{\Delta x^2}$$

resolviendo el sistema anterior se obtiene

$$a = \frac{1}{\Delta x^2}$$

$$b = \frac{-2}{\Delta x^2}$$

$$c = \frac{1}{\Delta x^2}.$$

Sustituyendo en (3.6), entonces se obtiene la expresión

$$f_c^2(x_0) = \frac{f(x_0 + \Delta x) - 2f(x_0) + f(x_0 - \Delta x)}{\Delta x^2}. \tag{3.7}$$

La deducción de una derivada de orden mayor se hace de manera análoga, aunque naturalmente más complicada. Finalmente, aclarando ideas, se tiene que para aplicar las diferencias finitas en un intervalo $[a, b]$, dada una malla en $[a, b]$

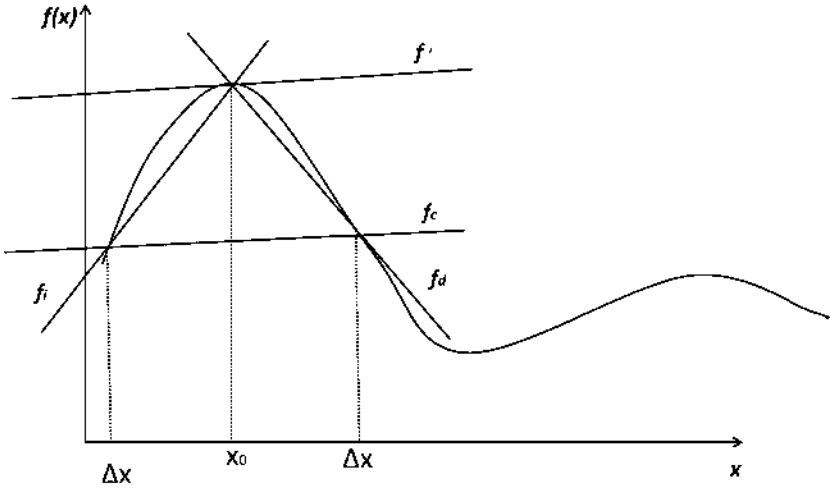

Figura 3.8: Aproximaciónes por la derecha, por la izquierda y centrada.

de una dimensión, se tenía un conjunto de puntos $a = x_0$, x_1, x_2, x_3, ... ,x_n. Para utilizar, por ejemplo, la formula de diferencias por la derecha f_d, basta con tener en cuenta quien juega el papel del nodo x_0 en cada aproximación y que $\Delta x = x_{i+1} - x_i$.

En el caso de dos dimensiones el teorema de Taylor también es la base para la deduccion de las diferencias finitas. A veces se pueden formular esquemas implícitos o explícitos, dependiendo de la naturaleza del problema que es crucial, pues la EDP a aproximar y la información del problema influye directamente en la selección del esquema. A continuación se aborda el esquema hacia adelante en el tiempo y centrado en el espacio llamado **FTCS** por sus siglas en inglés (Forward Time Centered Space).

El problema consiste en lo siguiente: sea Ω la región donde se aproxima la función u, sea u una función tal que $u : \mathbb{R}^2 \times \mathbb{R}^+ \to \mathbb{R}$, y que además depende del tiempo, por lo que denotamos a u como una funcion de $2+1D$, es decir, como $u(x, y, t)$, la ecuación de difusión isotrópica está dada por siguiente expresión

$$u_t = \nabla \cdot (c \nabla u), \tag{3.8}$$

donde c es una constante llamada coeficiente de conducción; además se tienen las siguientes condiciones de frontera de Dirichlet y la condición inicial dadas por

- $u(x, y, t) = h(x, y, t), \qquad \forall x, y \in \partial\Omega$
- $u(x, y, 0) = g(x, y, 0), \qquad \forall x, y \in \Omega.$

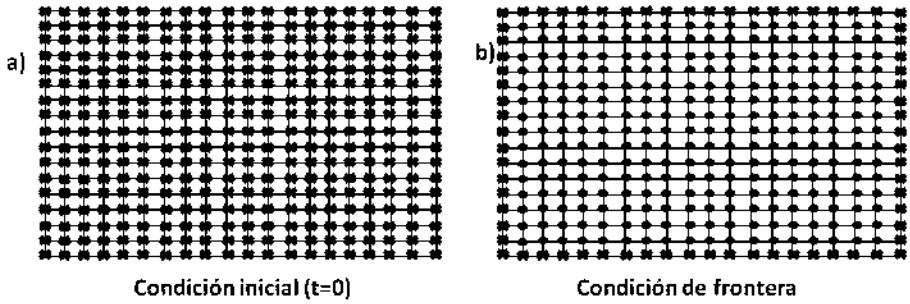

Figura 3.9: En **a)** se conocen los valores de la función en todos los nodos de la malla, en cambio, en **b)** sólo se conocen los valores de la función en los nodos de la frontera.

Nótese que las condiciones anteriores dadas por la función g significan que se conoce a la función u en los nodos de la malla en un inicio, es decir en $t = 0$ conocemos los valores de u en los nodos frontera e interiores, además a través del tiempo se conoce la función u en los nodos frontera.

Sea M una malla de dimensión de $n \times m$. La figura 3.9 muestra el planteamiento numérico del problema. Por medio del esténcil (molécula) FTCS que se muestra en la figura 3.10. Como se ve este esténcil es el resultado de utilizar diferencias centradas en el espacio (en las variables x e y) y utilizar las aproximaciones ya conocidas del paso de tiempo hacia adelante para conocer la aproximación en el próximo paso de tiempo, el superíndice k indica el paso de tiempo, los subindices i, j indican el nodo espacial de la malla que se utiliza; como se puede ver el nodo $n_{i,j}^{k+1}$ es aproximado usando los nodos ya conocidos del paso de tiempo anterior $n_{i,j}^{k}$,$n_{i+1,j}^{k}$,$n_{i-1,j}^{k}$,$n_{i,j+1}^{k}$ y $n_{i,j-1}^{k}$.

Al sustituir en la ecuación de difusión las formulas centradas de diferencias finitas, tanto en la dirección x e y, además de la formula hacia adelante en diferencias que aproxima u_t, la formula del esquema FTCS está dada por la siguiente expresión

$$u_{i,j}^{t+1} = u_{i,j}^{t+1} + \frac{c\Delta t(u_{i,j-1}^{t} - 4u_{i,j}^{t} + u_{i,j+1}^{t})}{\Delta x^2} + \frac{c\Delta t(u_{i-1,j}^{t} - 4u_{i,j}^{t} + u_{i+1,}^{t})}{\Delta y^2}. \quad (3.9)$$

Por mencionar un caso de la aplicación de la ecuación (3.8) con las condiciones mencionadas, por ejemplo, sería el problema de difusión de calor en una región Ω, donde se busca aproximar el comportamiento de la temperatura u a través del tiempo en una región Ω, donde en principio se conoce la temperatura inicial en la región Ω y la temperatura en la frontera de la región tambien es conocida en cualquier tiempo t, por lo que solamente hay que aproximar la temperatura en mediante diferencias finitas en los nodos interiores de la malla que está sobre

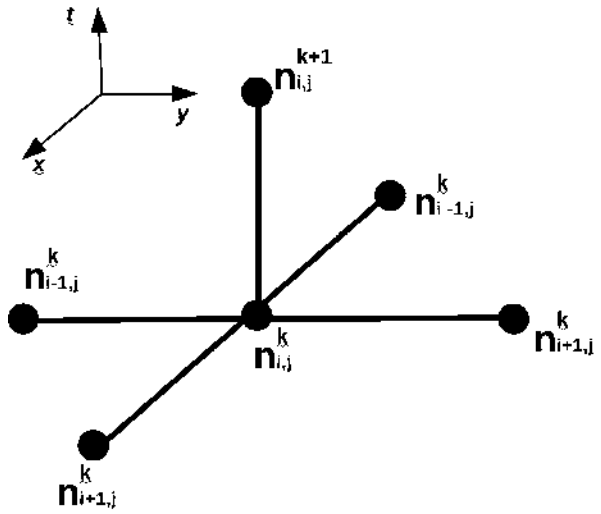

Figura 3.10: Forma del estencil FTCS para aproximar la ecuación de difusión isotrópica.

Ω.

3.3. Diferencias finitas generalizadas

3.3.1. Formulación general de un esquema en diferencias

Por tanto para comenzar la formulación general [5], sea Lu un operador diferencial lineal de segundo orden definido por

$$Lu(p_0) = A(p_0)u_{xx} + B(p_0)u_{xy} + C(p_0)u_{yy} + D(p_0)u_x + E(p_0)u_y + F(p_0), \quad (3.10)$$

donde se considera un conjunto de nodos $p_0, p_1, p_2, p_3, p_4, ..., p_q$.

Sea $\hat{L}$ una combinación lineal, la cual es en sí el esquema en diferencias finitas a encontrar, $\hat{L}$ está dado por la siguiente expresión

$$\hat{L}(p_0) = \Gamma_0 u(p_0) + \Gamma_1 u(p_1) + ... + \Gamma_q u(p_q), \quad (3.11)$$

donde p_0 es llamado nodo central, y $p_1, p_2, p_3, p_4, ..., p_q$ son los nodos vecinos de p_0, la figura 3.11 muestra una ilustración de este conjunto de nodos en el plano xy.

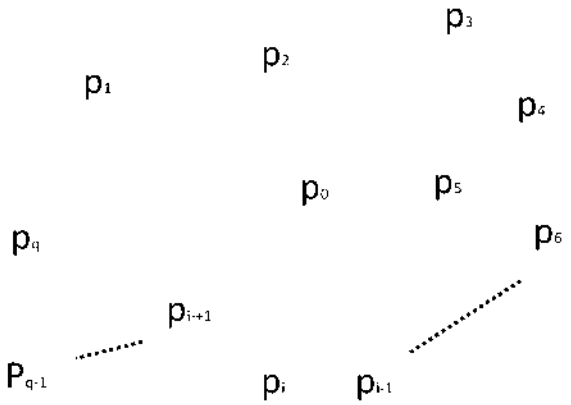

Figura 3.11: Nodos en el plano empleados para formular el esquema de diferencias, p_0 es el nodo central y $p_1, p_2, p_3, p_4, ..., p_q$ son los nodos que lo rodean, llamados nodos vecinos de p_0.

Por tanto, se dirá que el esquema en diferencias es **consistente** si la diferencia entre el operador diferencial en p_0 y el esquema en p_0 tiende a cero conforme los nodos vecinos $p_1, p_2, p_3, p_4, ..., p_q$ se acercan al nodo central p_0, es decir, el esquema en diferencias es consistente si se cumple lo suiguiente

$$Lu(p_0) - \hat{L}(p_0) \to 0, \quad si \quad p_1, p_2, p_3, p_4, ..., p_q \to p_0. \tag{3.12}$$

La expresión (3.12) se denomina error local de truncamiento. Al hacer la expansión en serie de Taylor del error local de truncamiento, se tiene

$$Lu(p_0) - \hat{L}(p_0) = \left[Au_{xx} + Bu_{xy} + Cu_{yy} + Du_x + Eu_y + Fu\right]_{p_0} - \sum_{i=0}^{q} \Gamma_i u(p_i) =$$

$$= \left(F - \sum_{i=0}^{q} \Gamma_i\right) u + \left(D - \sum_{i=1}^{q} \Gamma_i \Delta x_i\right) u_x + \left(E - \sum_{i=1}^{q} \Gamma_i \Delta y_i\right) u_y +$$

$$+ \left(A - \sum_{i=1}^{q} \frac{\Gamma_i (\Delta x_i)^2}{2}\right) u_{xx} + \left(B - \sum_{i=1}^{q} \Gamma_i \Delta x_i \Delta y_i\right) u_{xy} +$$

$$+ \left(C - \sum_{i=1}^{q} \frac{\Gamma_i (\Delta y_i)^2}{2}\right) u_{yy} + \mathcal{O}\left(\text{máx}\{\Delta x_i, \Delta y_i\}\right)^3$$

donde Δx_i y Δy_i son los tamaños de paso horizontal y vertical entre p_0 y el nodo vecino p_i.

De la expansión anterior se obtiene el sistema de ecuaciones (3.13), el cual consiste en seis ecuaciones y q incógnitas, por lo que es un sistema indeterminado, pero puede ser resuelto de distintas maneras.

$$\begin{pmatrix} 1 & 1 & ... & 1 \\ 0 & \Delta x_1 & ... & \Delta x_q \\ 0 & \Delta y_1 & ... & \Delta y_q \\ 0 & (\Delta x_1)^2 & ... & (\Delta x_q)^2 \\ 0 & \Delta x_1 \Delta y_1 & ... & \Delta x_q \Delta y_q \\ 0 & (\Delta y_1)^2 & ... & (\Delta y_q)^2 \end{pmatrix} \begin{pmatrix} \Gamma_0 \\ \Gamma_1 \\ \Gamma_2 \\ . \\ . \\ . \\ \Gamma_q \end{pmatrix} = \begin{pmatrix} F(p_0) \\ D(p_0) \\ E(p_0) \\ 2A(p_0) \\ B(p_0) \\ 2C(p_0) \end{pmatrix}. \tag{3.13}$$

Hasta ahora es importante resaltar lo siguiente: encontrar el esquema de diferencias, consiste en encontrar las constantes Γ_i.

Al observar la primera ecuación del sistema (3.13), se puede notar que la suma de todos los Γ_i es igual a $F(p_0)$, esto se muestra en la expresión siguiente

$$\Gamma_0 + \Gamma_1 + \Gamma_2 + \Gamma_3 + \Gamma_4 ... + \Gamma_q = F(p_0). \tag{3.14}$$

Por tanto basta encontrar $\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4, ..., \Gamma_q$ para determinar Γ_0, en otras palabras

$$\Gamma_0 = F(p_0) - \sum_{i=1}^{q} \Gamma_i, \tag{3.15}$$

de esta forma al simplificar el sistema (3.13) se obtiene el siguiente sistema

$$\begin{pmatrix} \Delta x_1 & ... & \Delta x_q \\ \Delta y_1 & ... & \Delta y_q \\ (\Delta x_1)^2 & ... & (\Delta x_q)^2 \\ \Delta x_1 \Delta y_1 & ... & \Delta x_q \Delta y_q \\ (\Delta y_1)^2 & ... & (\Delta y_q)^2 \end{pmatrix} \begin{pmatrix} \Gamma_1 \\ \Gamma_2 \\ . \\ . \\ . \\ \Gamma_q \end{pmatrix} = \begin{pmatrix} D(p_0) \\ E(p_0) \\ 2A(p_0) \\ B(p_0) \\ 2C(p_0) \end{pmatrix}. \tag{3.16}$$

Resolviendo (3.16), se obtienen $\Gamma_1, \Gamma_2, \Gamma_3, \Gamma_4, ..., \Gamma_q$, y después Γ_0, por lo que se conocen finalmente las constantes Γ_i y por tanto queda determinado el esquema de diferencias buscado.

3.4. Esquema de nueve puntos para la ecuación de difusión anisotrópica

3.4.1. Planteamiento

Trabajando ahora con la ecuación de difusión anisotrópica, la cual está dada por la ecuación

$$u_t = \nabla \cdot (c(x, y, t)\nabla u(x, y, t)) \tag{3.17}$$

donde $c(x, y, t)$ es llamada función de difusión. Expresando (3.17) vectorialmente, se tiene la siguiente expresión

$$u_t = c\nabla^2 u + \nabla c \cdot \nabla u,$$

entonces

$$u_t = cu_{xx} + cu_{yy} + c_x u_x + c_y u_y, \tag{3.18}$$

obteniendo finalmente la expresión (3.18), la cual se puede aproximar como se describió en la sección anterior, pues se tiene el operador

$$Lu = cu_{xx} + cu_{yy} + c_x u_x + c_y u_y, \tag{3.19}$$

esto es

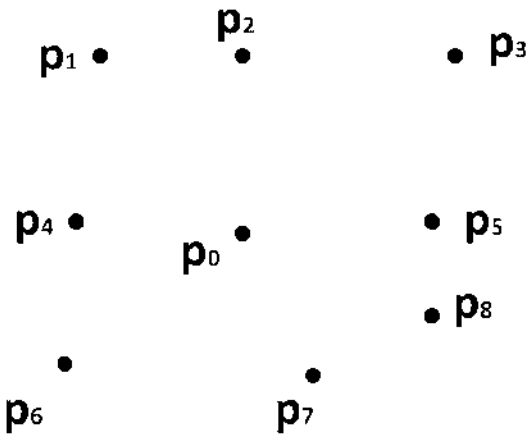

Figura 3.12: Esquema de nueve nodos.

$$\begin{pmatrix} 1 & 1 & ... & 1 \\ 0 & \Delta x_1 & ... & \Delta x_8 \\ 0 & \Delta y_1 & ... & \Delta y_8 \\ 0 & (\Delta x_1)^2 & ... & (\Delta x_8)^2 \\ 0 & \Delta x_1 \Delta y_1 & ... & \Delta x_8 \Delta y_8 \\ 0 & (\Delta y_1)^2 & ... & (\Delta y_8)^2 \end{pmatrix} \begin{pmatrix} \Gamma_0 \\ \Gamma_1 \\ \Gamma_2 \\ \cdot \\ \cdot \\ \cdot \\ \Gamma_8 \end{pmatrix} = \begin{pmatrix} 0 \\ c_x(p_0) \\ c_y(p_0) \\ 2c(p_0) \\ 0 \\ 2c(p_0) \end{pmatrix}. \tag{3.20}$$

La definición de la función c se muestra en el primer capítulo, donde se describió el modelo de Perona-Malik. Por medio de diferencias finitas se aproxima c_x y c_y. Por último, resolviendo el sistema anterior tal y como lo se discutió en la

sección anterior, es como se obtiene el esquema de diferencias para la ecuación de difusión anisotrópica.

3.5. Aproximación a la ecuación de Poisson

3.5.1. Planteamiento

El problema de Poisson es un ejemplo de un problema estacionario [13], es decir, un problema que no depende del tiempo, por lo que el problema de aproximar la ecuación de Poisson consiste en lo siguiente: se quiere aproximar la ecuación de Poisson en una región Ω, la ecuación de Poisoon está dada por la siguiente expresión

$$\nabla^2 u(x,y) = f(x,y),$$

donde u y f están definidas en Ω. Nótese que si $f(x,y) = 0$, entonces se estaría trabajando con la ecuación de Laplace.

Para aproximar numéricamente la ecuación de Poisson usando diferencias finitas, es necesario disponer de una malla en la región Ω y disponer de condiciones en la frontera, por tanto la formulación numérica del problema es la siguiente: sean f y u funciones definidas en una región Ω, donde f y u son funciones de dos variables y f es una función dada. Se quiere aproximar la solución de la ecuación de Poisson en los nodos de una malla rectángular lógicamente estructurada que está sobre Ω, pero además se disponen de condiciones de Dirichlet, es decir, si $(x,y) \in \partial\Omega$, entonces $u(x,y) = g(x,y)$, donde g es una función conocida.

Para llevar a cabo la deducción del esquema en diferencias, sea M una malla sobre la región Ω como la que se muestra en la figura 3.13, donde Δx y Δy son las distancias entre los nodos en la dirección x e y, respectivamente.

Los valores de u son conocidos en los nodos frontera, por lo que sólo hay que aproximar la solución en los nodos interiores, se denotará los valores (por aproximar) de u en los nodos interiores como se muestra en la figura 3.14. Las formulas de la segunda derivada respecto de x e y en diferencias finitas centradas en el espacio están dadas por

$$u_{xx}(x_{i,j}, y_{i,j}) \approx \frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{\Delta x^2}, \quad u_{yy}(x_{i,j}, y_{i,j}) \approx \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{\Delta y^2},$$

donde $u_{i,j} = u(x_{i,j}, y_{i,j})$.

Sustituyendo las formulas anteriores en la ecuación de Poisson,se tiene que

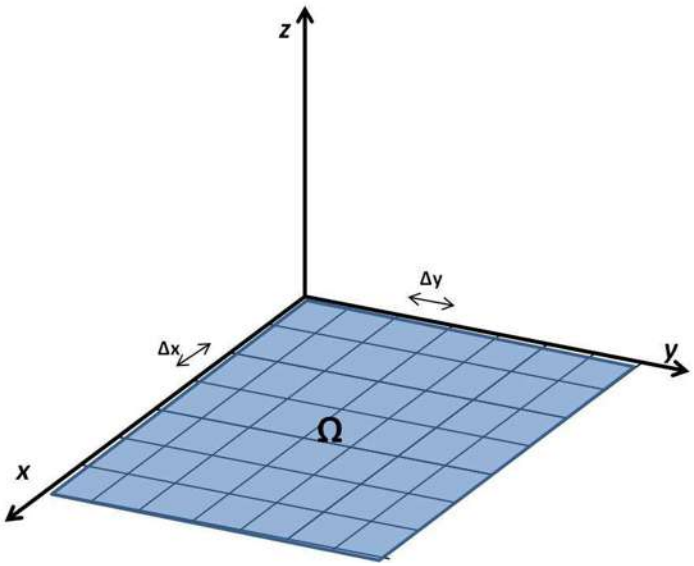

Figura 3.13: Región Ω en el plano, en la cual se tiene una malla rectángular lógicamente estructurada.

$$\frac{u_{i,j-1} - 2u_{i,j} + u_{i,j+1}}{\Delta x^2} + \frac{u_{i-1,j} - 2u_{i,j} + u_{i+1,j}}{\Delta y^2} = f(x_{i,j}, y_{i,j}),$$

reacomodando la expresión anterior se obtiene la siguiente expresión

$$\Delta y^2 u_{i,j-1} + (-2\Delta x^2 - 2\Delta y^2) u_{i,j} + \Delta y^2 u_{i,j+1} + \Delta x^2 u_{i-1,j} + \Delta x^2 u_{i+1,j} = \Delta x^2 \Delta y^2 f(x_{i,j}, y_{i,j}).$$

Lo que se ha hecho hasta ahora es obtener una expresión que sirva para aproximar la función u en el nodo $u_{i,j} = (x_{i,j}, y_{i,j})$, pero en este caso no se puede obtener una formula explicita para aproximar $u_{i,j}$, pues en la expresión anterior se tienen cinco variables: $u_{i-1,j}$, $u_{i,j}$, $u_{i+1,j}$, $u_{i,j-1}$ y $u_{i,j+1}$, que en el caso de que $u_{i,j}$ sea un nodo adyacente a los nodos frontera, a lo más se podran tener dos valores de las variables anteriores, es decir, en el mejor de los casos, se seguirá teniendo tres variables en la expresión anterior.

Sin embargo, se puede ver que parar $u_{2,2}$ se tiene

$$\Delta y^2 u_{2,1} + (-2\Delta x^2 - 2\Delta y^2) u_{2,2} + \Delta y^2 u_{2,3} + \Delta x^2 u_{1,2} + \Delta x^2 u_{3,2} = \Delta x^2 \Delta y^2 f(x_{2,2}, y_{2,2}),$$

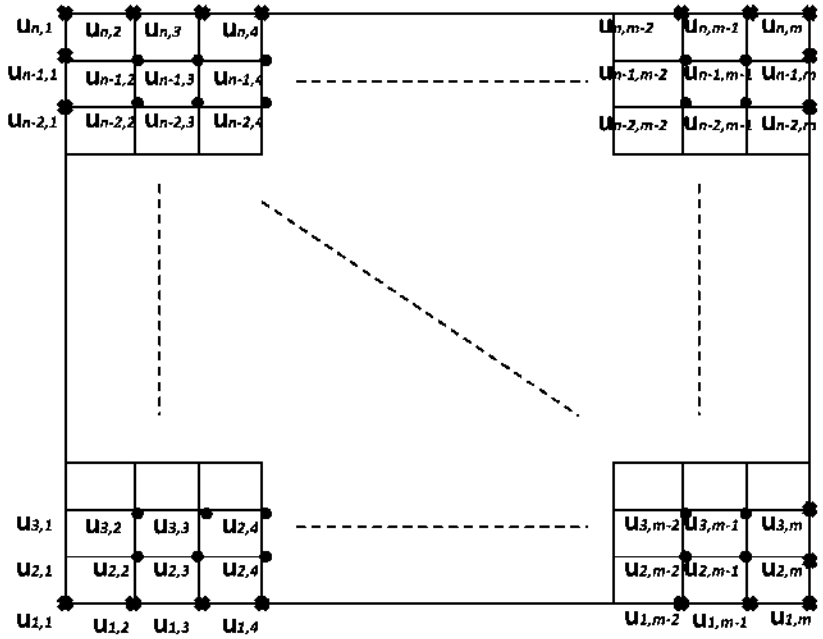

Figura 3.14: Manera en que son enumerados los nodos de la malla.

donde $u_{1,2}$ y $u_{2,1}$ están sobre la forntera y por tanto son valores conocidos. Si los valores conocidos se pasan al lado derecho de la igualdad, entonces se tiene que

$$(-2\Delta x^2-2\Delta y^2)u_{2,2}+\Delta y^2 u_{2,3}+\Delta x^2 u_{3,2} = \Delta x^2\Delta y^2 f(x_{2,2},y_{2,2})-\Delta y^2 u_{2,1}-\Delta x^2 u_{1,2}, \tag{3.21}$$

análogamente, para aproximar $u_{2,3}$ se tendrá que

$$\Delta y^2 u_{2,2}+(-2\Delta x^2-2\Delta y^2)u_{2,3}+\Delta y^2 u_{2,4}+\Delta x^2 u_{3,3} = \Delta x^2\Delta y^2 f(x_{2,2},y_{2,2})-\Delta x^2 u_{1,3}, \tag{3.22}$$

y para la aproximación de $u_{2,4}$ se tiene

$$\Delta y^2 u_{2,3}+(-2\Delta x^2-2\Delta y^2)u_{2,4}+\Delta y^2 u_{2,5}+\Delta x^2 u_{3,4} = \Delta x^2\Delta y^2 f(x_{2,2},y_{2,2})-\Delta x^2 u_{1,4}. \tag{3.23}$$

Si se tiene un malla de $n \times m$, entonces se tendrán $(n-2)\times(m-2)$ nodos interiores, por lo que si se obtienen las expresiones de los $(n-2)\times(m-2)$

nodos interiores de manera análoga como se obtuvieron (3.21), (3.22) y (3.23), entonces se tendrán $(n-2)\times(m-2)$ ecuaciones, las cuales pueden acomodarse en forma del siguiente sistema de ecuaciones

$$\mathbf{KX} = \mathbf{F},$$

donde el vector $\mathbf{X}$ es tal que

$$X = \begin{pmatrix} u_{2,2} \\ u_{2,3} \\ . \\ . \\ . \\ u_{2,(m-1)} \\ u_{3,2} \\ u_{3,3} \\ . \\ . \\ . \\ u_{3,(m-1)} \\ u_{4,2} \\ . \\ . \\ . \\ . \\ u_{(n-1),(m-1)} \end{pmatrix}. \tag{3.24}$$

La matriz $\mathbf{K}$ es de dimensión $(n-2)(m-2)\times(n-2)(m-2)$ y tiene la forma

$$K = \begin{pmatrix} A & B & O & O & ... & O & O & O & O \\ B & A & B & O & ... & O & O & O & O \\ O & B & A & B & ... & O & O & O & O \\ & & & . & & & & & \\ & & & & . & & & & \\ & & & & & . & & & \\ O & O & O & O & ... & B & A & B & O \\ O & O & O & O & ... & O & B & A & B \\ O & O & O & O & ... & O & O & B & A \end{pmatrix}, \tag{3.25}$$

donde A, B y O son submatrices de dimensión $(m-2)\times(m-2)$ tal que

$$A = \begin{pmatrix} \beta & \Delta y^2 & 0 & ... & 0 & 0 & 0 \\ \Delta y^2 & \beta & \Delta y^2 & 0 & ... & 0 & 0 \\ 0 & \Delta y^2 & \beta & \Delta y^2 & 0 & ... & 0 \\ & & & . & & & \\ & & & & . & & \\ & & & & & . & \\ 0 & 0 & ... & \Delta y^2 & \beta & \Delta y^2 & 0 \\ 0 & 0 & ... & 0 & \Delta y^2 & \beta & \Delta y^2 \\ 0 & 0 & ... & 0 & 0 & \Delta y^2 & \beta \end{pmatrix}, \tag{3.26}$$

$$B = \begin{pmatrix} \Delta x^2 & 0 & 0 & ... & 0 & 0 & 0 \\ 0 & \Delta x^2 & 0 & 0 & ... & 0 & 0 \\ 0 & & \Delta x^2 & 0 & 0 & ... & 0 \\ & & & . & & & \\ & & & & . & & \\ & & & & & . & \\ 0 & & 0 & 0 & 0 & \Delta x^2 & 0 \\ 0 & & 0 & 0 & 0 & ... & \Delta x^2 \end{pmatrix}, \tag{3.27}$$

y donde $\beta = -2\Delta x^2 - 2\Delta y^2$. Finalmente el vector $\mathbf{F}$ es tal que

$$F = \begin{pmatrix} \Delta x^2\Delta y^2 f(x_{2,2}, y_{2,2}) - \Delta x^2 u_{1,2} - \Delta y^2 u_{2,1} \\ \Delta x^2\Delta y^2 f(x_{2,3}, y_{2,3}) - \Delta y^2 u_{1,3} \\ \Delta x^2\Delta y^2 f(x_{2,4}, y_{2,4}) - \Delta y^2 u_{1,4} \\ \vdots \\ \Delta x^2\Delta y^2 f(x_{2,(m-2)}, y_{2,(m-2)}) - \Delta y^2 u_{1,(n-2)} \\ \Delta x^2\Delta y^2 f(x_{2,(m-1)}, y_{2,(m-1)}) - \Delta y^2 u_{1,n-1} - \Delta x^2 u_{2,n} \\ \Delta x^2\Delta y^2 f(x_{3,2}, y_{3,2}) - \Delta y^2 u_{3,1} \\ \Delta x^2\Delta y^2 f(x_{3,3}, y_{3,3}) \\ \vdots \\ \Delta x^2\Delta y^2 f(x_{3,(m-1)}, y_{3,(m-1)}) - \Delta y^2 u_{3,(m-1)} \\ \Delta x^2\Delta y^2 f(x_{4,2}, y_{4,2}) - \Delta y^2 u_{4,1} \\ \Delta x^2\Delta y^2 f(x_{4,3}, y_{4,3}) \\ \vdots \\ \Delta x^2\Delta y^2 f(x_{4,(m-1)}, y_{4,(m-1)}) - \Delta y^2 u_{4,(m-1)} \\ \vdots \\ \Delta x^2\Delta y^2 f(x_{4,(m-1)}, y_{4,(m-1)}) - \Delta y^2 u_{4,(m-1)} \\ \Delta x^2\Delta y^2 f(x_{4,(m-1)}, y_{4,(m-1)}) - \Delta y^2 u_{4,(m-1)} \\ \vdots \\ \Delta x^2\Delta y^2 f(x_{4,(m-1)}, y_{4,(m-1)}) - \Delta x^2 u_{n,(m-2)} \\ \Delta x^2\Delta y^2 f(x_{(n-1),(m-1)}, y_{(n-1),(m-1)}) - \Delta x^2 u_{n,(m-1)} - \Delta x^2 u_{(n-1),m} \end{pmatrix}. \tag{3.28}$$

3.5.2. Ejemplo de aplicación

En la sección anterior se dedujo un esquema implícito para aproximar la ecuación de Poisson, sin embargo, para que se puedan ver los resultados del planteamiento a continuación se muestra un ejemplo de aplicación, en el cual se aproxima numéricamente la ecuación de Poisson con un par de funciones f y g específicas.

Para comparar resultados, se usará la función $u = e^{(4x+4y)}$, pues al conocer de antemano la solución analítica es como se podrá hacer una comparación y observar que tan precisas son las aproximaciones con el esquema.

Por tanto, sea la ecuación de Poisson dada por

$$\nabla^2 u(x,y) = f(x,y),$$

y sean:

- $\Omega = [0,1] \times [0,1]$ la región del plano xy donde se aproxima la solución de u.
- Como se utiliza $u(x,y) = e^{(4x+4y)}$, entonces por la ecuación de Poisson se tiene que

$$f(x,y) = 16e^{(4x+4y)} + 16e^{(4x+4y)} = 32e^{(4x+4y)}$$

  .
- Sea $g(x,y) = e^{(4x+4y)}$ las condiciones de Dirichlet.

Aplicando el esquema implícito descrito en la sección anterior a mallas uniformes sobre Ω de 11×11 y 41×41, se tienen los resultados mostrados en las figuras 3.15 y 3.16, respectivamente.

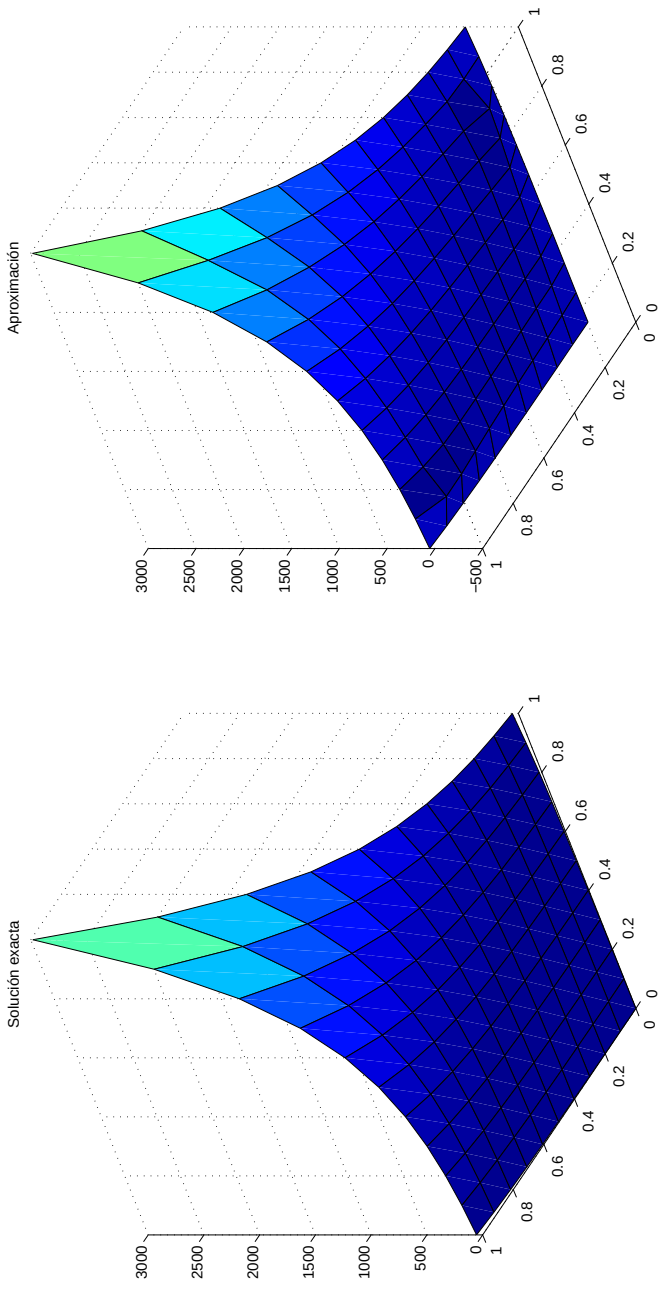

Figura 3.15: Resultado en una malla uniforme de 11×11.

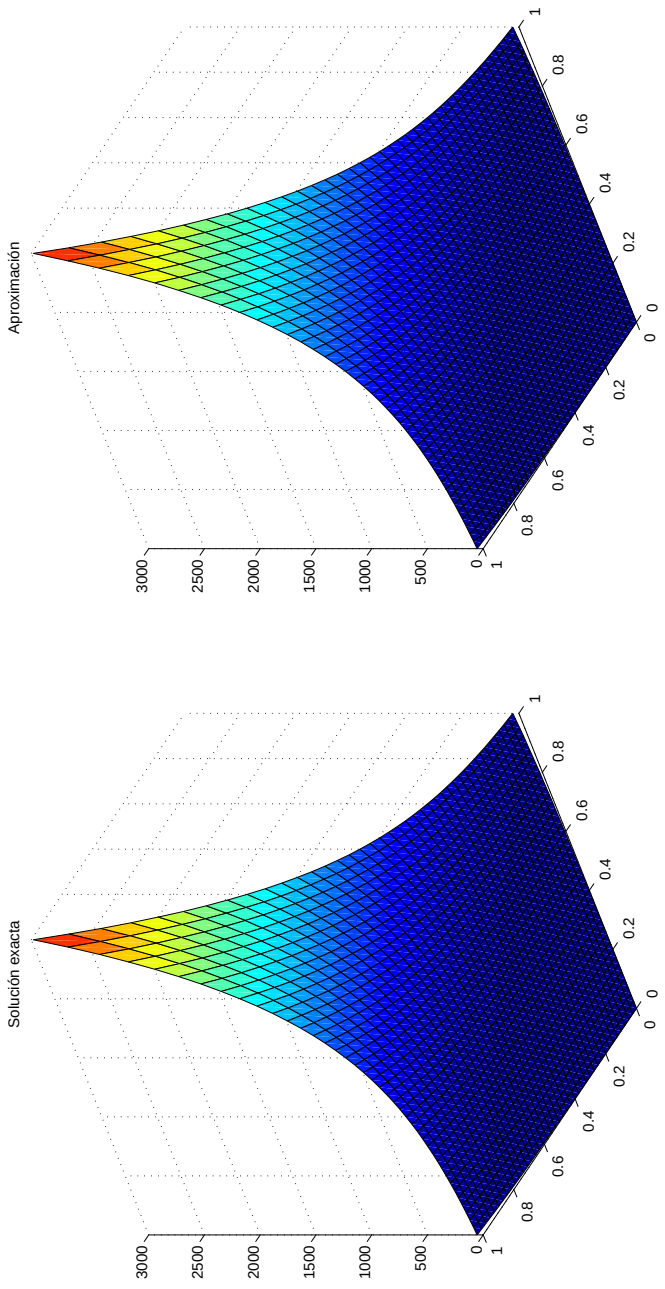

Figura 3.16: Resultado en una malla uniforme de 41×41.

Capítulo 4

Aplicación y resultados en la detección de bordes

4.1. Implementación.

4.1.1. Introducción general

Finalmente en este capítulo se describen los resultados obtenidos de la implementación de la teoría de los tres capítulos anteriores, pues el objetivo es aplicar dicha teoría en el problema de detección de bordes. Especificamente, en este capítulo se muestran los resultados en la detección de bordes por medio de mallas adaptativas y del modelo de Perona-Malik empleando diferencias finitas, así como resultados de un algoritmo que no se tenía contemplado en un inicio, pues durante el intento de implementar el método LBFGS-B surgió la idea de un nuevo algoritmo al que denominó algoritmo Fuerte, tal algoritmo es descrito en este capítulo. Se investigó si este algoritmo ya había sido publicado he implementado, pero hasta la edición de este trabajo no se encontró ningun algoritmo que ya aplicara las ideas del algoritmo Fuerte.

En este análisis de utilizaron principalmente cuatro imaǵenes con diferentes resoluciónes para la obtención de resultados, además de hizo una comparación con los resultados de los filtros de Canny y Sobel. Finalmente, se realizó la construción de una interfaz gráfica en Matlab que conjunte las implementaciones de mallas adaptativas y algoritmo Fuerte para usarlas en el modelo de Perona-Malik, pues la base de las mallas adaptativas radica en el gradiente de la función de intensidad. El gradiente de la superficie a la cual se adapta la malla juega un papel especial en la generación de mallas adaptativas, sumando esto a que el modelo de Perona-Malik con la función de difusión planteada también se basa en el gradiente de la función de intensidad de la imagen, entonces, tanto las mallas adaptativas y el modelo de Perona-Malik tienen como principal estrategia el uso del gradiente de la función de intensidad, es por esto que en la imple-

mentación de estos dos conceptos se unen. Por otro lado, el algoritmo Fuerte es independiente de las mallas adaptativas y puede ser empleado por si sólo para la detección de bordes, sin embargo, la convexidad de las mallas adaptativas es fundamental para la establilidad del modelo de Perona-Malik, por lo que también se muestra en este capitulo los resultados de la concatenación del algoritmo Fuerte y el método LBFGS para la generación de mallas adaptativas convexas. Si la imagen tiene una complejidad en la distribución de color, en un sentido de una clase difuminación de color, es decir, que ni siquiera a simple vista se pueden visualizar los bordes de la imagen, un ejemplo de este sealamiento sería el tratar de generar una malla adaptativa de una imagen al óleo de Van Gogh, que se muestra en la figura 4.1.

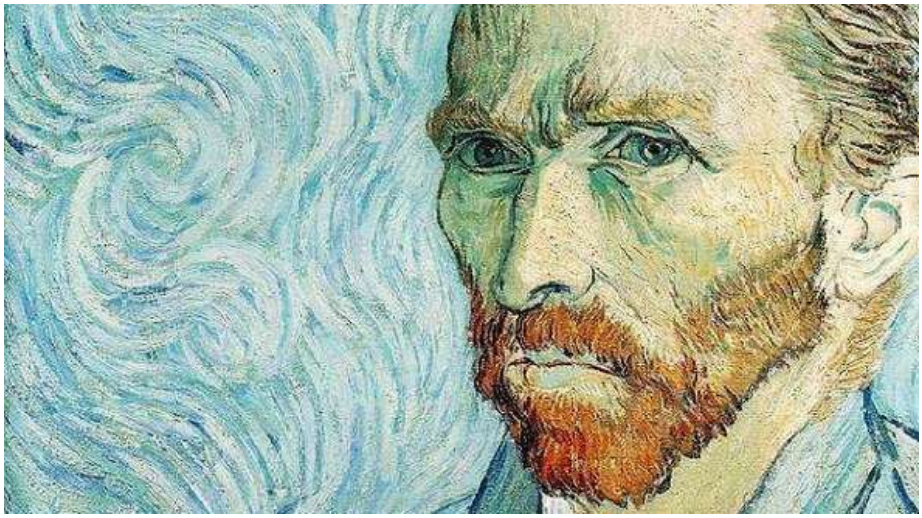

Figura 4.1: Imagen digital de la pintura al óleo de Van Gogh.

Es importante mencionar que casi la totalidad del código utilizado son implementaciones propias, es decir, no se hizo uso solamente de funciones propias de Matlab que tan sólo tuvieran que llamarse, sino que casi todas las funciones que utiliza la interfaz fueron generadas en el transcurso del presente trabajo.

4.1.2. Interfaz gráfica en Matlab

La interfaz gráfica se construyó en el ambiente GUI (Graphical User Interface) de Matlab, la intención fue construir una interfaz lo más intuitiva posible para el usuario en la cual se pueda analizar la detección de bordes en imágenes digitales con las implementaciones mencionadas.

Para remediar sustancialmente el problema de tiempo de ejecución en las im-

plementaciones de los métodos de optimización programados en Matlab se utilizaron los famosos archivos MEX, pues estos archivos son mucho más eficientes en cuanto a tiempo de ejecución, comparados con los clasicos archivos *.m utilizados para programar funciones (rutinas) en Matlab.

Los archivos MEX propios de Matlab deben su nombre a las siglas **Matlab Executable**, esta clase de archivos tienen una extructura particular y muy diferente a otros archivos de Matlab. Los archivos MEX de Matlab sirven principalmente para poder llamar y ejecutar rutinas escritas en C, C^{++} y Fortran como si hubieran sido escritas como archivos $*.m$ de Matlab.

Una de las principales características de los archivos MEX es que solamente tienen una función (rutina) que tiene el mismo nombre que el archivo MEX, además del hecho de que los archivos MEX son ligados dinámicamente a rutinas que el interprete de Matlab puede ejecutar, por lo que una vez creado un archivo MEX, este puede ser llamado como cualquier función común $*.m$ de Matlab.

Hay varias maneras de crear un archivo MEX, por ejemplo, una manera sería escribir la rutina en C, C^{++} o Fortran para posteriormente compilarla en Matlab, dando como resultado el archivo MEX. Otra manera sería que si ya se tiene una función de tipo $*.m$ de Matlab, entonces usar la herramienta de Matlab llamada **Coder**, la cual es una especie de compilador, en el que se puede generar el archivo MEX a partir de la función *.m.

Esta ultima alternativa fue la que se usó para generar los archivos MEX necesarios para la interfaz gráfica, pues dado que ya se tenian las funciones programadas de los métodos BFGS y LBFGS en formato $*.m$, las cuales no fueron sencillas de programar, entonces solamente se utilizó *coder* para obtener los archivos MEX.

Al generar los archivos MEX usando *coder* se tienen ventajas y desventajas, la principal ventaja es que si se está solamente familiarizado con la programación de funciones $*.m$ en Matlab, entonces es muy práctico usar coder, pues evita la complicación de recurrir a la programación de las funciones en otros lenguajes de programación, y por tanto sólo es necesaria la función $*.m$ y *coder* para generar el archivo MEX.

Sin embargo, por otra parte hay algunas desventajas, pues al generar un archivo MEX a partir de un archivo $*.m$ usando *coder*, se generan una carpeta llamada codegen en la cual se genera una gran cantidad de archivos de distintos formatos que son necesarios para el archivo MEX, pudiendo inferir que esto reduce la velocidad con la que se ejecuta el archivo MEX creado.

Otra desventaja es que al escribir una función $*.m$ en Matlab, se utilizan comunmente funciones propias de Matlab como lo es la función **display**, la cual es una función que simplemente imprime texto en la linea de comandos, y funciones

como estas no son aceptadas por *coder* si no se agrega al inicio de la función la linea **coder.extrinsic('display')**.

Por otra parte, otra desventaja es que cuando se está muy familiarizado con las facilidades de la notación de Matlab, como lo es la concatenación de matrices o vectores en forma breve, muchas de estas abreviaciones en la contrucción de matrices y vectores en Matlab no son aceptadas en *coder*, por lo que hay que modificar el código de una manera menos breve y eficiente, incluso modificar el código de una manera nada optima en la función $*.m$ para que pueda ser compilada en *coder*, complicando mucho la programación de funciones para poder compilarlas en *coder* y obtener el archivo MEX.

Sin embargo es importante señalar que se uso *coder* no por ahorrar trabajo, sino porque ya se tenian programadas las funciones en formato $*.m$ y por falta de tiempo para volver a hacer la programación en otro lenguaje de programación, pues la implementación de los métodos fue en la parte final del presente trabajo. Pues el recurrir a el uso de los archivos MEX fue consecuencia de un inconveniente que surgió durante la elaboración de esta tesis.

Sin embargo, ya se tiene todo contemplado para que en futuros trabajos se pueda cambiar esto, pudiendo crear los archivos MEX de forma básica para así explotar todo su potencial y optimizando el funcionamiento de los códigos programados o programar todo desde cero en un lenguaje de programación como Java, C, C^{++} o Fortran.

Al crear una interfaz en GUI se crean dos archivos que van de la mano, pues la interfaz no funcionará si no se tiene este par de archivos en el mismo directorio. Los archivos que componen la interfaz tienen la extensión $*.m$ y el otro la extensión $*.fig$, donde el archivo $*.fig$ es el que se usa para construir la parte que se visualiza de la interfaz, es decir, es el archivo que compone la parte gráfica de la interfaz, pudiendo ensamblar los elementos que componen la interfaz, mientras que el archivo $*.m$ de la interfaz es el archivo que contiene toda la programación de transfondo que se realiza al utilizar un objeto gráfico del archivo *.fig.

Al modificar algo en el archivo $*.fig$ de la interfaz, automaticamente se escribe código en el archivo $*.m$ de la interfaz, para que haya congruencia entre ambos archivos, no es recomendable modificar los archivos por separado, pues esto causa incongruencias entre los códigos, produciendo errores muy dificiles de detectar.

El ambiente de la herramienta GUI de Matlab consiste en un entorno muy intuitivo para la contrucción de interfaces gráficas, pues basta con incrustar objetos gráficos como paneles, botones, menus, etc, y una vez terminado el diseño de la interfaz en el archivo $*.fig$, es decir, la construcción visual de la interfaz gráfica, entonces sólo falta programar que hará cada elemento de la interfaz.

Por ejemplo, para programar lo que debe hacer un botón determinado que es parte de la interfaz, se debe seleccionar su **Callback**, que al ser seleccionado nos manda automaticamente al archivo $*.m$ donde se encuentra la función que contiene las lineas de código que se realizarán al oprimir ese botón.

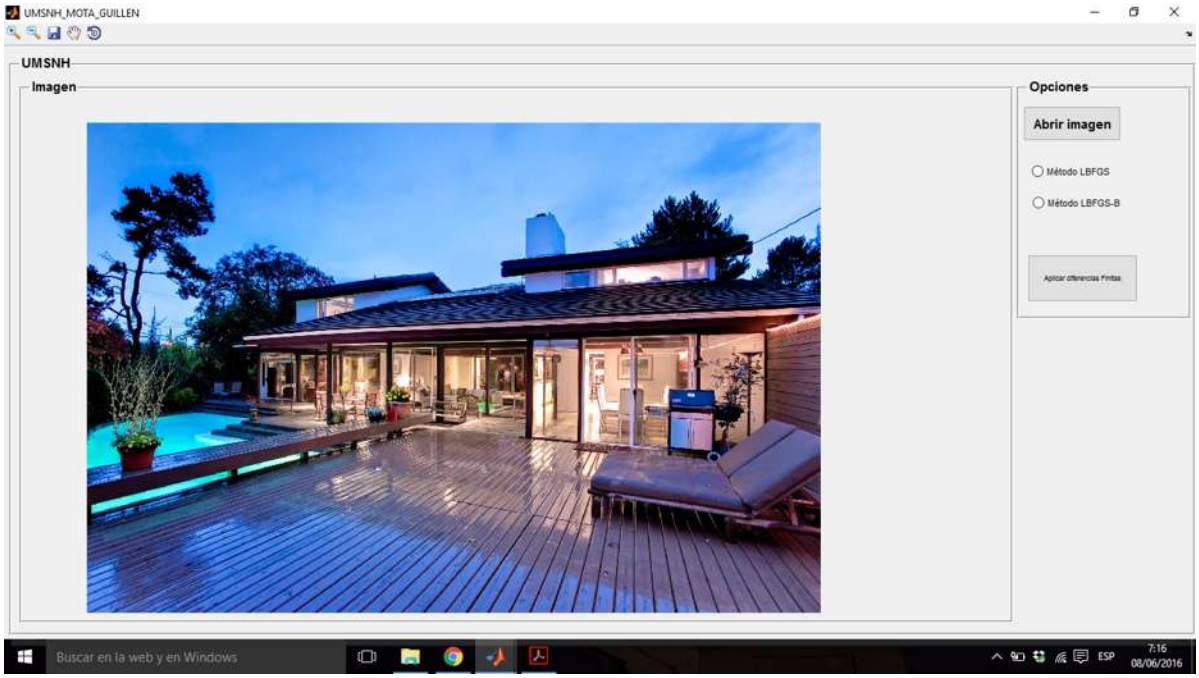

Figura 4.2: Imagen de la interfaz gráfica hecha en Matlab.

En esta tesis no se profundiza en el funcionamiento de GUI y Coder debido a que Matlab evoluciona continuamente, por lo que si se profundizará en las características de estas herramientas, pudiera suceder que estas no serían ciertas si el presente trabajo es leido en los proximos años, donde GUI y Coder podrían ser completamente diferentes a como lo fueron durante la elaboración del presente trabajo.

Sin embargo esta no es una limitante, pues Matlab cuenta con una documentación muy descriptiva, así como foros donde se pueden consultar dudas y resolver problemas. En la figura 4.2 se muestra una imagen de la interfaz gráfica construida durante el presente trabajo.

4.1.3. Interfaz gráfica desarrollada en Java

Puesto que la interfaz elaborada en Matlab requiere de muchos recursos computacionales debido a las razones ya explicadas, se optó por desarrollar una versión preliminar usando Java como lenguaje de programación. Esta implementación es completamente propia, pues toda la programación fue hecha durante la elaboración de la presente tesis.

La principal diferencia en el código entre las dos interfaces es que en la interfaz en Matlab se usó interpolaciíon para encontrar la función de intensidad de la malla adaptada en cada iteración, sin embargo en la interfaz en Java se hizo algo totalmente distinto, pues en este caso, cada vez que los nodos de la malla se movian y era necesario obtener una nueva función de intensidad, se optó por emplear un algoritmo que asignara a cada nodo un nuevo valor, dependiendo de la localización, simpre basandose en la función de intensidad original de la imagen digital.

Existen algunas diferencias importantes entre la interfaz en Matlab y la hecha en Java, pues por ejemplo, en la interfaz en Java en ocasiones algunas en algunas imágenes no se generan buenas mallas adaptativas usando el método LBFGS si la dimensión de la malla es menor a 400×400, otro fenómeno visto en la interfaz desarrollada en Java es que todas las mallas parecían ser convexas, pero esto se debe a que para considerar una definición de convexidad se debe tomar en cuenta la descripción del estandar $IEEE754$.

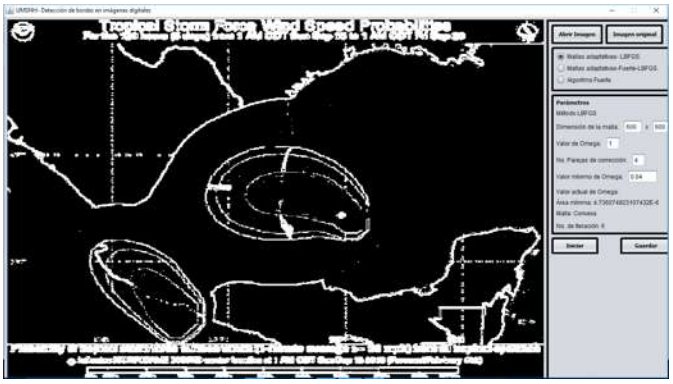

Figura 4.3: Interfaz gráfica desarrollada en Java.

Aunque esta implementación funciona, es lla preliminar pues se pueden realizar muchas mejoras, como lo son la optimización del código y el funcinamiento de la parte gráfica, así como incorporar métodos como el filtro de Perona-Malik.

4.1.4. Algoritmo Fuerte

El algoritmo aquí presentado surgió debido a una problemática presentada al intentar realizar la implementación del método LBFGS-B, pues en un intento de reducir la cntidad de datos para resolver el problema de optimización a gran escala involucrado al generar mallas adaptativas, fue que surgieron las ideas para el algoritmo Fuerte. La inspiración de este algoritmo son las características que tienen los bordes en la función de intensidad, pues intuitivamente los bordes se localizan en las discontinuidades más claras de la funcíon de intensidad, es por esto que el algoritmo aquí descrito está basado solamente en la función de intensidad, por lo que este algoritmo es totalmente ajeno a mallas adaptativas, BFGS, LBFGS y LBFGS-B.

La idea principal de este algotimo es plantear que una imagen está compuesta sólo por dos clases de regiones. La primera clase sería las regiones donde se encuentran los bordes y la otra clase son las regiones cuasi-homogéneas, esta clase de región debe ser entendida como las zonas de la imagen donde la variación de la función de intensidad I es menor, es decir, zonas de la imagen en las cuales I es constante o casi constante.

Así, para detectar bordes en una imagen entonces basta con detectar las regiones cuasi-homogéneas para anularlas (asignado un sólo color a estas regiones), pues por inercia, al anular las regiones que no son bordes se estarán detectando los bordes. Al hablar de detectar regiones de bordes o regiones cuasi-homogéneas debe quedar claro que se habla de detectar los pixeles de la imagen que se encuentren en los bordes o en una región cuasi-homogénea. Para realizarlo es necesario conocer la función de intensidad I de la imagen.

Es importante recordar que la función I de una imagen cualquiera es representada computacionalmente por una matriz en la cual cada entrada es un valor nuérico de 0 a 255 (en la escala de grises) que representa el color de un determinado píxel, por lo que es útil usar una matriz dual $\mathbf{M}$ con las mismas dimensiones que la matriz de la función de intensidad, pero con la diferencia de que todas las entradas de la matriz $\mathbf{M}$ tendrán el valor de 255 inicialmente, pues se usó el color blanco y negro para los bordes y regiones cuasi-homogéneas, respectivamente.

Sin pérdida de generalidad, supongase que la imagen tiene una resolución $n \times m$, esto implica que la matriz $\mathbf{M}$ será de dimensión $n \times m$. La figura 4.4 se muestran las vecindades usadas tanto para píxeles interiores y los que se encuentran en la frontera de la imagen. Por ejemplo en el inciso **e)**, es decir, para los pixeles interiores se usó una vecindad de 3×3, en todas las vecindades se tiene un píxel central y sus vecinos, el píxel central se denotará como $p_{i,j}$ en todas las clases de vecindades que se abordan en el algortimo aquídescrito.

Si la diferencia absoluta entre el píxel central y uno de sus vecinos es menor a

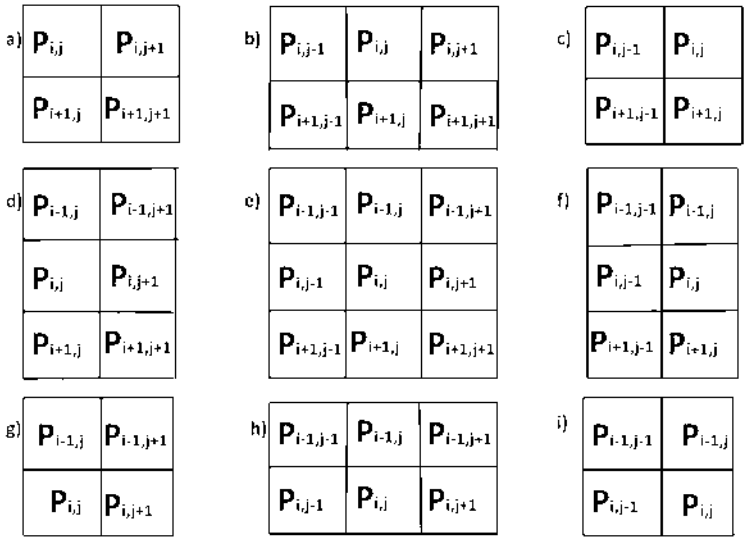

Figura 4.4: Los distintos casos de vecindades, **a)** esquina superior izquierda, **b)** frontera superior, **c)** esquina superior derecha, **d)** frontera izquierda, **e)** interior, **f)** frontera derecha, **g)** esquina inferior izquierda, **h)** frontera inferior y **i)** esquina inferior derecha.

un parámetro k dado, entonces, por ejemplo, en el caso de un píxel interior y usando la notación de la vecindad de la figura 4.4 **e)**, el píxel central $p_{i,j}$ y el píxel vecino $p_{i,j-1}$ son considerados de un color semejante por el parámetro k si

$$A_{i,j-1} = |p_{i,j-1} - p_{i,j}| <= k,$$

por tanto la vecindad será una región con poca variación en la función I si se cumple la expresión anterior para cada uno de los vecinos de $p_{i,j}$, en otras palabras, la vecindad será cuasi-homogénea si el píxel central y cada uno de sus vecinos tiene color semejante por el parámetro k, es decir, si se cumple que

$$A_{i,j-1}+A_{i-1,j-1}+A_{i-1,j}+A_{i-1,j+1}+A_{i,j+1}+A_{i+1,j+1}+A_{i+1,j}+A_{i+1,j-1} <= L,$$

donde $L = 8 * k$. Si se cumple la expresión anterior, entonces al píxel central dual de la matriz $\mathbf{M}$ se le asigna el valor 0 (color negro), indicando que ese píxel pertenece a una región cuasi-homogénea.

No sólo se aplica esto a los píxeles que están en el interior, este algoritmo es a cada píxel de la imagen, solamente hay que tener cuidado en que clase de vecindad se está, pues los píxeles de la frontera tienen 5 vecinos, los píxeles interiores

Figura 4.5: Imagen original.

tienen 8 vecinos, mientras que los pixeles esquina tienen sólo 3 vecinos.

El número de vecinos del píxel central y su localización en la matriz es lo que determina la forma de la desigualdad que se debe cumplir para considerar que el píxel central no pertenece a un borde, por ejemplo, para el píxel central que está en la esquina superior izquierda la vecindad esta ilustrada en el inciso **a)** de la figura 4.4, la desigualdad que debe cumplir para ser descartado como parte de los bordes (pertenece a una región cuasi-homogenea) es la siguiente

$$A_{i,j+1} + A_{i+1,j+1} + A_{i+1,j} <= L,$$

Figura 4.6: Resultado de la aplicación del algoritmo con $k = 17$.

Figura 4.7: Resultado de la aplicación del algoritmo con $k = 19$.

donde en este caso $L = 3k$. La figura 4.10 muestra el pseudoalgoritmo del algoritmo Fuerte. Aquí surge una interrogante importante, pues en principio no se

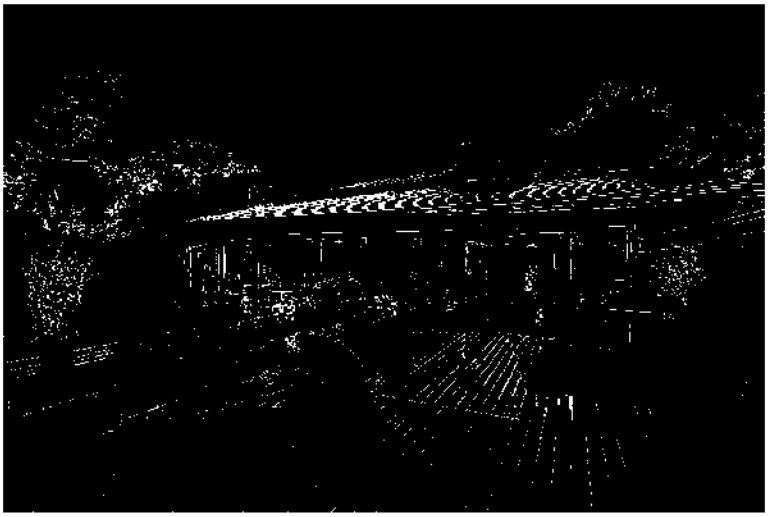

Figura 4.8: Resultado de la aplicación del algoritmo con $k = 24$.

Figura 4.9: Resultado de la aplicación del algoritmo con $k = 33$.

sabe que parámetro k usar y si existe un valor de k que se pueda usar para toda imagen. Sin embargo a través de la aplicación exhaustiva de este algoritmo se

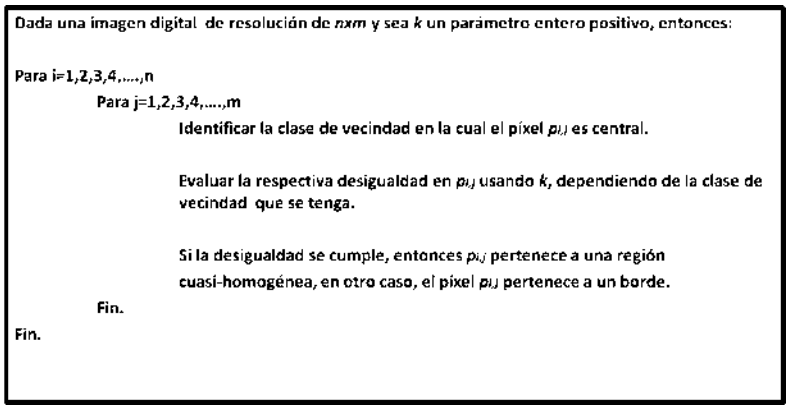

Figura 4.10: Pseudoalgoritmo del algoritmo Fuerte.

observó que los valores de k entre 4 y 33 funcionan muy bien, las figuras 4.6, 4.7, 4.8 y 4.9 muestran los resultados obtenidos al aplicar el algoritmo a la imagen 4.5 con los valores $k = 17$, $k = 19$, $k = 24$ y $k = 33$. Es claro que si el valor de k aumenta, entonces cada vez se detectaran menos bordes.

Por último, a continuación las figuras 4.12, 4.14, 4.16, 4.18, 4.20 y 4.22 muestran algunos resultados más obtenidos con el algoritmo Fuerte, los siguientes resultados son obtenidos a partir de algunas imágenes de resolución relativamente alta.

Figura 4.11: Imagen de un grupo de pinguinos.

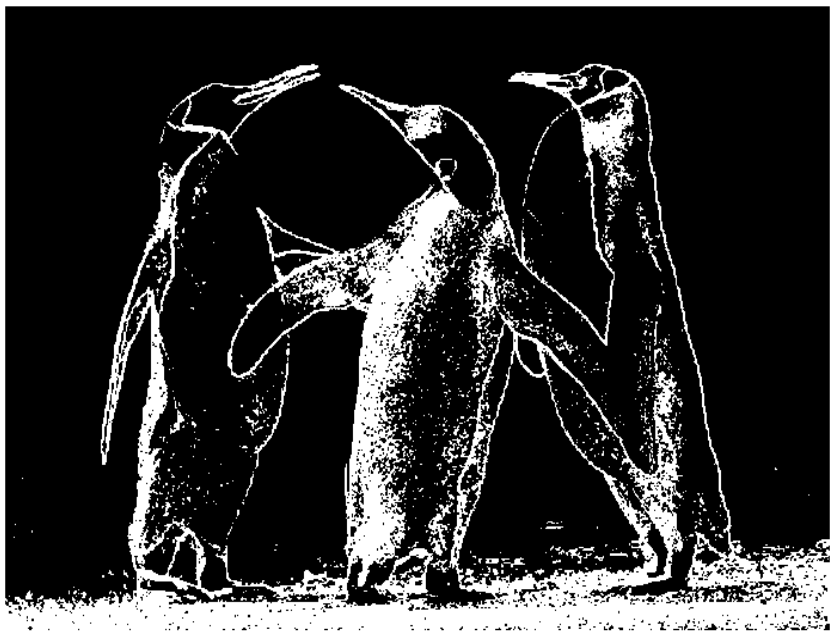

Figura 4.12: Resultado del algoritmo al aplicarlo a la imagen 4.11.

Figura 4.13: Imagen de un koala.

Figura 4.14: Resultado del algoritmo al aplicarlo a la imagen 4.13.

Figura 4.15: Imagen de una medusa marina.

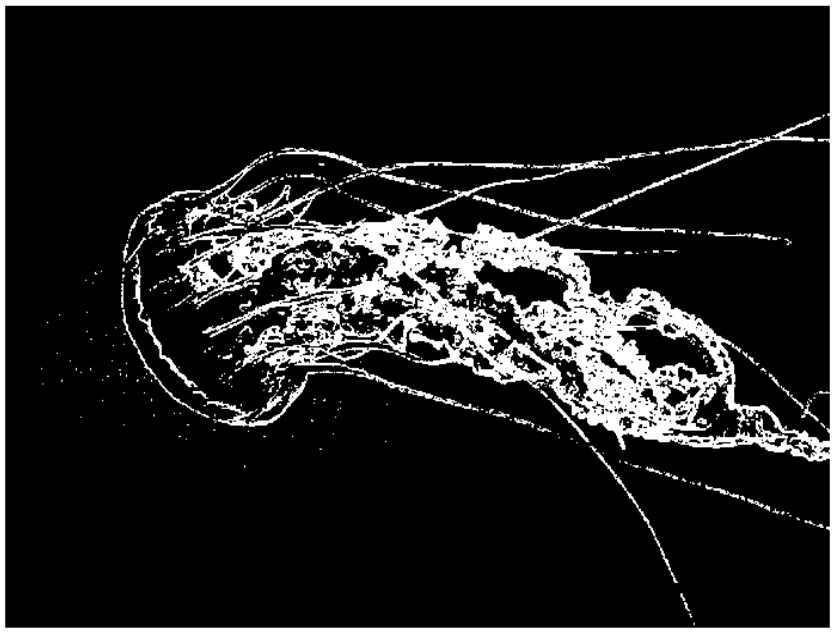

Figura 4.16: Resultado del algoritmo al aplicarlo a la imagen 4.15.

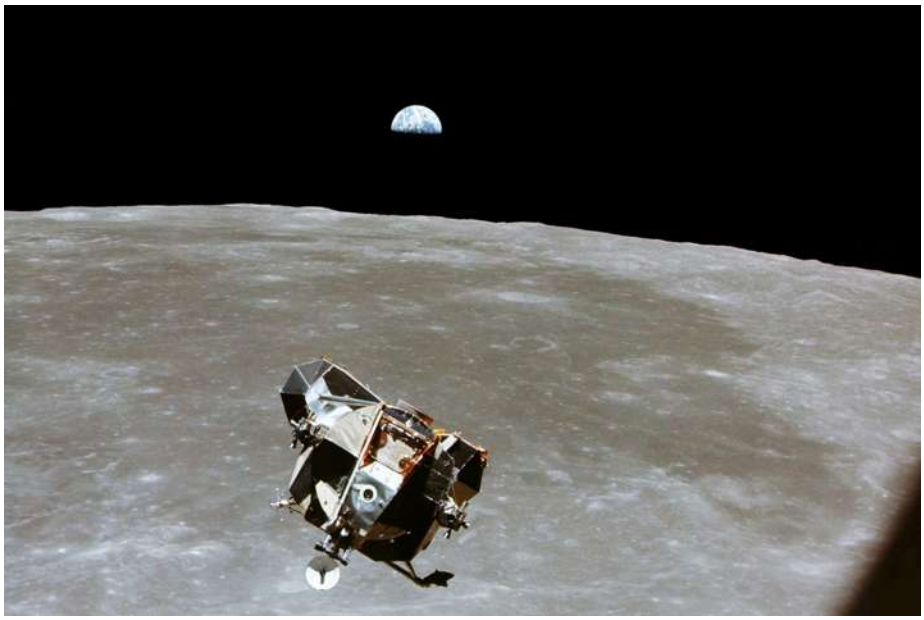

Figura 4.17: Imagen del alunizaje del Apolo 11.

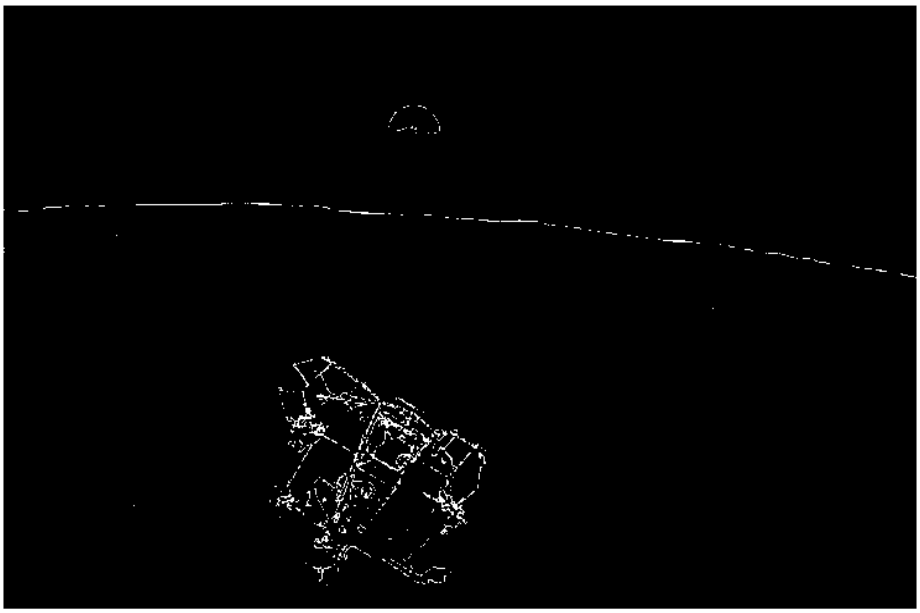

Figura 4.18: Resultado del algoritmo al aplicarlo a la imagen 4.17.

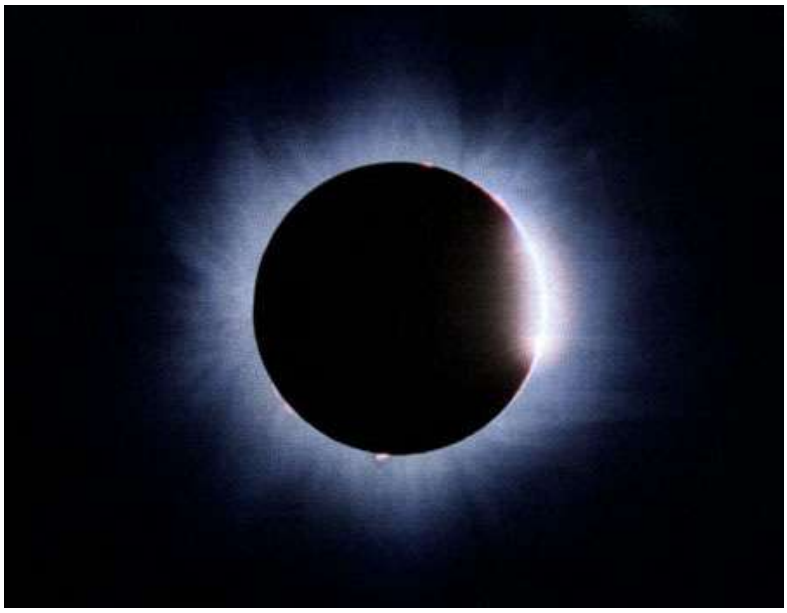

Figura 4.19: Imagen de un eclipse solar visto desde Cuitzeo, Michoacán.

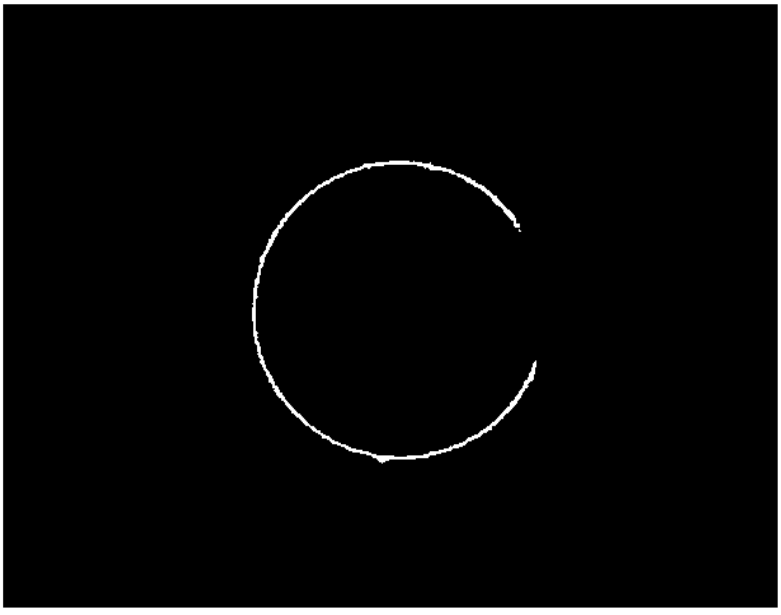

Figura 4.20: Resultado del algoritmo al aplicarlo a la imagen 4.19.

Figura 4.21: Imagen de un eclipse solar visto desde Cuitzeo, Michoacán.

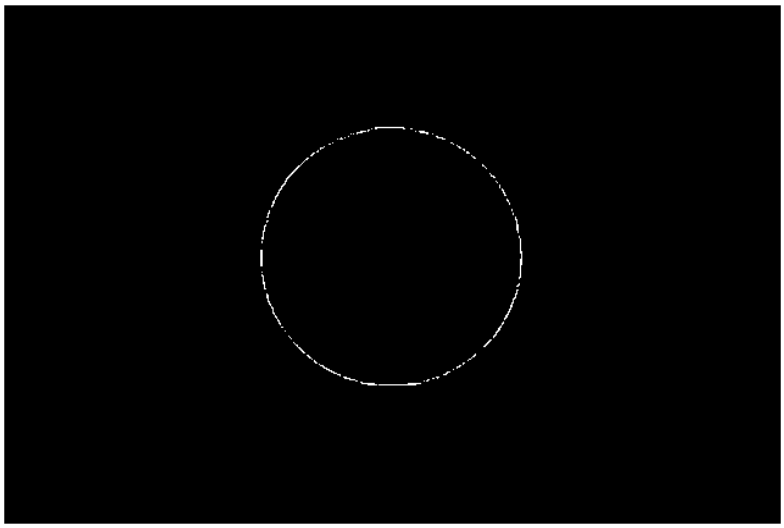

Figura 4.22: Resultado del algoritmo al aplicarlo a la imagen 4.21.

4.1.5. Resultados del método LBFGS

En esta sección se muestran las mallas adaptativas generadas con el método LBFGS, así como los recursos que se usaron para su visualización y finalmente una comparación con los resultados de los filtros de Canny y Sobel, el pseudo-codigo del algoritmo para tratar de obtener mallas convexas se muestra en la figura 4.23.

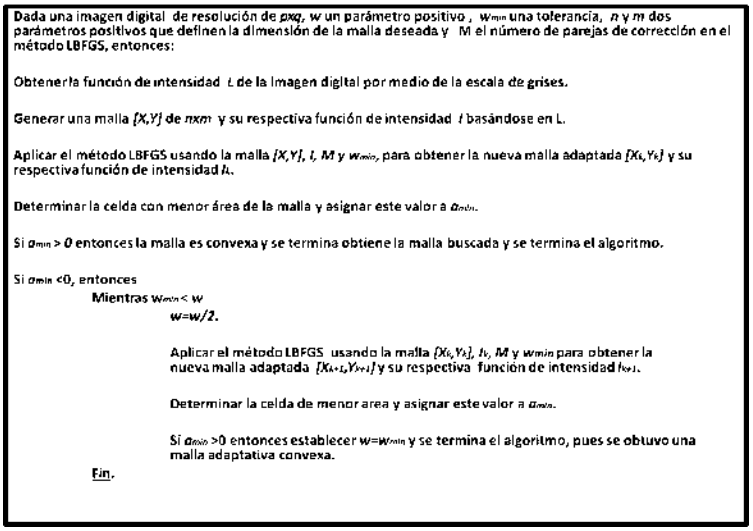

Figura 4.23: Pseudoalgoritmo del algoritmo usado para intentar obtener mallas convexas.

Figura 4.24: Imagen de original de Lenna Gray de resolución 220×229.

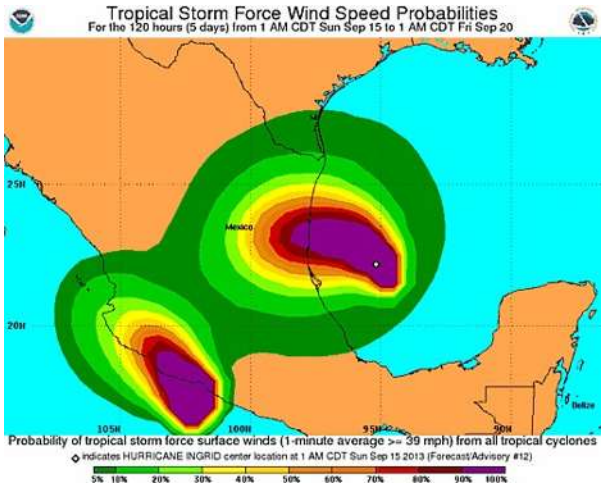

Figura 4.25: Imagen de original del huracan Manuel sobre México de resolución 500×400.

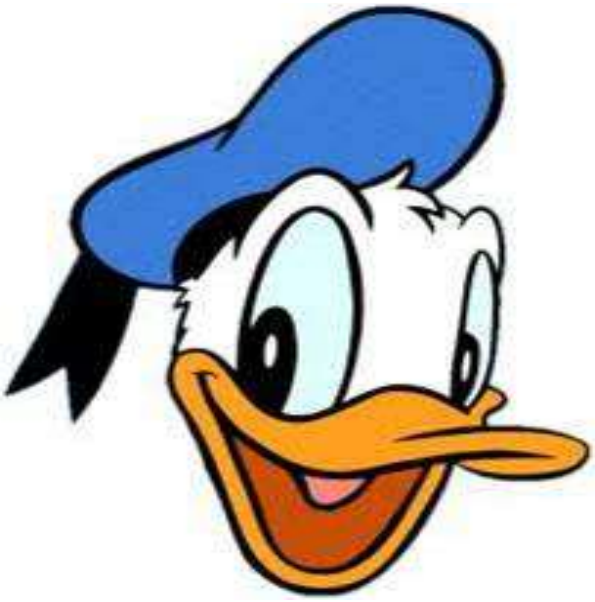

Figura 4.26: Imagen de original de resolución 202×250.

Figura 4.27: Imagen de original de resolución 2400×1600.

Los resultados mostrados en esta sección fueron generados a partir de cuatro imágenes únicamente, tales imágenes son las mostradas en las figuras 4.24, 4.25, 4.26 y 4.27, estas imágenes tienen una resolución de 220×229, 500×400,

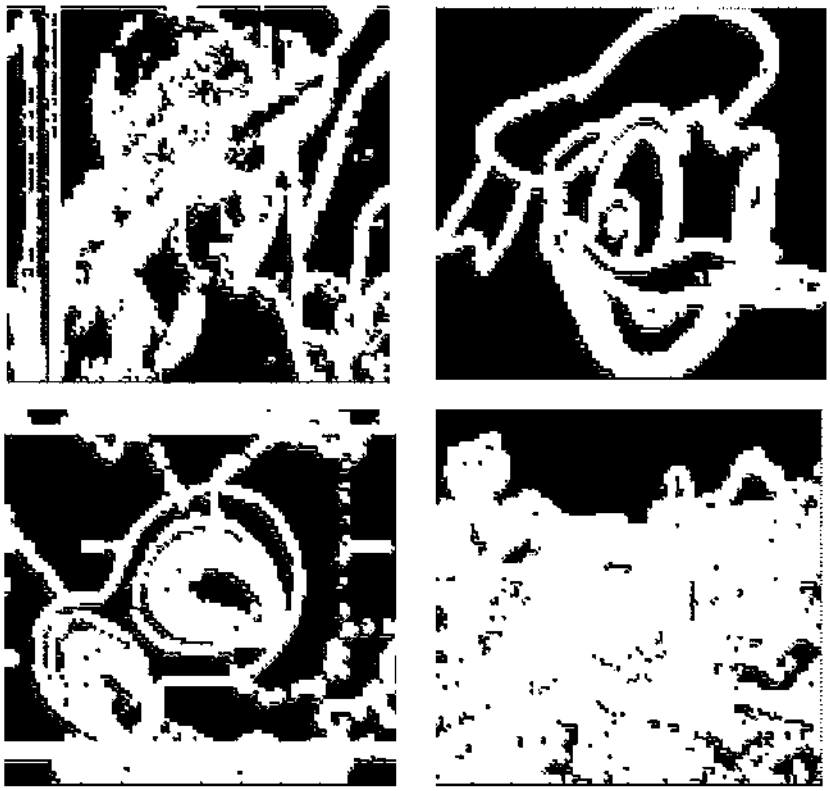

Figura 4.28: Mallas adaptativas de dimensión 100×100 de las imágenes 4.24, 4.26, 4.25 y 4.27.

202×250 y 2400×1600 pixéles, respectivamente. En la generación de todas las mallas adaptativas se usaron cuatro parejas de corrección, $\omega = 1$ y un valor mínimo de 0,004.

Es importante recalcar que si la imagen original tiene resolución de $n \times m$, debido a la forma que Matlab lee las imágenes para representarlas en forma matricial, entonces la matriz que representa a la imagen se modifica de manera que al graficarlas, las imágenes aparecen volteadas, por lo que se programó una función que acomodara la matriz para que al momento de graficar la imagen y la malla adaptativa estuvieran en una posición normal.

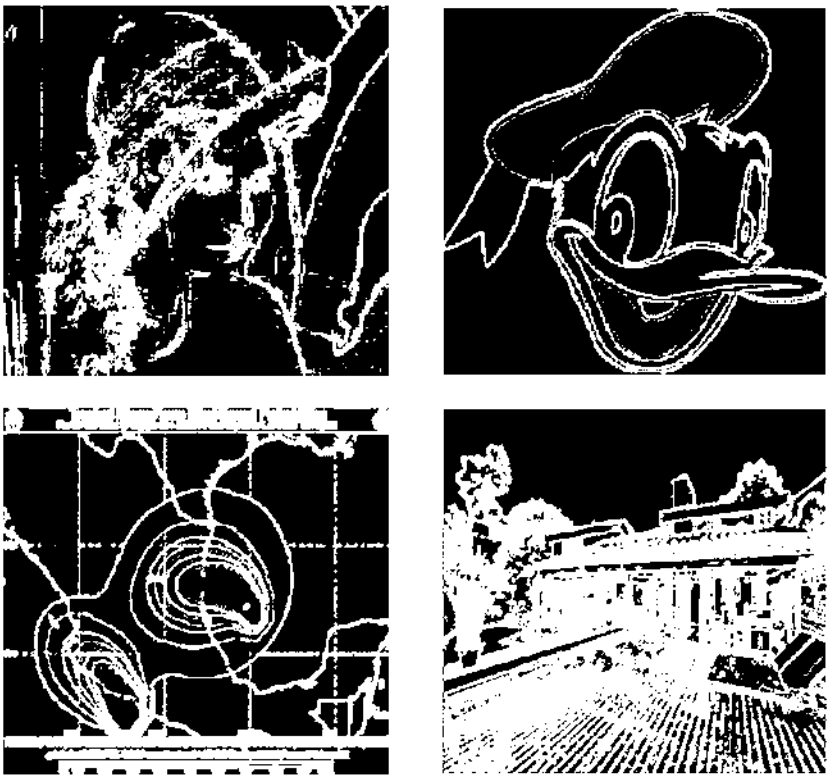

Figura 4.29: Mallas adaptativas de dimensión de 400×400 de las imágenes 4.24, 4.26, 4.25 y 4.27.

La figura 4.28 muestra las mallas adaptativas generadas con el método LBFGS de dimensión de 100×100 de las imágenes originales mencionadas, de igual manera, la figura 4.29 muestra las mallas adaptativas de las mismas imágenes, pero de dimensión 400×400, es decir, mallas adaptativas más finas. Puede observarse que se obtiene una mejor detección de los bordes entre más se aumente las dimensiónes de las mallas, para mostrar esto, se obtuvieron mallas adaptativas de 600×600, las cuales se muestran en las figuras 4.30, 4.31, 4.32 y 4.33.

Es importante mencionar que para mostrar las mallas adaptativas generadas se hizo uso de la función de distorsión mencionada en el capítulo de generación variacional de mallas adaptativas, pues debido a que la malla adaptativa está escalada en el cuadrado unitario (para poder aplicar la teoría de generación de mallas estudiada), entonces solamente se puede distinguir una malla adaptativa a simple vista si la malla no es muy fina, pues si una malla es muy fina entonces al graficarla solamente se vería un cuadrado de un sólo color. Las figuras 4.34 y 4.35 muestran ejemplos de mallas de dimensión de 300×300 donde si es posible distinguir un poco la malla adaptativa directamente, es decir, sin usar la

Figura 4.30: Malla adaptativa de dimensión de 600×600.

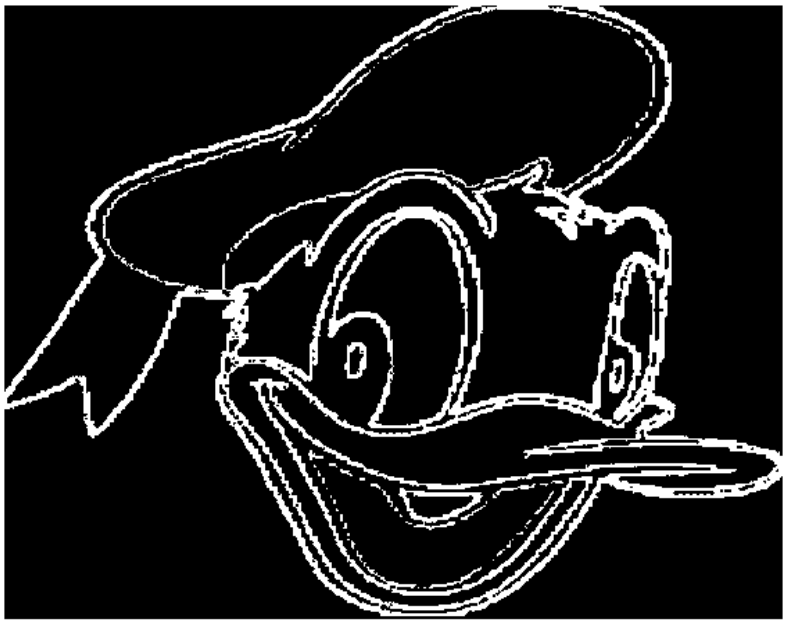

Figura 4.31: Malla adaptativa de dimensión de 600×600.

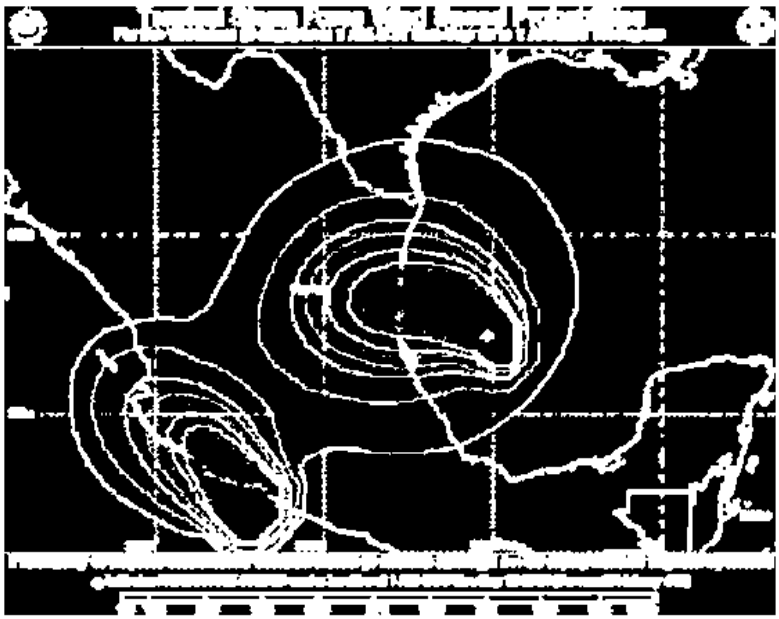

Figura 4.32: Malla adaptativa de dimensión de 600×600.

Figura 4.33: Malla adaptativa de dimensión de 600×600.

función de distorsión, pero para dimensiones mayores a 400×400 la malla no es distinguible.

La función de distorsión por su parte, usando $\lambda = \frac{1}{2}$, calcula que tan elongada esta una celda, pues en una celda que sea un cuadrado, la función de dispersión valdría uno en esa celda. Recordando que las mallas adaptativas tienen celdas elongodas cerca de los bordes, entonces por si solas y junto con la función de dispersión ya detectan bordes con cierta precisión, pues los valores de la función de dispersión se grafican en el centroide de cada celda.

La función de distorsión que se programó inicialmente consistió en una función que usa la función de distorsión para evaluarla en cada celda de la malla adaptativa. La función programada necesita un parámetro l, el cual es utilizado para determinar si una celda es suficientemente elongada para considerar si la debe mostrar como posible borde.

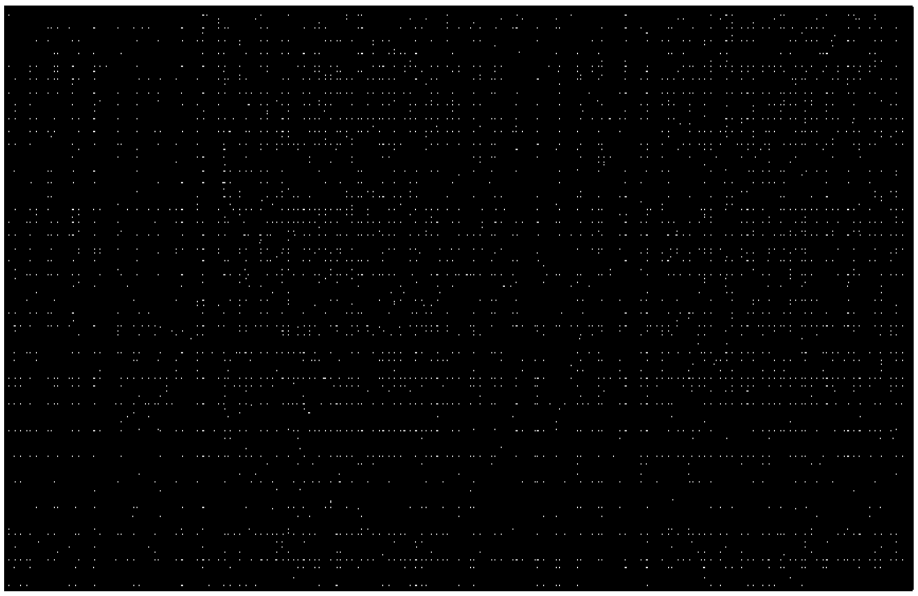

Figura 4.34: Malla adaptativa de la figura 4.24 de 300×300.

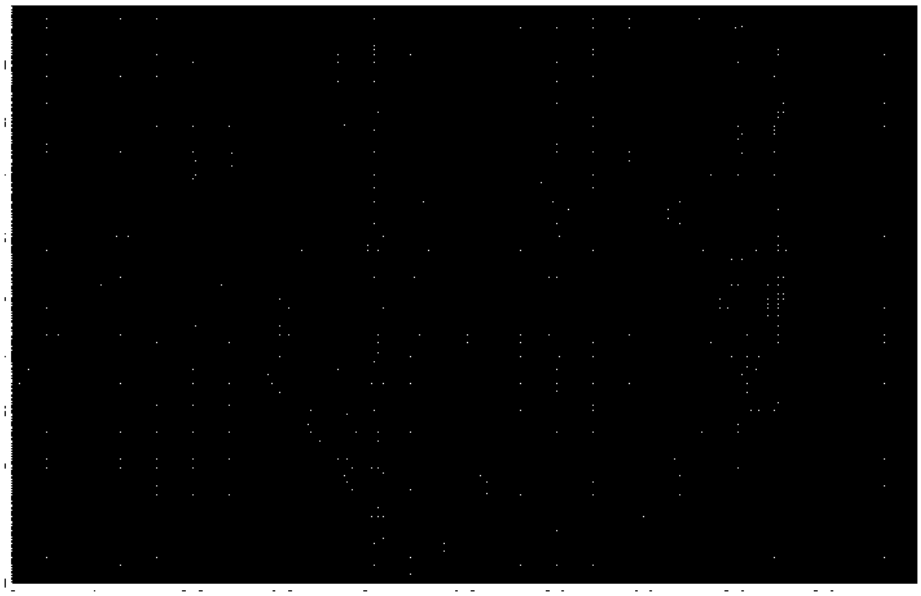

Figura 4.35: Malla adaptativa de la figura 4.26 de 300×300.

La función programada recibe como parámetros a la malla adaptativa, la función de intensidad de la malla adaptativa, el parámetro λ de la función de distorsión

y el parámetro l. Si el valor de la función de distorsión en una celda de la malla tiene un valor mayor o igual a l, entonces se calculan las coordenadas del centroide de la celda y a este punto se le asigna el valor de color blanco en la escala de grises, por otra parte, si el valor de la función de distorsión en la celda es menor a l, entonces al centroide de la celda se le asigna el valor del color negro de la escala de grises. Es así como se obtuvierón las imágenes que ayudaron a visualizar las malla adaptativas de altas dimensiones usando la función de distorsión.

Sin embargo, como se puede ver en las mallas generadas de la manera descrita con la función de distorsión, dependiendo de la imagen de la cual se genera la malla adaptativa, puede no tenerse mucha nitidez para visualizar la malla adaptativa con la función de distorsión. Es por esto que se programó otra función que generaba también imágenes de las mallas adaptativas usando la función de distorsión para visualizarlas, pero a diferencia de la primera función, la cual asiganaba el color blanco o negro al centroide de la celda, otra alternativa es asignar el color blanco o negro a la propia celda segun sea el caso, dependiendo del valor de la función de distorsión y el parámetro l, para así visualizar la malla adaptativa con celdas de color blanco o negro.

En las figuras 4.36 y 4.37 se puede apreciar las mismas mallas adaptativas de dimensión 100×100 obtenidas, pero con la diferencia de que se utilizo la función de distorsión modificada para visualizarlas. Las figuras 4.38, 4.39, 4.40 y 4.41 muestran las mallas adaptativas de 400×400.

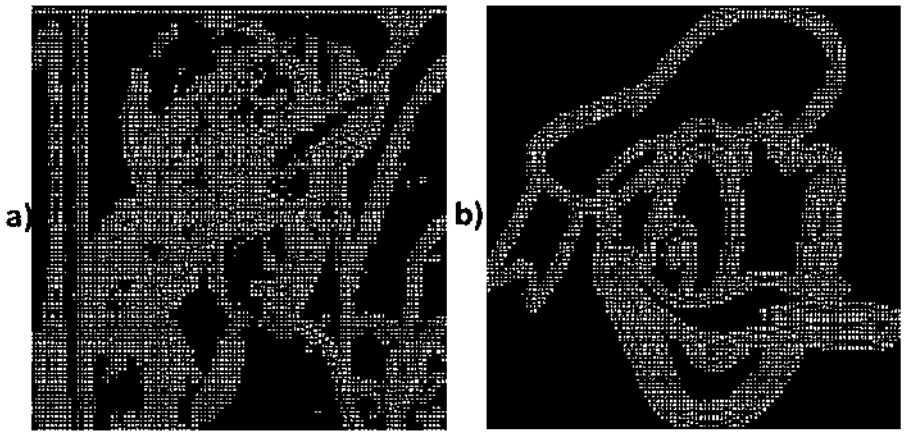

Figura 4.36: Imágenes de las mallas adaptativas de 100×100 generadas con la función de distorsión, asignando un color a la celda en lugar de al centroide de la celda.

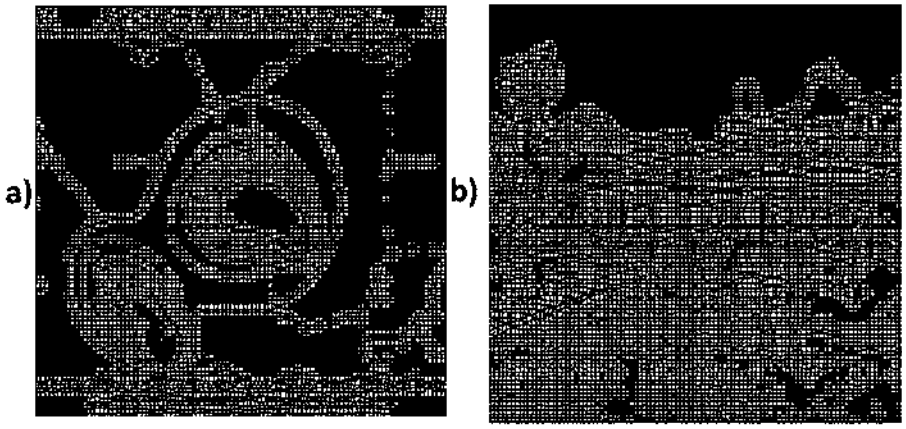

Figura 4.37: Imagen de las mallas adaptativas de 100×100 generada con la función de distorsión, asignando un color a la celda en lugar de al centroide de la celda, **a)** Imagen 4.25 y **b)** Imagen 4.27.

En las figuras 4.42, 4.43, 4.44 y 4.45 se muestran las comparaciones entre las mallas adaptativas de 600×600 generadas por medio del método LBFGS y los resultados de la detección de bordes por medio de los filtros de Sobel y Canny.

Por último, es importante mencionar que aunque la aplicación del LBFGS dió excelentes resultados, existe una desventaja en el uso de las mallas mostradas en esta sección, tal inconveniente es que sólo en pocas imágenes se obtuvieron mallas convexas. La no convexidad de una malla repercute directamente en la

Figura 4.38: Imagen de la malla adaptativa de 400×400 generada con la función de distorsión, asignando un color a la celda en lugar de al centroide de la celda.

Figura 4.39: Imagen de la malla adaptativa de 400×400 generada con la función de distorsión, asignando un color a la celda en lugar de al centroide de la celda.

estabilidad del esquema en diferencias usado en el modelo de Perona-Malik.

Figura 4.40: Imagen de la malla adaptativa de 400×400 generada con la función de distorsión, asignando un color a la celda en lugar de al centroide de la celda.

Figura 4.41: Imagen de la malla adaptativa de 400×400 generada con la función de distorsión, asignando un color a la celda en lugar de al centroide de la celda.

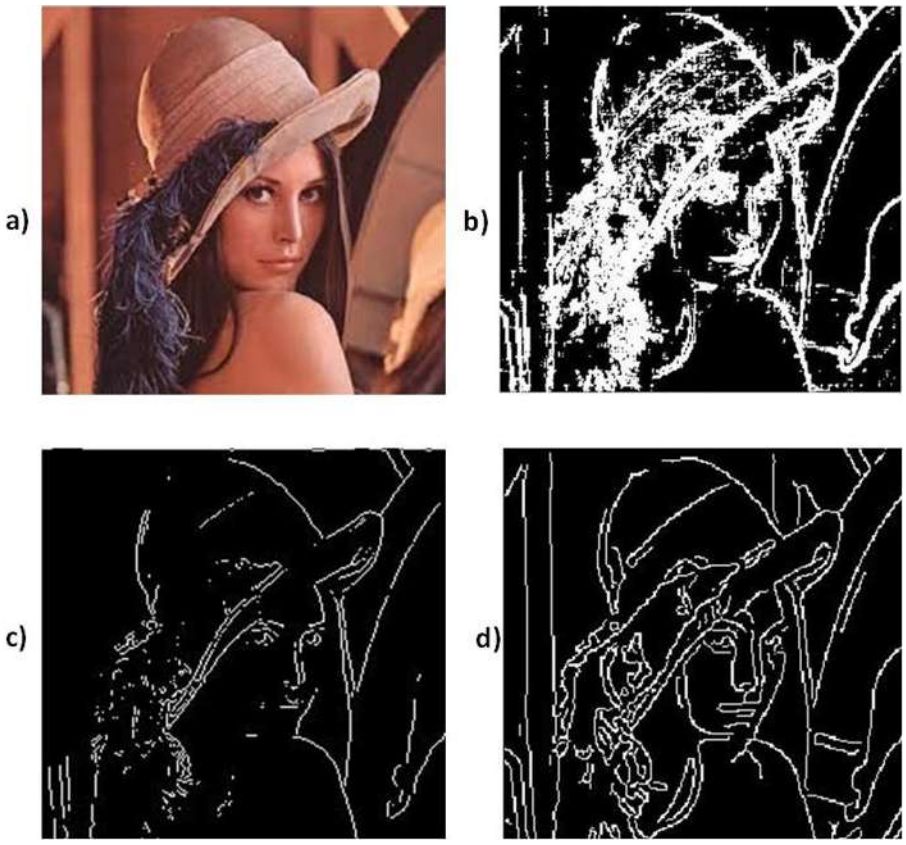

Figura 4.42: **a)**Imagen original, **b)**malla adaptativa de 600×600, **c)**filtro de Sobel y **d)**filtro de Canny.

4.1.6. Resultados del método LBFGS y el algoritmo Fuerte

La no convexidad de las mallas adaptativas obtenidas con el método LBFGS fue la motivación de buscar otras alternativas. Después de realizar algunas pruebas con el algoritmo Fuerte en imágenes digitales, surgió la idea de aplicar en conjunto el algoritmo Fuerte y el LBFGS. La forma de aplicar estos dos métodos es aplicar el algoritmo Fuerte a la imagen y despues el método LBFGS a la imagen resultante.

El resultado fue sorpresivo, pues se encontró que al concatenar el algoritmo Fuerte y el LBFGS se obtenian mallas adaptativas convexas, que aunque no detectan de manera ejemplar los bordes de la imagen, la ventaja es la convexidad, pues con la convexidad se obtiene estabilidad en el esquema en diferencias.

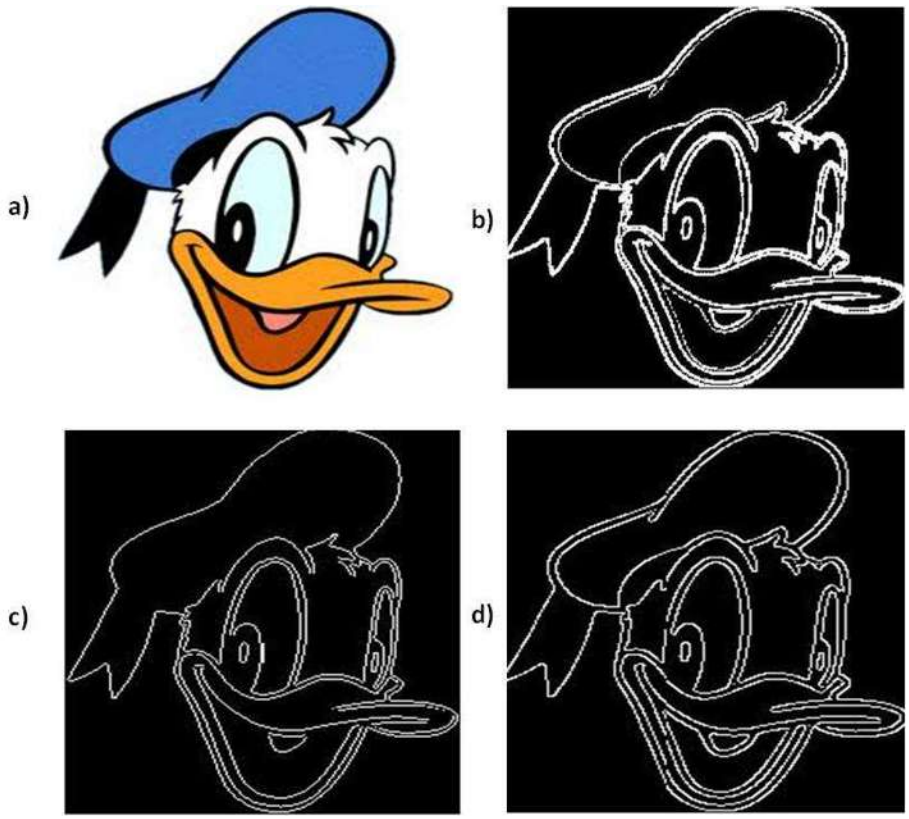

Figura 4.43: **a)**Imagen original, **b)**malla adaptativa de 600×600, **c)**filtro de Sobel y **d)**filtro de Canny.

Sin embargo existe un detalle importante para obtener convexidad en las mallas adaptativas con esta nueva metodología, hay que recordar que el algoritmo Fuerte requiere de un parámetro k para establecer el criterio utilizado para determinar que píxeles forman parte de un borde. Si k no es el parámetro indicado, pueden generarse mallas adaptativas no convexas.

Existen casos en los que se encuentra un parametro k con el cual mallas de dimensión entre 100×100 y 600×600 son convexas pero con una dimensión mayor no son convexas, por ejemplo, en la imagen de lenna se generaron mallas con un parámetro $k = 10$ y de dimensiones 200×200, 400×400 y600×600, pero al generar la malla de dimensión 1000×1000 con el mismo parámetro $k = 10$,

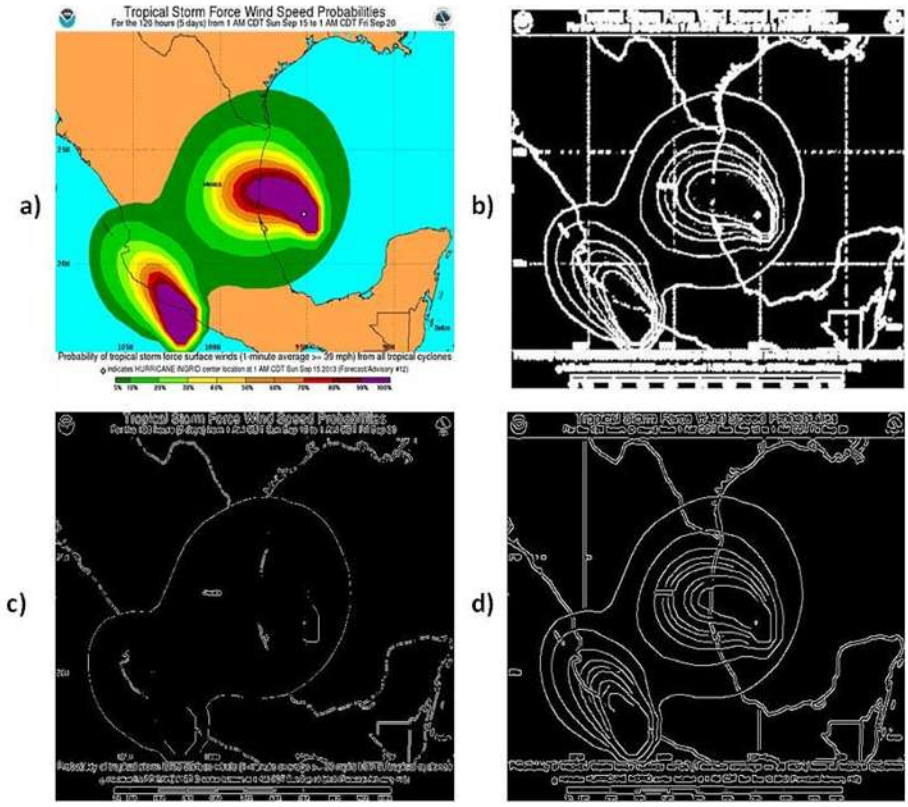

Figura 4.44: **a)**Imagen original, **b)**malla adaptativa de 600×600, **c)**filtro de Sobel y **d)**filtro de Canny.

la malla obtenida era no convexa.

Las figuras 4.46, 4.47, 4.48 y 4.49 muestran los resultados de las mallas adaptativas generadas a partir de las imágenes 4.24, 4.26, 4.25 y 4.27, aplicando el algoritmo Fuerte y posteriormente el método LBFGS.

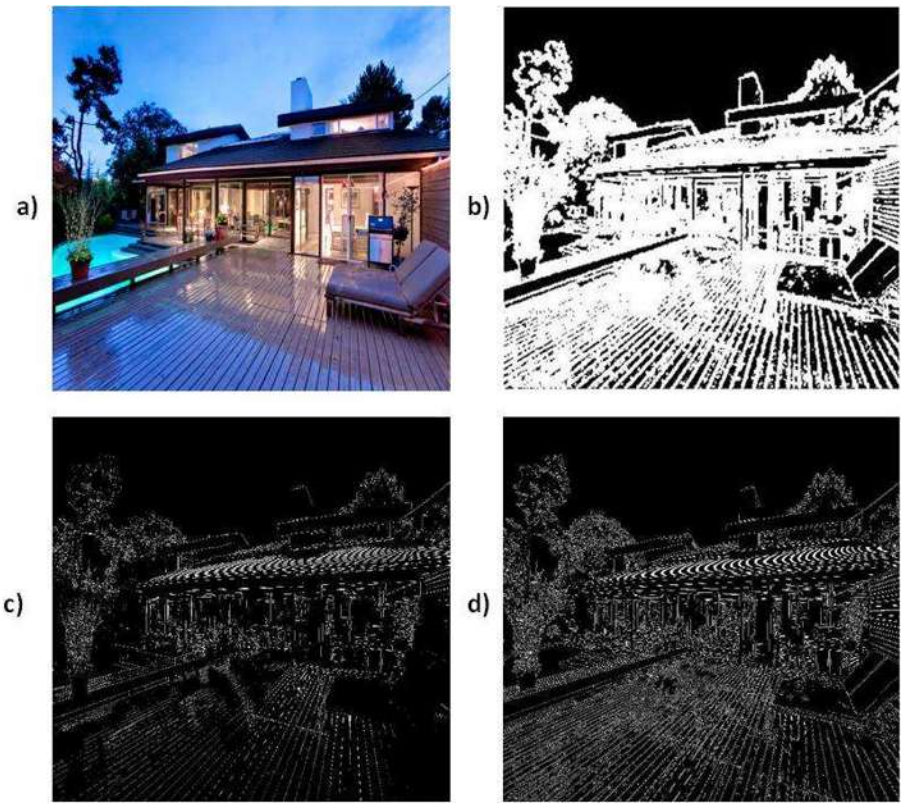

Figura 4.45: **a)**Imagen original, **b)**malla adaptativa de 600×600, **c)**filtro de Sobel y **d)**filtro de Canny.

4.1.7. Resultados del modelo Perona-Malik

Al aplicar diferencias finitas al modelo de Perona -Malik se encontraron algunas desventajas en la implementación, principalmente el tiempo de ejecución de este proceso, pues el número de pasos en el tiempo requeridos para lograr estabilidad en el esquema FTCS para este problema es muy alto. Al hacer pruebas con distintas imágenes con distintas mallas adaptativas, en promedio se requerian 400000 pasos en el tiempo, el cálculo de cada paso en el tiempo para mallas adaptativas convexas de entre 400×400 y 1000×1000 tardaba entre 0.4 seg y 1.2 seg, esto implica que el proceso tardaria desde 1.9 dias hasta 4.6 dias en terminar.

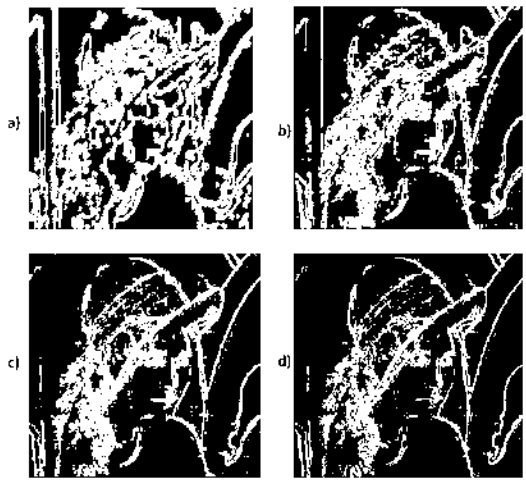

Figura 4.46: Mallas adaptativas generadas con $k = 10$ y de dimensiones **a)**200×200, **b)**400×400, **c)**600×600 y **d)**1000×1000.

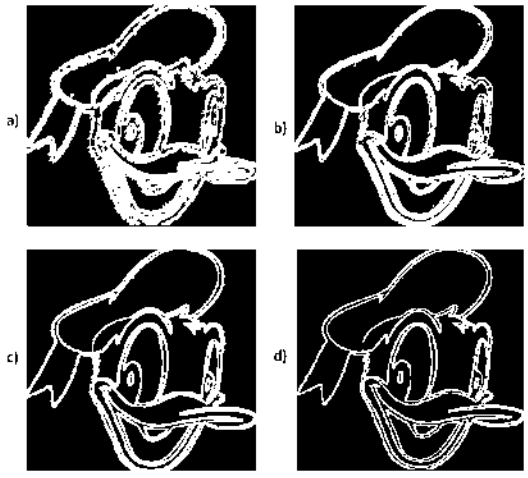

Figura 4.47: Mallas adaptativas generadas con $k = 19$ y de dimensiones **a)**200×200, **b)**400×400, **c)**600×600 y **d)**1000×1000.

Fue por esta razón que se sólo se obtuvieron los resultados de 400 pasos en el tiempo. Las figuras 4.50, 4.51, 4.52 y 4.53 se muestran los resultados del modelo

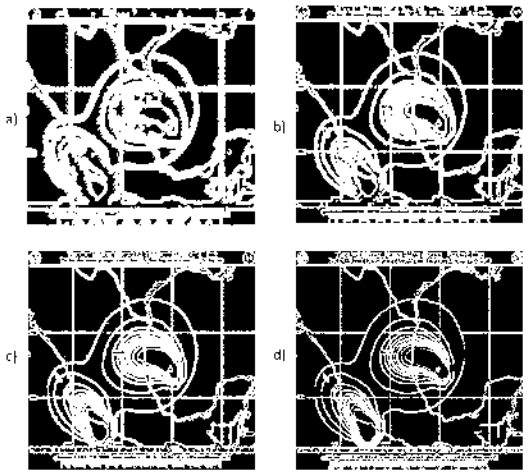

Figura 4.48: Mallas adaptativas generadas con $k = 6$ y de dimensiones **a)**200×200, **b)**400×400, **c)**600×600 y **d)**1000×1000.

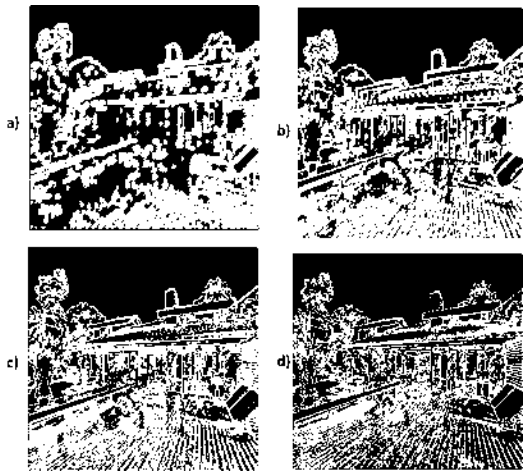

Figura 4.49: Mallas adaptativas generadas con $k = 11$ y de dimensiones **a)**200×200, **b)**400×400, **c)**600×600 y **d)**1000×1000.

de Perona-Malik para mallas adptativas convexas de 400×400 obtenidas con el algoritmo Fuerte y LBFGS, Los parámetros para obtener las mallas son los mismos que se usaron en la sección anterior y en el modelo de Perona-Malik se

establecieron los parámetros $\lambda = 0{,}4$ en el intervalo de tiempo $[0, 10^{-4}]$.

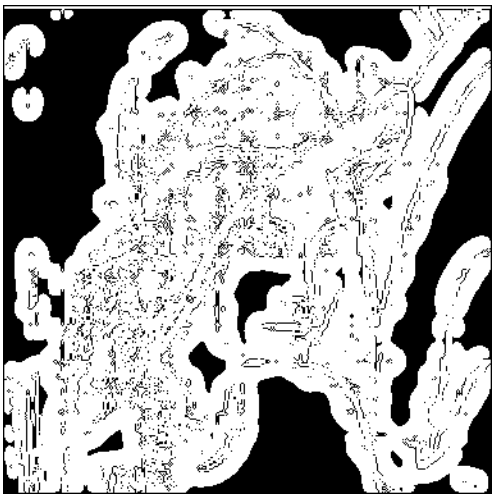

Figura 4.50: Detección de bordes con el modelo de Perona-Malik en la imagen 4.24.

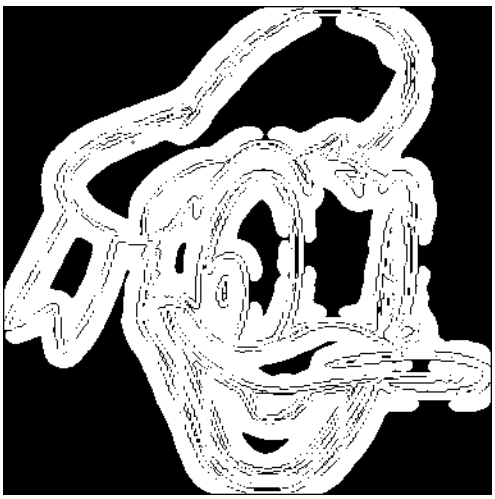

Figura 4.51: Detección de bordes con el modelo de Perona-Malik en la imagen 4.26.

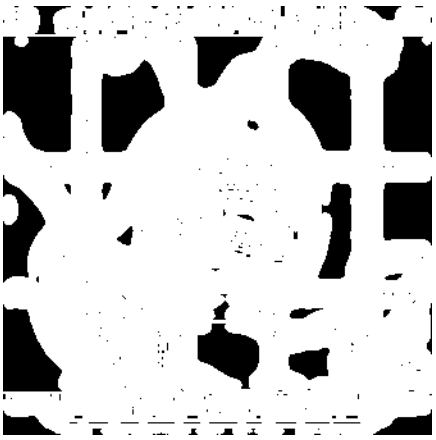

Figura 4.52: Detección de bordes con el modelo de Perona-Malik en la imagen 4.25.

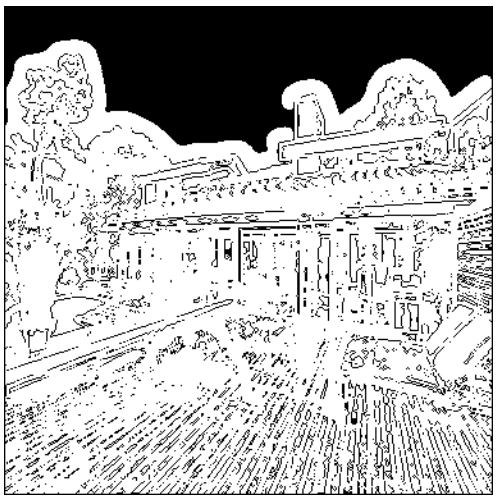

Figura 4.53: Detección de bordes con el modelo de Perona-Malik en la imagen 4.27.

Se puede observar en las figuras 4.50, 4.51, 4.52 y 4.53 que la detección de bordes no es óptima, esto se debe a obtener los resultados despues de 400×400 pasos

en el tiempo.

Capítulo 5

Conclusiones

5.1. Conclusiones

5.1.1. Conclusiones generales

La programación de la teoría estudiada durante esta tesis fue sin duda un trabajo que llevo un tiempo considerable,pues la gran mayoria de las funciones utilizadas en este trabajo son implementaciones propias, es decir, no se hizo sólamente uso de funciones propias de Matlab para la detección de bordes.

La programación hecha en Matlab tiene ventajas y desventajas, pues aunque Matlab es un software que tiene un gran potencial numérico, para los objetivos de este trabajo no resultó muy útil. Al usar Matlab como lenguaje de programación, pues se presentaron principalmente dos problemas en la ejecución de las rutinas: tiempo de ejecución muy alto y un uso considerable de recursos computacionales, esto no pasó en la interfaz desarrollada en Java.

La naturaleza del problema de esta tesis es la causa de que una implementación por medio de Matlab no sea un camino recomendable. Sin embargo, Matlab es una herramienta muy útil para hacer prototipos, es decir, para poder visualizar el funcionamiento de la teoría en la práctica de manera relativamente rapida y fue por esta razón que se optó por la implementación en de la teoría en Matlab, pues en un principio no se tenia una idea concreta de los posibles resultados en la detección de bordes, una vez visto el funcionamiento de la teoría es que se comenzó con el desarrollo en Java.

A la interfaz preliminar hecha en Java le hace falta todavia muchas mejoras, pues la implementación no es fácil, es decir, para desarrollar una implementación cada vez más completa se requiere mucho trabajo, y por tanto mucho tiempo de desarrollo.

5.1.2. Detección de bordes mediante el algoritmo Fuerte

La inspiración para plantear este algoritmo fueron las complicaciones presentadas durante la implementación del método LBFGS-B y la fución de intensidad, por lo que desde un inicio no se tenia contemplado este algoritmo. Aunque este algoritmo está únicamente basado en la función de intensidad, los resultados en la detección de bordes fueron mucho mejores de lo esperado.

Las ventajas de este algoritmo son que tiene un tiempo de ejecución muy bajo, pues se trata de un algoritmo de $\mathbf{O(n)}$, siendo n el número de píxeles de la imagen, en cambio si se quiere interpretar el orden de este algoritmo en terminos de la resolución de la imagen, entonces para una imagen de resolución de $n \times m$, este algoritmo sería de orden $\mathbf{O(nm)}$. Este algoritmo presenta además bajo costo computacional y por lo tanto se puede aplicar a imágenes de alta resolución. Lo único restante es investigar que parámetro k es el indicado para una imagen dada.

Hasta la útlima edición de este trabajo, no se encontró ningún articulo donde este algoritmo ya hubiera sido palnteado.

5.1.3. Detección de bordes mediante mallas adaptativas

En la detección de bordes por medio de mallas adaptativas, se obtuvieron excelentes resultados. Hay que recordar que se generaron mallas adaptativas de dos maneras: por medio del método LBFGS y por medio de un método compuesto por el algoritmo Fuerte y el método LBFGS. Los resultados fueron mejor de lo esperado en ambos casos, aunque si lo que se busca son mallas adaptativas convexas, entonces es mejor llevar a cabo el algoritmo Fuerte y el LBFGS en conjunto.

La principal problemática es la implementación del método LBFGS, esto se debe al problema de optimización a gran escala. Dado que día con día las imágenes tienen una mayor resolución, entonces es necesario desarrollar implementaciones eficientes de este método. La implementación de estos métodos en un lenguaje de programación distinto a Matlab es el primer paso.

5.1.4. Detección de bordes mediante el modelo de Perona-Malik

Los resultados obtenidos con la mallas adaptativas fueron mejores de lo esperado, sin embargo, los resultados del modeo de Perona-Malik no fueron lo que se esperaba. Aunque esto se debe a la interrupción del proceso, el hecho de que se requiera un tiempo muy grande para que el modelo de Perona-Malik comience a generar buenos resultados es la principal desventaja para generar buenos

resultados, pues para considerar que un método es eficiente, también entra en juego la rapidez del método.

El tiempo requerido para obtener una buena detección de bordes fue la principal desventaja de este modelo, pues dependiendo de la dimensión de la malla, eran requeridos al menos 100000 pasos en el tiempo para lograr estabilidad, esto fue algo sorpresivo, pues fue hasta las corridas del método que surgió este inconveniente.

La falta de un número de Courant para una mejor determinación del número de pasos en el tiempo requeridos es algo que no permitio obtener mejores resultados, sin duda, en un incio se esperaba más de este método. Lo que se debe intentar para mejorar este método es: un análisis del número de Courant, así como la optimización del método, no sólo en el código, sino también recurriendo a opciones computacionales como paralelizar el proceso, para así ver si el método mejora.

5.1.5. Trabajo a futuro

Los planes a futuro son principalmente comenzar la investigación del número de Courant para el esquema empleado en el modelo de Perona-Malik, así como la implementación empleando diferencias finitas generalizadas , donde de igual manera se tiene pensado lograr un funcionamiento óptimo, es decir, donde la aplicación del modelo tenga un tiempo de ejecución razonable.

Otro objetivo es realizar mejoras a la versión preliminar de la interfaz gráfica desarrollada en Java, así como incluir métodos que por el momento requieren de más investigación, tal como el uso de diferencias finitas en mallas adaptativas empleando el filtro de Perona-Malik.

El seguimiento del estudio de las mallas adaptativas para la detección de bordes es otro objetivo que puede ser visto como trabajo a futuro, pues las mallas adaptativas dan resultados razonables por si solas.

Por último, también es recomendable llevar a cabo distintas aplicaciones de la detección de bordes por medio de mallas adaptativas o el algoritmo Fuerte, pues es muy posible que haya problemas en áreas de la ciencias donde esta herramienta computacional resulte eficaz.

Bibliografía

[1] John Canny. A computational approach to edge detection. PAMI-8, 1986.

[2] Mónica Zaima Víquez Cano. *Un sistema para la genración de mallas armónicas*. Tesis de Maestría, UNAM, D.F., México, 2008.

[3] Carlos A. Júnez Ferreira. *Algoritmos para la reduciión de ruido Gaussiano y sal y pimienta en imágenes digitales*. Tesis de Doctorado, UMSNH, Morelia, Michoacán, Mexico, 2011.

[4] Guilmer Ferdinand Gonzalez Flores. *Un sistema automatico para la generación numérica de mallas basado en los nuevos funcionales de área y suavidad*. Tesis de Maestría, UNAM, D.F., México, 2003.

[5] Pablo M. Fernández Valdez Erika Ruiz Díaz y Gerardo Tinoco Guerrero Francisco J. Domínguez Mota, José G. Tinoco Ruiz. An heuristic finite difference scheme on irregular plane regions. 2014.

[6] Bernd Jähne. *Digital image processing*. University of Heidelberg, Alemania, 2001.

[7] Ciyou Zhu Jorge Nocedal, Richard Byrd. A limited memory algorithm for bound constrained optimization. 1994.

[8] Jorge Nocedal. *Numerical optimization*. segunda edición, 2006.

[9] Francisco J. Domínguez Mota Pablo Barrera Sánchez, Longina Castellanos Noda. Adaptive discrete harmonic grid generation. 2007.

[10] Guilmer Gonzalez Flores y José G. Tinoco Ruiz Pablo Barrera Sánchez, Francisco J. Domínguez Mota. Generating quality structured convex grids on irregular regions. 34, 2009.

[11] Jitendra Malik Pietro Perona. Scale-space and edge detection using anisotropic diffusion. 12, 1990.

[12] Sobel. A 3×3 isotropic gradient operator for image processing. 1968. www.researchgate.net/publication/239398674_An_Isotropic_3_3_Image_Gradient_Operator.

[13] Pablo Michel Fernández Valdez. *Un esquema directo de diferencias finitas para la solución numérica de ecuaciones elípticas en regiones irregulares del plano.* Tesis de Licenciatura,UMSNH, Morelia, Michoacán, México, 2012.

[14] Randall J. Le Veque. *Finite difference methods for ordinary and partial differential equations.* Society for industrial and applied mathematics, Philadelphia, 2007.

[15] Joachim Weickert. *Anisotropic diffusion in image processing.* Department of computer science, University of Copenhagen, Dinamarca, 1998.