UNIVERSIDAD DE CHILE
FACULTAD DE CIENCAS FÍSICAS Y MATEMÁTICAS
DEPARTAMENTO DE CIENCIAS DE LA COMPUTACIÓN

RECONOCIMIENTO DE BORDES EN IMÁGENES APLICADO A ANILLOS DE ÁRBOLES

MEMORIA PARA OPTAR AL TÍTULO DE INGENIERO CIVIL EN COMPUTACIÓN

PABLO AGUSTÍN AGUILAR VERGARA

PROFESORA GUÍA:
SRA. NANCY HITSCHFELD KAHLER

MIEMBROS DE LA COMISIÓN:
SRA. MARÍA CECILIA RIVARA ZÚÑIGA
SR. MAURICIO PALMA LIZANA

SANTIAGO DE CHILE
ENERO 2008

RECONOCIMIENTO DE BORDES EN IMÁGENES APLICADO A ANILLOS DE ÁRBOLES

El objetivo de esta memoria es construir una aproximación, utilizando polilíneas, de los anillos de un árbol, a partir de una imagen de los mismos. Para lograr esto, se busca crear un método mixto, que consiste en utilizar primero filtros para eliminar ruido y otras imperfecciones de las imágenes y resaltar los bordes en las mismas, y luego mallas geométricas para construir las polilíneas que representan los anillos.

Este trabajo comienza analizando diversos trabajos relacionados con el tema, incluyendo dos memorias realizadas anteriormente: la primera utiliza mallas geométricas para detectar la posición de los anillos, y la segunda los detecta mediante el uso del filtros.

A continuación se selecciona un conjunto de filtros de imágenes, criterios para la inserción de puntos y un algoritmo de generación de mallas y se diseña un algoritmo de construcción de las polilíneas para representar los anillos, basándose en la Transformada de Hough Generalizada. Luego se elabora un diseño orientado a objetos que permita agregar fácilmente nuevos filtros, algoritmos de generación de mallas, mecanismos de inserción de puntos y estrategias para construir las polilíneas. La herramienta implementada en Java incluye los filtros de normalización, ecualización, umbral, doble umbral, umbral adaptativo y gaussiano; el algoritmo de generación de mallas de Delaunay, los mecanismos de inserción de puntos del punto central y el punto de mayor error, y el algoritmo de construcción de polilíneas basado en la Transformada de Hough Generalizada.

Finalmente, se analizan los resultados de la aplicación de los algoritmos descritos, en forma individual y conjunta, tanto en imágenes fabricadas como reales, para comparar y evaluar su desempeño; aquí se ve que el método mixto desarrollado se desempeña correctamente y resulta ser una buena solución al problema enfrentado para árboles cuyos anillos son similares entre sí.

AGRADECIMIENTOS

Agradezco a la profesora Nancy Hitschfeld por su paciencia, dedicación y valiosos consejos durante el desarrollo de esta memoria.

Agradezco también a mi familia por haberme apoyado incondicionalmente y haberlo dado todo por ayudarme a conseguir mis metas, no sólo durante este año, sino que en toda mi carrera y mi vida.

Finalmente, agradezco especialmente a Carolina, la persona que he tenido al lado en los momentos más difíciles y también los más gratos, y que ha sido un pilar fundamental de mi desarrollo como profesional, y también como ser humano, por su dedicación, paciencia y amor.

A todos ustedes, gracias de todo corazón.

Índice general

Capítulo 1

Introducción

El procesamiento de imágenes es, como su nombre lo indica, un proceso que toma una imagen como entrada y da como resultado otra imagen, o bien un conjunto de características de la misma; de esta manera, se puede simplificar la cantidad de recursos necesarios para describirla, descartando información que puede ser considerada como menos relevante. De todas las características que pueden ser extraídas de una imagen, una de ellas son sus bordes.

Los bordes dentro de una imagen se pueden definir como los puntos donde su intensidad de luminosidad cambia abruptamente; éstos usualmente reflejan cambios importantes de propiedades de los objetos en la imagen, como por ejemplo: cambios en el material del objeto, discontinuidades en la orientación de una superficie, cambios en la profundidad, etc.

1.1. Motivación

La motivación de esta memoria es aplicar diferentes métodos de reconocimiento de bordes a imágenes de cortes de árboles, para detectar sus anillos. El hecho de saber de antemano qué es lo que se quiere reconocer, y su geometría aproximada, supone una ventaja a la hora de seleccionar, aplicar y refinar estos métodos, ya que se pueden aprovechar algunas de sus características para acotar los resultados que se obtendrán. Sin embargo, éstas no se pueden detectar fácilmente dado que las imágenes no son de buena calidad y los anillos pueden tener deformaciones.

Como parte de la revisión bibliográfica realizada para abordar el tema, se revisaron dos memorias realizadas previamente por alumnos de la profesora guía, que abordan el mismo problema, pero utilizando métodos completamente diferentes para resolverlo.

La memoria de Mauricio Cerda [10] se centra en el uso de filtros para procesar las imágenes, de manera que los anillos de árboles resaltaran lo suficiente como para ser detectados, y poderlos representar como una polilínea. Su trabajo, desarrollado en Matlab, consistió en:

- Estudiar algoritmos existentes, y evaluar su efectividad para detectar los anillos en árboles.
- Proponer e implementar modificaciones a algunos de estos algoritmos, para aprovechar las ventajas que le da las restricciones de la geometría del patrón buscado.
- Comparar la efectividad de los nuevos algoritmos con la de los originales.

Los resultados que obtuvo muestran que las modificaciones que hizo a los algoritmos generales de reconocimiento de bordes a través de filtros los hicieron entregar mejores resultados, ya que permitieron filtrar gran cantidad de los bordes incorrectamente detectados por los algoritmos tradicionales; aunque esta ganancia tuvo un costo, ya que también se omitieron algunos bordes verdaderos.

La memoria de Cristián Maluenda [11], en cambio, se centra en el uso de mallas geométricas para detectar los anillos en las imágenes. El objetivo de este método es crear una malla sobre la imagen de los anillos, refinándola sobre los mismos hasta que cada triángulo se encuentre completamente dentro o fuera de un anillo, o se llega a un tamaño mínimo del triángulo. Su trabajo, desarrollado en C++, consistió en:

- Estudiar algoritmos de refinamiento de mallas geométricas.
- Rediseñar un programa de refinamiento de mallas (gme2D).

- Incorporar los criterios de mejoramiento relevantes al problema al programa.

El resultado final fue obtener mallas que se encuentran mucho más refinadas dentro de los anillos que fuera de ellos; sin embargo, no se llegó a analizar la malla generada, ni a trazar la polilínea correspondiente a la aproximación de cada anillo.

1.2. Objetivos

1.2.1. Objetivo General

El principal objetivo de esta memoria es detectar anillos de árboles en imágenes, y entregar una aproximación de éstos utilizando polilíneas. Para esto, se busca crear un método mixto, que utilice tanto filtros para eliminar ruido y otras imperfecciones de las imágenes y resaltar los bordes en las mismas, como mallas geométricas para construir las polilíneas que representan los anillos.

1.2.2. Objetivos Específicos

Diseñar e implementar en Java una herramienta que cumpla con las siguientes características:

- Integre un algoritmo de refinamiento de triángulos y criterios para seleccionar los triángulos a refinar de acuerdo a las características de la imagen que se quieren evaluar.
- Utilice conjuntamente algoritmos basados en filtros y mallas. En particular, los filtros se utilizarán para eliminar ruido y resaltar los bordes, y las mallas se usarán para seleccionar los bordes reales, extrapolar los incompletos y eliminar lo que no sirve.
- Permita evaluar el comportamiento de los nuevos algoritmos con imágenes reales, y compararlos con los algoritmos originales.
- Permita testear nuevos algoritmos.
- Sea extensible a nuevos filtros de imágenes, algoritmos de refinamiento de mallas, criterios de selección, formatos de las imágenes y formatos de las mallas geométricas.

1.3. Contenido de la Memoria

En el capítulo 2: Trabajo Relacionado se muestra un resumen de algunos de los papers y otras fuentes de información revisadas que se consideraron relevantes para el tema estudiado.

En el capítulo 3: Metodología se analiza en detalle el trabajo realizado en esta memoria. En particular, se analiza la teoría e implementación de los algoritmos utilizados, tanto los de filtrado de imágenes como los de generación de mallas geométricas y polilíneas; junto con los criterios utilizados. Además, se discute el diseño y la implementación de la herramienta en Java, que integra los algoritmos y criterios anteriores en un diseño orientado a objetos modular y extensible.

En el capítulo 4: Resultados se muestran los resultado de la aplicación de los algoritmos antes mencionados, en forma individual y conjunta, tanto en imágenes fabricadas como reales, para evaluar y comparar su desempeño.

Finalmente, en el capítulo 5: Conclusiones se entregan las conclusiones obtenidas de la realización de esta memoria; se discute la utilidad de los algoritmos implementados, la efectividad del diseño utilizado y las maneras en que este trabajo puede ser extendido en el futuro.

Capítulo 2

Trabajo Relacionado

A continuación se dará una descripción de los trabajos relacionados con el tema de esta memoria que fueron revisados.

2.1. Reconocimiento de bordes en imágenes, un enfoque aplicado

La memoria de Mauricio Cerda [10] trata sobre métodos para detectar bordes en imágenes mediante la utilización de filtros y su aplicación al problema de reconocimiento de anillos de árboles. Los algoritmos discutidos son:

Filtro Gaussiano: se utiliza para eliminar ruido blanco y negro de la imagen.

Thresholding: se utiliza para separar todos los puntos de la imagen en dos grupos: los que tienen mayor o menor intensidad que un cierto valor (el *umbral*).

Normalización: se utiliza para maximizar su rango dinámico; es decir, para resaltar lo más posible las diferencias de intensidad de la imagen.

Equalización: se utiliza para definir lo más posible las características de las imágenes.

Todos los algoritmos descritos a continuación se utilizan para detectar bordes en imágenes.

Canny Edge Detector: se basa en la información del gradiente de la imagen.

LoG: se basa en la segunda derivada de la función de intensidad de la imagen.

Double-Threshold Linking: se basa en utilizar dos umbrales diferentes para combinar las imágenes resultantes.

Transformada de Hough: se basa en la parametrización de una función asociada al objeto que se está buscando (en este caso, círculos para aproximar anillos).

Contornos activos: se basa en la definición de un *funcional de energía*, que mide la calidad de la aproximación de su contorno.

2.2. Experimentación con mallas geométricas para el reconocimiento de anillos de árboles

En la memoria de Cristián Maluenda [11] se tratan algoritmos para detección de bordes en imágenes basados en mallas geométricas, y su aplicación en la detección de anillos de árboles. Los algoritmos vistos son:

Lepp-Delaunay: se utiliza para refinar selectivamente los triángulos de una malla geométrica que no satisfacen un cierto criterio (ángulo mínimo y ángulo máximo, entre otros), para mejorar la calidad de sus elementos. En este caso, el criterio está basado en las propiedades de los píxeles que están bajo el triángulo.

Bisección: en este caso, el lado más largo de un triángulo se divide en la mitad en caso de no cumplir con el criterio antes mencionado.

2.3. Segmentación de imágenes utilizando un algoritmo transversal a un árbol

Es fácil detectar bordes en imágenes cuando éstas tienen un alto contraste y bajo ruido; sin embargo, la dificultad aumenta cuando se tienen imágenes con mucho ruido y bordes borrosos. Esto es especialmente cierto cuando se deben identificar varios objetos, cada uno con diferentes niveles de brillo. Estas dificultades han llevado al desarrollo de varias técnicas, que usualmente son de dos tipos:

Detección directa de regiones: Elementos de la imagen (píxels) son unidos secuencialmente para formar regiones más grandes, utilizando restricciones globales.

Detección extendida de bordes: Se utiliza un sistema de diferenciación de regiones más complejo, unido a seguimiento de bordes.

El algoritmo descrito en este paper [1] es una generalización de la detección directa de regiones.

Métodos de detección de regiones

Los métodos de detección de regiones previamente existentes se pueden dividir en los siguientes tipos:

Unión o bottom-up: La imagen es dividida en un gran número de regiones pequeñas, que luego se van uniendo para formar regiones más grandes, siguiendo un cierto criterio.

División o top-down: La imagen es sucesivamente dividida en regiones cada vez más pequeñas hasta que se cumple un cierto criterio.

El método descrito a continuación es una mezcla de ambos.

Descripción del algoritmo

Sea X el dominio de una imagen. Sea $f(x, y)$ la función de brillo definida en X. Se define un predicado lógico P en subconjuntos S de X como sigue:

$$P(S) = \begin{cases} \text{verdadero} & \text{si existe una constante } a \text{ tal que } |f(x,y) - a| \leq e \\ & \text{para todos los puntos } (x, y) \in S \\ \text{falso} & \text{de otro modo} \end{cases}$$

donde e es una tolerancia de error predefinida (este predicado puede ser alterado si se desea dividir la imagen en función de otro parámetro, como por ejemplo el color).

Una segmentación de X es una partición de X en subconjuntos S_i, $i = 1, \ldots, m$, para algún m (no necesariamente único) tal que:

(a) $X = \bigcup_{i=1}^{m} S_i$

(b) $S_i \bigcap S_j = \emptyset$ para todo $i \neq j$

(c) $P(S_i) =$ verdadero para todo i

(d) $P(S_i \bigcup S_j) =$ falso para todo $i \neq j$ tal que S_i y S_j son adyacentes

Es decir, se divide el conjunto de puntos (que pueden ser píxeles) X en un conjunto de conjuntos propios y disjuntos de X, tal que todos los puntos de cada uno de esos conjuntos están lo suficientemente cerca (en este caso, en brillo) entre sí; y ninguno de esos subconjuntos podría unirse con otro adyacente, de manera que esa unión cumpliera la misma condición. Un algoritmo de unión empieza con una partición que satisface (c) y procede a cumplir (d); un algorimo de división empieza con una partición que satisface (d) y procede a cumplir (c). En cambio, un algoritmo mixto de unión y división empieza de una partición que no cumple ningúna de esas condiciones, y procede a cumplir ambas.

La formulación anterior se puede expresar como un árbol cuyos nodos corresponden a los subconjuntos de X tal que:

1. X es la raíz del árbol.
2. Los sucesores del nodo correspondiente a un subconjunto S de X corresponden a subconjuntos propios, disjuntos y collectivamente exhaustivos de S.
3. Las hojas del árbol representan los subconjuntos de X más pequeños en consideración (por ejemplo, píxeles).

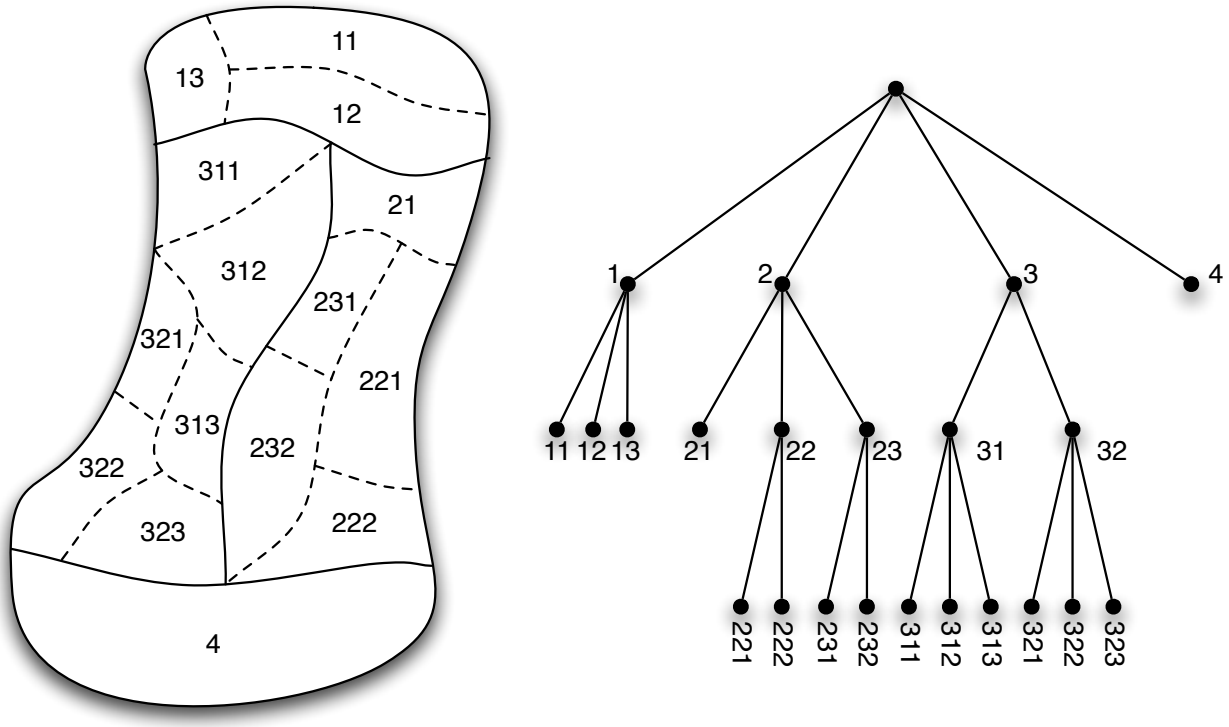


Figura 2.1: Ejemplo de una división por regiones, y su representación como árbol.

Se puede mostrar que una segmentación corresponde a un conjunto de nodos mínimo que separa la raíz de todas las hojas, de forma que para cada hoja L perteneciente al árbol se debe cumplir una de las siguientes condiciones:

- La raíz (R) está en el conjunto.
- La hoja (L) está en el conjunto.
- Existe uno y sólo un nodo N en el conjunto, tal que N es ancestro de L y sucesor de R.

Para que el conjunto de nodos correspondiente a la segmentación de la imagen cumpla con las condiciones antes descritas (definidas por el predicado $P(S)$), se debe cumplir que no existe otro subconjunto de nodos más cercano a la raíz tal que $P(S)$ es verdadero. Se puede observar que se puede visualizar los métodos *top-down* y *bottom-up* como los movimientos a través del árbol de segmentación y las alteraciones correspondientes al subconjunto de nodos. En el caso del algoritmo ahora presentado, se comienza con un cierto subconjunto de nodos que no cumple con el predicado $P(S)$, y se deben realizar movientos subiendo y bajando por el árbol hasta cumplirlo.

Otra operación, comocida como *agrupación*, se puede lograr transformando la estructura del árbol, ya sea combinando hojas, o particionando el subconjunto de nodos final en regiones de equivalencia. Como resultado, nodos adyacentes con diferentes ancestros y pertenecientes a diferentes niveles pueden ser combinados o agrupados para formar subconjuntos S más grandes pero irregulares (siempre y cuando $P(S)$ se mantenga verdadero).

2.4. Integrando crecimiento de regiones y detección de bordes

En este paper de Theo Pavlidis [2], se muestra cómo la combinación de dos técnicas para segmentación de imágenes produce resultados mucho mejores que cada una de las técnicas por separado. Estas técnicas con el Crecimiento de Regiones y la Detección de Bordes.
La aplicación de cualquier proceso de crecimiento de regiones da como resultado una buena aproximación inicial de los bordes, pero puede llevar a tres tipos de errores:

- El límite entre regiones no es un borde de la imagen, y no hay bordes cerca.
- Un límite corresponde a un borde pero no coincide con él.
- Existen bordes sin límites cerca.

El tercer tipo de error puede reducirce fácilmente hasta prácticamente eliminarse manejando adecuadamente los parámetros, pero con el costo de obtener una imagen demasiado segmentada (es decir, se aumenta el número de errores del primer tipo). Se eligió aceptar este tradeoff y reducir después los errores del primer tipo a través de la detección de bordes.
Para corregir los errores del primer tipo, para cada límite entre regiones se calcula una función de mérito, y los límites que arrojen un valor bajo de esa función son candidatos para ser eliminados. La función de mérito tiene la siguiente forma:

$$f_1(\text{contraste}) + \beta f_2(\text{artefactos de segmentación})$$

El primer término de esta función de mérito depende directamente del contraste que se observa a lo largo del límite, y el segundo término está relacionado con los errores que pueden introducirse por el hecho de estar utilizando un criterio de uniformidad demasiado estricto; por ejemplo, en el caso de una imagen con un cambio uniforme en la intensidad de la luz, el algoritmo podría insertar varios bordes entre las áreas más clara y oscura, siendo que en realidad no había un límite definido. Peor aún, es probable que esos bordes falsos reflejen las estrategias y estructuras de datos utilizados en la etapa de crecimiento de regiones.

Para corregir errores del segundo tipo, se calcula una expresión de la forma

$$f_3(\text{contraste}) + \alpha f_4(\text{suavidad})$$

a lo largo del límite y su vecindad, y se selecciona como borde la curva que maximiza la expresión. Esta etapa se llama el Paso de Modificación de Contornos.

2.5. Combinando División de Regiones y Detección de Bordes a través de Subdivisión de Imágenes Delaunay Guiada

En este paper [3], se propone un método adaptativo de división y unión para segmentar una imagen, en el cual se utiliza una triangulación a la cual se agregan nuevos puntos basándose en la semántica de la imagen.
El algoritmo propuesto utiliza una triangulación de Delaunay incremental para dividir la imagen en regiones; esto porque puede ser utilizada en una forma sistemática e incremental, añadiendo puntos uno a uno como vértices a la triangulación para refinar localmente donde la imagen lo requiera.
El método se divide en tres pasos:

Inicialización: acá se debe definir una triangulación de Delaunay inicial (que llamaremos D^0) que encierre todos los puntos de la imagen. En el caso particular en que la imagen es rectangular, se definen dos triángulos rectángulos iguales unidos por la hipotenusa, cuyos vértices son las esquinas de la imagen.

División: sea el triángulo d_i^j el i-ésimo triángulo de la j-ésima triangulación. Luego de la construcción de D^0, el algoritmo examina sucesivamente los triángulos d_i^j calculando el predicado de similitud $H()$, que evalúa el trozo de la imagen que se ubica bajo el triángulo d_i^j. Si el predicado es falso, los pixeles bajo las aristas del triángulo se clasifican topográficamente basándose en su vecindad por la función de diferencias $L()$. Después

de la detección de bordes, la función de división $S()$ asigna un *error de transición* a cada punto bajo las aristas. El punto que resulta tener el menor error de transición es insertado en la triangulación D^j para formar la próxima triangulación D^{j+1}. Este proceso se repite hasta que todos los triángulos cumplen con $H()$.

Unión: una vez que se tiene la triangulación final D^N, se procede a unir triángulos vecinos basándose en el predicado de similitud $H()$. Esto se logra buscando iterativamente; en cada paso se busca el triángulo más grande que queda de D^N, y se une con todos sus vecinos adyacentes tal que la unión cumple con $H()$. La iteración termina cuando no quedan triángulos por revisar. El resultado de esta operación es una división de la imagen en regiones definidas por polilíneas.

2.6. Segmentación de regiones por medio de división y unión deformable guiada por modelos

En este paper [4] se describe un método para segmentar una imagen basándose en la semejanza de las regiones con un modelo deformable provisto como parámetro.
El algoritmo consiste en varios pasos. Primero, se sobre-segmenta la imagen por medio de métodos tradicionales de unión de regiones; es decir, se segmenta hasta obtener regiones más pequeñas de lo que se podría considerar útil, para que todas las regiones resultantes queden encerradas en sólo una región de la segmentación final (aunque esto no se puede garantizar). Además, se computa un mapa de bordes, utilizando algoritmos de procesamiento de imágenes estándar.
Luego, se prueban varias combinaciones de agrupaciones candidatas de regiones para obtener una segmentación óptima de la imagen. El modelo que se busca detectar se deforma para hacerlo coincidir cada hipótesis de agrupación g_i de manera de minimizar una función de costo:

$$E(g_i) = \alpha E_{color} + (1-\alpha)((1-\beta)E_{área} + \beta E_{deformación})$$

donde α y β son constantes escalares con valores en el rango $[0, 1]$ que controlan la importancia relativa los tres términos:

E_{color} indica la compatibilidad del color de la agrupación de regiones.

$E_{área}$ indica el área que se superpone entre el modelo y la agrupación de regiones.

$E_{deformación}$ indica la energía de deformación del modelo.

En el paso anterior, también se calcula el costo de cada región (sin agrupar). Si el costo de la región evaluada es mayor que un cierto umbral, entonces se convierte en un candidato

para ser dividido. Esto se hace porque pese a que la imagen se sobresegmenta al comienzo del algoritmo, no se puede garantizar que alguna región no esté lo suficientemente dividida.

Para probar la calidad de una posible partición, se define una función de costo global:

$$\varepsilon = (i - \gamma) \sum_{i=1}^{n} r_i E(g_i) + \gamma n$$

donde γ es un factor constante, n es el número de agrupaciones en la partición de la imagen que se está evaluando, r_i es la proporción del área de la i-ésima agrupación de regiones con respecto al área total de las regiones conectadas, y $E(g_i)$ es la función de costo del grupo de regiones g_i.
Finalmente, se utiliza el algoritmo *la mayor confianza primero (HCF)* para encontrar un valor óptimo aproximado para ε.

Si el resultado final obtenido tiene un costo global mayor a un cierto umbral, se divide en dos todas las regiones candidatas a división definidas previamente, para luego repetir el proceso de agrupación de regiones. Esto se logra seleccionando un corte entre varios posibles en cada región, los que a su vez son seleccionados superponiendo la región con el modelo original y obteniendo los puntos medios de las partes convexas de la región que quedan dentro del modelo. La selección del corte se realiza comparando la función de costo de la región original, con el costo de las regiones resultantes. Si el corte de menor costo tiene un costo menor a la región original, ésta se divide. Al terminar este proceso con todas las regiones marcadas para división, se repite el proceso para encontrar una partición adecuada, mediante HFC.

2.7. Análisis de anillos de árbol

En este paper [5], se describe un método para detectar el borde y el área de anillos de árboles basándose en fotografías en escala de grises, y posteriormente modelar el tronco en 3D.

Modelos de contornos borrosos

La detección de contornos de este método fue pensada para funcionar bien en imágenes borrosas y con ruido. Como primer paso se detectan los contornos borrosos, los que posteriormente serán refinados para obtener bordes de un píxel de ancho.

Los contornos borrosos se modelan con la siguiente función:

$$C(x) = k(\lambda + sen(\alpha\beta|x|))e^{-\alpha|x|}$$

donde k y λ son constantes de normalización, β representa el parámetro de borrosidad y α corresponde al parámetro de escala relacionado con el ruido.

Filtro de detección de bordes

Sea $C(x)$ el modelo de contorno de una imagen. El borde de esa imagen puede evaluarse como el Laplaciano de $C(x)$:

$$f(x) = \frac{\partial^2 C(x)}{\partial x^2}$$

donde $f(x)$ corresponde a la función de detección de bordes dada por la ecuación:

$$f(x) = -signo(x)2k\alpha^2\beta e^{-\alpha|x|}\Big(1 - cos(\alpha\beta|x|) + (\frac{1-\beta^2}{2\beta})sen(\alpha\beta|x|)\Big)$$

Para evitar oscilaciones en el eje x, debe cumplirse que $0 < \beta < 1$. Si β es cercano a 0, $f(x)$ se utiliza para detectar contornos poco borrosos; en caso contrario, $f(x)$ se adapta bien a contornos borrosos.

Clasificación de pixeles y demarcado de contornos

Una vez que se ha realizado la detección de bordes con el filtro propuesto, se procede a la selección mediante un umbral. Si se escoge un umbral demasiado alto se elimina bien el ruido, pero también puntos que pertenecen a bordes reales de la imagen; por lo que se obtienen menos bordes de los deseados, y además parcialmente cortados. En cambio, si se escoge un umbral demasiado bajo, se obtienen todos o casi todos los bordes, pero además muchos bordes falsos y ruido. Para evitar estos problemas, se utiliza un umbral doble. En este caso, el proceso de umbral doble consiste en escoger dos valores de umbral: uno alto y otro bajo. Primero, se utiliza el umbral alto para detectar puntos que muy probablemente pertenecen a un borde real; luego, se utiliza el umbral bajo para detectar todos o casi todos los puntos que pertenecen a un borde real, aunque también aparece mucho ruido y bordes falsos. Después, se procede a escanear la imagen fila por fila, buscando puntos que están sobre el umbral alto; cada vez que se encuentra uno, se procede a seguir su contorno utilizando los puntos sobre el umbral bajo, y el gradiente de color para elegir la dirección. Finalmente, se eliminan todos los píxeles aislados y los bordes con un largo menor a un cierto valor dado.

Reconstrucción 3D

La reconstrucción 3D del tronco se basa en la triangulación tridimensional de Delaunay. Para esto, se realiza el proceso anterior para varias secciones del tronco a diferentes alturas. Luego, en cada imagen se trazan líneas radiales (partiendo del centro del anillo más pequeño), y cada punto de intersección entre un radio y un anillo se agrega a la triangulación. Al final de este proceso se obtiene una representación 3D del tronco.

2.8. Diseñando una familia de herramientas de creación de mallas

En este paper [8] se describe un diseño de una familia de herramientas de creación y manipulación de mallas geométricas, utilizando técnicas de orientación a objetos para permitir una fácil comprensión del procedimiento, reutilización de componentes y adición de nuevos algoritmos.

Pasos para la generación de una malla geométrica

Los pasos principales para cualquier proceso de generación de mallas geométricas son:

- Generación de una malla inicial que se ajuste a la geometría del dominio.
- Generación de una malla intermedia que satisfaga los requerimientos de densidad especificados por el usuario.
- Generación de una malla mejorada que satisfaga los requerimientos de calidad.
- Generación de la malla final.

Herramientas a implementar

Se desea que la familia de productos sea fácilmente extensible en los siguientes aspectos:

- Estrategias para la generación de mallas iniciales.
- Estrategias para el refinamiento de mallas.
- Criterios de refinamiento y mejoramiento.
- Estrategias para generar mallas finales.
- Forma de las regiones de refinamiento y mejoramiento.

Cada uno de estos aspectos es representado por una variación de una herramienta de la familia, y representa la base del diseño de la misma.

Implementación de las herramientas

La malla geométrica es representada en la clase *Mesh*, que a su vez está compuesta de triángulos, aristas y vértices. Esta clase también provee los métodos necesarios para acceder a sus componentes y manipularlos. La organización de los procesos anteriormente descritos se hizo definiendo cada paso de la generación de mallas como un tipo diferente, utilizando el patrón de diseño *strategy* [13], ya que permite que hayan varias maneras de implementar la misma operación, y éstas sean intercambiables; y permite que el software sea extensible a nuevos miembros de la familia, incluyendo así nuevos criterios y estrategias con muy pocos cambios al cógido original. Los cuatro algoritmos generales identificados son:

- Algoritmo de generación de malla inicial.
- Algoritmo de refinamiento.
- Algoritmo de mejoramiento.
- Algoritmo de malla final.

Para cada uno de ellos se creó una clase abstracta con un método para aplicarlos a una malla, que permite extenderlos con diferentes implementaciones. Además, el segundo y tercer algoritmos aplican un criterio sobre una región de la malla; para permitir extender tanto los criterios como las regiones, también fueron implementados como una clase abstracta.
La utilización del diseño orientado a objetos descrito en este paper permite la creación de una familia de herramientas de creación de mallas extensible y fácilmente configurable, lo que era el objetivo principal.

Capítulo 3

Metodología

En este capítulo se detallará el trabajo realizado en la memoria. Ya que el objetivo de la misma es crear una herramienta que permita utilizar una combinación de filtros y mallas geométricas para detectar los anillos en las imágenes, se dividirá este capítulo en tres secciones, que abordarán, respectivamente, los algoritmos basados en filtros, los algoritmos basados en mallas y el diseño y programación de la herramienta. La manera en que se aplican los algoritmos es la siguiente: primero, se toma una imagen adecuada y se la pasa por cero o más filtros para resaltar las características más importantes de la misma, de forma de facilitar el trabajo al generador de mallas. Luego se utiliza un algoritmo para generar una malla geométrica basada en la imagen, la cual seleccione sus puntos más relevantes; de esta manera, se tiene una representación de la imagen que se enfoca en sus características más importantes, y se descarta lo superfluo. Finalmente se realiza un proceso de generación de polilíneas, el cual toma como base la malla geométrica (en este punto, ya no se vuelve a utilizar la imagen original).

3.1. Algoritmos basados en filtros

En esta sección se detallan los algoritmos basados en filtros que fueron estudiados e implementados. Como el objetivo de aplicar los filtros a las imágenes de anillos es resaltar los mismos para facilitar la tarea de detectarlos en pasos posteriores, no se utilizaron filtros orientados a detectar directamente los bordes (como por ejemplo, el detector de bordes Canny).

3.1.1. Normalización

La normalización (también conocida como *estiramiento de contraste*) es un proceso orientado a mejorar el contraste de una imagen cambiando el tono de cada pixel de manera de ocupar con ellos todo el rango de tonos; es decir, se "estira" el rango dinámico de la imagen

para que el pixel de tono más oscuro que aparezca en la imagen corresponda al negro, y el más claro, al blanco. En la implementación se utilizó la siguiente fórmula para obtener el tono de cada pixel normalizado:

$$F_{x,y} = (I_{x,y} - I_{min})\frac{F_{max} - F_{min}}{I_{max} - I_{min}} + F_{min} \tag{3.1}$$

donde I_{max}, I_{min} e $I_{x,y}$ corresponden a los tonos máximo, mínimo y de cada pixel de la imagen original, y F_{max}, F_{min} y $F_{x,y}$ corresponden a los tonos máximo, mínimo y de cada pixel de la imagen normalizada.

El problema con esta fórmula es que la presencia de ruido en la imagen (especialmente pixeles blancos o negros dispersos en la misma; es decir, ruido del tipo *salt & pepper*) puede llevar a valores de I_{max} e I_{min} poco representativos del resto de la imagen, lo que a su vez lleva a un escalamiento poco representativo de la misma. Para superar esta dificultad, se utilizó el histograma de la imagen, fijando los valores I_{max} e I_{min} de manera de dejar fuera del rango cierto percentil de la misma. Por ejemplo, se puede considerar los pixeles que se encuentran entre el 5^{o} y 95^{o} percentil, descartando el resto; es decir, un 5 % de los pixeles de la imagen original tendrán un tono menor a I_{min}, y un 5 % lo tendrán mayor a I_{max}. Esta solución produjo imágenes con mucho mayor contraste, sin una pérdida apreciable de información.

(a) (b) (c)

Figura 3.1: Imagen original (a), con normalización utilizando todos los pixeles (b), y con normalización descartando el 5 % superior y el 5 % inferior de los pixeles (c).

3.1.2. Ecualización

La ecualización del histograma es un proceso orientado a aumentar el contraste local de una imagen; de esta manera, las intensidades quedan mejor distribuidas en el histograma. Al igual que la normalización, la ecualización consiste en transformar cada pixel $I_{x,y}$ de la imagen

inicial, en otro pixel $F_{x,y}$. Esta transformación consiste en una función de la acumulación de cada pixel; es decir, de la cantidad de veces que el tono de ese pixel, y todos los tonos menores a él, han aparecido en la imagen. En la implementación se utilizó la siguiente fórmula para obtener el tono de cada pixel ecualizado:

$$F_{x,y} = \frac{F_{max} - F_{min}}{alto \cdot ancho} \sum_{i=I_{min}}^{I_{x,y}} frecuencia(i) \tag{3.2}$$

donde $alto \cdot ancho$ corresponde al número total de pixeles de la imagen, I_{min} e $I_{x,y}$ corresponden a los tonos mínimo y de cada pixel de la imagen original; F_{max}, F_{min} y $F_{x,y}$ corresponden a los tonos máximo, mínimo y de cada pixel de la imagen ecualizada, y $frecuencia(i)$ corresponde al número de pixeles de la imagen original que tienen el tono i.

(a) (b)

Figura 3.2: Imagen original (a) y ecualizada (b).

3.1.3. Umbral

El filtro de umbral (o *thresholding*) consiste en clasificar los pixeles de la imagen de entrada en dos conjuntos: uno para los que tienen un tono mayor al valor de umbral (τ), y otro para los que tienen un tono menor. En la imagen resultante, se muestran los pixeles sólo con los tonos extremos: blanco y negro, que corresponden a los pixeles que quedan sobre y bajo el umbral, respectivamente.

Este filtro tiene dos problemas: es muy sensible al valor del umbral y a los cambios de luminosidad en la imagen. Si el valor del umbral no es elegido adecuadamente, se puede obtener como resultado una imagen compuesta casi solamente de pixeles blancos o negros; y si hay cambios de luminosidad muy pronunciados en la imagen, el resultado no será bueno

no importando el umbral escogido. Por eso, se implementaron dos filtros derivados de éste, con modificaciones para superar esos problemas: el filtro de *doble umbral* y el filtro de *umbral adaptativo*.

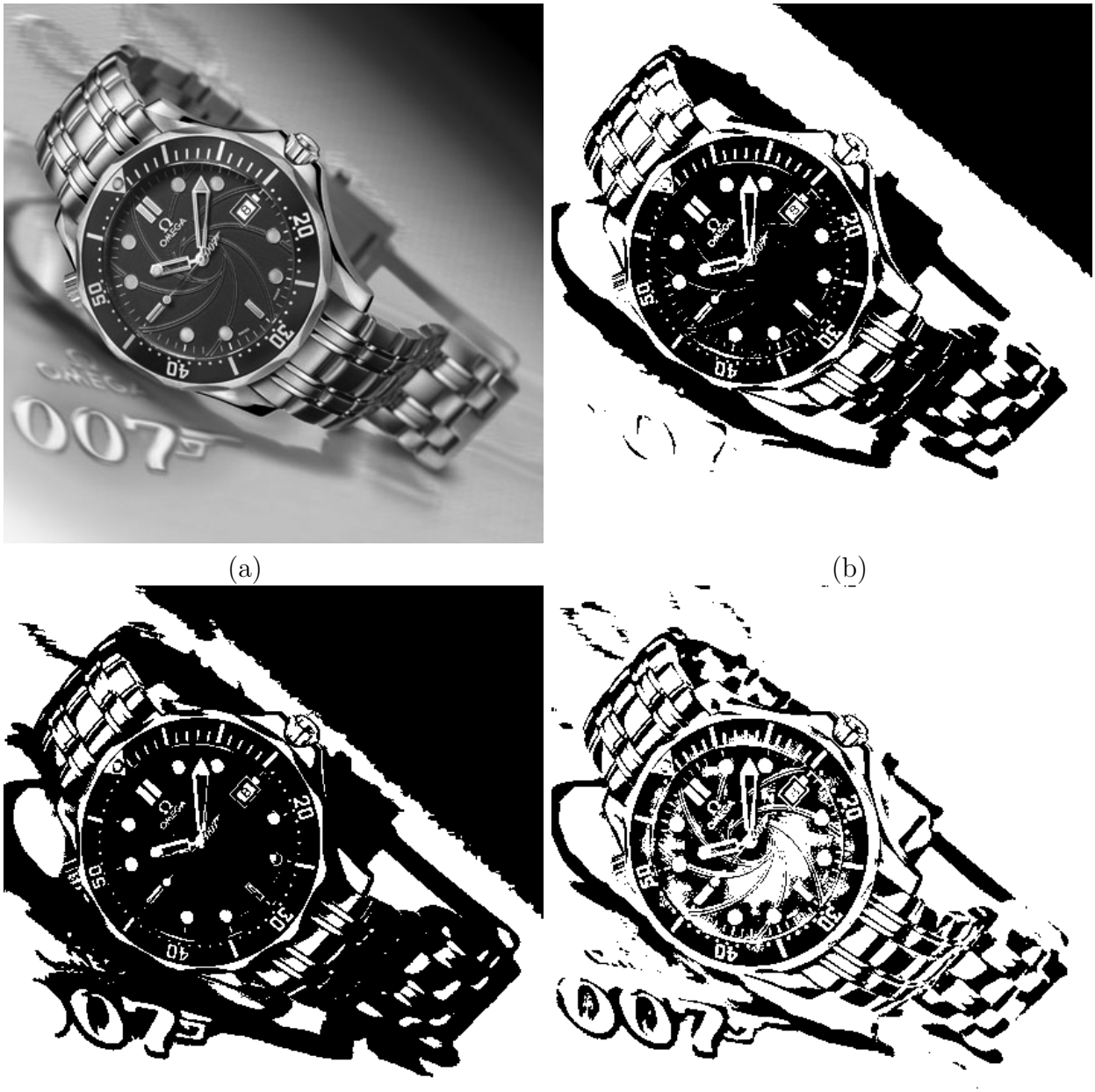
Doble umbral

El filtro de doble umbral es una modificación al filtro de umbral, mediante la cual se definen dos valores umbral (τ_1 y τ_2) en vez de uno solo. El mayor de los umbrales es un valor alto, el que al ser aplicado a la imagen de entrada resalta sólo pixeles que con mucha probabilidad pertenecen a bordes de la imagen, pero pasa por alto pixeles que también pertenecen a los bordes, aunque aparezcan menos definidos. En cambio, al aplicar el menor umbral se obtendrá como resultado una imagen que resalta casi todos los pixeles pertenecientes a bordes de la imagen, pero también muchos pixeles que no corresponden a los mismos (es decir, genera ruido). Luego, se combinan ambos resultados en una imagen final, que contiene todos los pixeles obtenidos del umbral alto, y los obtenidos del umbral bajo que estén directamente conectados con los pixeles obtenidos del umbral alto.

Umbral adaptativo

El filtro de umbral adaptativo es otra modificación al filtro de umbral. En este caso el algoritmo no recibe como parámetro el valor del umbral, sino que un radio (r). Esto es porque para cada pixel de la imagen resultante se utiliza un umbral diferente, el que se calcula como la media del tono de los pixeles que están en la vecindad de pixel original; esta vecindad queda determinada por el radio que se recibe como parámetro. Mientras más grande es el radio, más pixeles se utilizarán para calcular cada umbral; si es demasiado grande, se toman en cuenta pixeles que están muy lejos, por lo que el filtro no aporta ninguna mejora con respecto al original. Por otra parte, si el radio es demasiado pequeño, no se toman en cuenta suficientes pixeles para calcular la media (es decir, el umbral), por lo que se obtiene un resultado pobre.

El algoritmo antes descrito funciona bastante bien, ya que no tiene el problema de la sensibilidad a los cambios de luminosidad globales en la imagen, y además es no es tan dependiente del radio, pero tiene un problema: en las zonas donde hay muchos pixeles juntos de tonos parecidos (en particular, el fondo de la imagen), el resultado es azaroso. Para solucionar este problema, se cambia el umbral: en vez de usar la media de la vecindad, se usa la media menos una constante (en este caso, el valor del radio). Así se mantiene el buen resultado en las zonas más heterogéneas de la imagen, y se mejora en las zonas más homogéneas.



(a) (b)

(c) (d)

Figura 3.3: Imagen original (a), con filtro de umbral simple con $\tau = 120$ (b), con filtro de umbral doble con $\tau_1 = 80$ y $\tau_2 = 160$ (c), y con filtro de umbral adaptativo con $r = 13$ (d).

3.1.4. Filtro Gaussiano

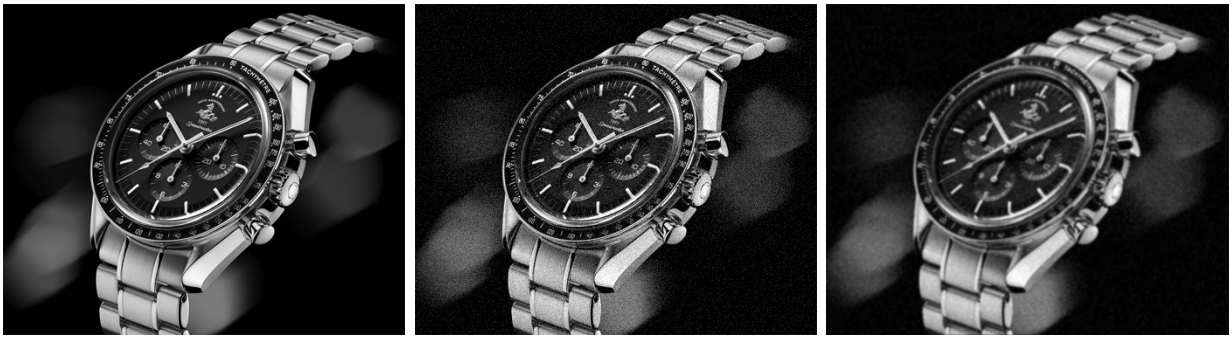
El filtro gaussiano es un efecto de emborronamiento uniforme que se aplica a la imagen, el cual disminuye su nivel de detalle y reduce su nivel de ruido; en particular, el ruido en forma de puntos blancos y negros esparcidos en la imagen (ruido *salt & pepper*). Como efecto negativo del filtro, está la pérdida de nitidez antes mencionada; por lo mismo, al aplicar este filtro hay que hacer un *tradeoff* entre la cantidad de ruido que se va a eliminar, y la nitidez que se va a perder. Sin embargo, un filtro gaussiano ligero es muy utilizado en el reconocimiento de bordes, ya que estos algoritmos tienden a ser bastante sensibles al ruido.

Este filtro corresponde a aplicar a la imagen una convolución con una distribución Gaussiana (o normal), en donde se recibe como único parámetro la desviación estándar de la distribución normal (σ), lo que determina la intensidad del filtro. Esto equivale a decir que cada pixel de la imagen resultante se calcula como un promedio ponderado de los pixeles de la imagen original, donde el peso de la ponderación de cada pixel corresponde a su distancia al punto que se está calculando.

Una consideración que se tuvo en la implementación es que este filtro cumple con la propiedad de ser *linealmente separable*; es decir, es un filtro bidimensional que se puede aplicar como dos filtros unidimensionales. En la implementación se aprovechó esta propiedad, ya que disminuye el tiempo de aplicación del filtro. Por lo tanto, la fórmula de la operación que se realiza en cada pixel es:

$$F_{x,y} = \frac{1}{2\pi\sigma^2} \sum_{k=1}^{m} \sum_{l=1}^{n} e^{-\frac{k^2+t^2}{\sigma^2}} I_{i-k,j-l} = \frac{1}{2\pi\sigma^2} \sum_{k=1}^{m} e^{-\frac{k^2}{\sigma^2}} \sum_{l=1}^{n} e^{-\frac{t^2}{\sigma^2}} I_{i-k,j-l} \tag{3.3}$$

donde $I_{x,y}$ es el tono del pixel de coordenadas (x, y) en la imagen original, σ es la desviación estándar de la distribución normal, m y n son el alto y ancho de la imagen en pixeles, y $F_{x,y}$ es el tono del pixel de coordenadas (x, y) en la imagen resultante.
Otra consideración que se tuvo en la implementación es que teóricamente la distribución en cada punto de la imagen es distinta a cero; por lo tanto, en el cálculo de cada pixel se deberían tener en cuenta todos los demás. Sin embargo, hacer esto es muy costoso en cálculos, y es innecesario ya que la importancia de cada pixel decae exponencialmente con la distancia. Por eso, en la implementación se consideró una vecindad con un radio de 3σ para el cálculo del tono de cada pixel.



(a) (b) (c)

Figura 3.4: Imagen original (a), con ruido (b), y con ruido y un filtro gaussiano con $\sigma = 1$ (c).

3.2. Algoritmos basados en mallas

En esta sección se detalla el algoritmo de generación de mallas que fue estudiado, modificado e implementado, así como los criterios de refinamiento que se utilizaron para insertar puntos en la misma.

3.2.1. Representación de la imagen como una malla geométrica

Parte del objetivo de esta memoria es modelar las imágenes de cortes de árboles que se reciben como entrada utilizando una malla geométrica que aproxime las características más importantes de la misma, y posteriormente procesarla para obtener los anillos. Para hacer esto, se eligió cubrir la imagen con una malla geométrica, y asignar un único tono a cada triángulo, que corresponde al promedio de los tonos de los pixeles que están dentro de ese triángulo. De esta manera, se reduce mucho la cantidad de información utilizada para representar la imagen; pero la calidad de la representación depende de la diferencia entre el tono de cada triángulo y el tono de los pixeles que contiene.

Otro punto que se debe considerar es que no se quiere utilizar una única malla inicial, un único algoritmo de refinamiento de mallas, un único criterio para evaluar la calidad de la misma, un único algoritmo de selección de puntos, ni un único algoritmo para generar las polilíneas que corresponden a los anillos, sino que se quiere diseñar una herramienta que permita crearlos e intercambiarlos posteriormente. A continuación se describirán los algoritmos que fueron programados y utilizados.

3.2.2. Malla inicial

La malla inicial es la malla geométrica que se crea inicialmente a partir de la imagen, para ser posteriormente refinada utilizando la información visual de la misma.

(a) (b)

Figura 3.5: Imagen original (a) y su representación como malla geométrica (b).

Malla inicial de Delaunay

La malla inicial que fue programada. Consiste en un rectángulo de la misma dimensión de la imagen, formado por dos triángulos rectángulos iguales unidos por su hipotenusa. Como su nombre lo indica, esta malla cumple con la condición de Delaunay, por lo que puede ser utilizada directamente para generar una malla de este tipo, insertando puntos.

3.2.3. Refinamiento de la malla

Este algoritmo crea la malla geométrica que aproxima la imagen de entrada, a partir de una malla inicial.

Malla de Delaunay

Se escogió implementar un algoritmo de refinamiento cuyo resultado es una triangulación de Delaunay, ya que por sus propiedades ésta garantiza tener triángulos lo más "equiláteros" posible; así se reduce de gran manera la aparición de *slivers*.

A continuación se describe el funcionamiento general del algoritmo:

- Recibe una imagen, su malla inicial correspondiente (que debe ser una triangulación de Delaunay), y un criterio de refinamiento.
- Para cada triángulo de la malla:

- Revisar si cumple con el criterio de refinamiento dado.
- Si cumple con el criterio, quiere decir que el triángulo es de calidad suficiente como para mantenerlo en la malla.
- Si no lo cumple, quiere decir que debe ser refinado. Para esto, se selecciona un punto que se encuentra completamente dentro del triángulo, el que es insertado mediante una inserción de Delaunay (la que garantiza que, luego de la inserción, la malla resultante sigue cumpliendo con la condición de Delaunay). Producto de esta inserción se generan tres nuevos triángulos, los que reemplazan el triángulo antiguo en la malla.

3.2.4. Selección del punto

Este algoritmo corresponde a la manera en que se elige qué punto de un triángulo de la malla será insertado para refinarlo.

Punto de mayor error

Para seleccionar el punto del triángulo que será insertado en la malla, se eligió tomar como puntos candidatos los puntos cuyas coordenadas corresponden a los pixeles que están completamente contenidos por el triángulo, y de entre esos puntos, se inserta el punto cuyo tono tiene la mayor diferencia con el tono del triángulo (que es el tono promedio de los pixeles que cubre). Si dos o más pixeles cumplen con esta condición, se elige entre ellos el más cercano al centro del triángulo (calculado como el promedio de las coordenadas de sus puntos).

Punto central

El punto a insertar es el punto central del triángulo. Las ventajas que se vieron en este método fueron que es muy rápido, y también genera una división bastante uniforme del triángulo.

3.2.5. Criterio de refinamiento

El criterio de refinamiento corresponde a la manera en que se decide si un triángulo de la malla es de suficiente calidad como para permanecer en la misma, o si debe ser refinado.

Diferencia de intensidad de pixeles

Este criterio se centra en el tono del triángulo y el de sus pixeles contenidos. Si la diferencia entre el tono del triángulo que se está examinando y el tono de todos sus pixeles contenidos

es menor a un cierto nivel dado (que se entrega como parámetro al criterio), entonces el triángulo se considera como de suficiente calidad como para permanecer en la triangulación (es decir, cumple el criterio). En caso contrario, debe ser refinado.

Área máxima

Este criterio se centra en el área del triángulo. Si la misma es menor a un cierto nivel dado (que se entrega como parámetro al criterio), entonces el triángulo cumple el criterio. Esto permite dejar de refinar triángulos muy pequeños.

3.2.6. Generación de polilíneas

El algoritmo de generación de polilíneas es el paso final del reconocimiento de anillos, y consiste en procesar una malla generada a partir de una imagen en el paso anterior, para obtener como resultado polilíneas que corresponden a los anillos de la imagen. Es en este paso donde se decidió ocupar la información geométrica de la imagen; es decir, propiedades que se sabe de antemano que los anillos poseen.

El algoritmo que fue programado consta de varias etapas, las que se fueron aplicando secuencialmente. Éstas serán descritas a continuación.

Filtro de aristas por diferencia de tono

Corresponde a la primera estrategia que se analizó e implementó para extraer anillos de la malla geométrica. Para implementarla, se utilizó una propiedad de los anillos: en la imagen se ven como diferencias de tono, ya que el árbol crece a diferentes ritmos en las distintas estaciones del año, y por eso su madera tiene diferencias de densidad y color. Por lo tanto, una arista tiene una alta probabilidad de pertenecer al borde de un anillo si el tono del triángulo a su izquierda es significativamente diferente al tono del triángulo a su derecha. Por eso, se programó este filtro como un filtro de umbral: se revisa cada arista de la triangulación, y se conservan sólo aquellas cuyos triángulos vecinos tienen una diferencia de tono superior a un umbral definido por el usuario. Este filtro elimina una buena cantidad de aristas que no pertenecen al borde de un anillo, pero tiene un problema: no diferencia entre aristas que pertenecen al borde externo de un anillo, al borde interno, o a imperfecciones en la madera (como nudos, cortes o manchas). Por lo tanto, se vio la necesidad de seguir filtrando este resultado, para extraer sólo las aristas que corresponden a bordes externos de anillos.

Filtro de aristas por orientación respecto al centro

En este paso, se recibe un conjunto de aristas de la malla con una buena probabilidad de pertenecer al borde de un anillo, ya sea por su lado interno o externo, o a imperfecciones; y se quiere eliminar aristas que probablemente pertenezcan al lado interno de los anillos. Para esto, se utiliza otra propiedad geométrica de los anillos: éstos son más oscuros y están más claramente definidos por su lado externo que por su lado interno. Esto ocurre porque la madera del lado interno del anillo corresponde a las estaciones más calurosas del año, por lo que el árbol crece a un ritmo mayor, y la madera es menos densa y más clara; en cambio, en las estaciones frías, el crecimiento del árbol se estanca, y la madera resulta más densa y oscura. Por lo tanto, este filtro conserva sólo las aristas cuyo triángulo vecino más oscuro está orientado hacia el centro del tronco. El centro del tronco se recibe como parámetro, por lo que podría ser calculado automáticamente, o bien seleccionado manualmente por el usuario.

Filtro de aristas por ángulo de inclinación

En este paso, se recibe un conjunto de aristas de la malla con una buena probabilidad de pertenecer al borde externo de un anillo, o a imperfecciones en la madera; y se quiere conservar sólo los primeros. Para esto, se utiliza otra restricción geométrica de los anillos: todos los anillos son versiones escaladas y concéntricas del anillo externo, y ningún anillo se intersecta con otro, sino que los anillos más pequeños están totalmente contenidos por los anillos más grandes. Esta restricción es sólo una aproximación, ya que hay ciertas excepciones: los anillos exteriores pueden no tener la misma forma que la de los interiores, debido, por ejemplo, a una rama o nudo; también puede ser que dos anillos se toquen en un punto (pero nunca intersectarse, por la manera en que se generan). Se decidió aceptar esta aproximación ya que representa correctamente una gran cantidad de imágenes.

Como se acepta que cada anillo se puede representar como una versión escalada y concéntrica de otro más grande (en particular, el anillo exterior), se puede deducir que si una arista de la triangulación pertenece al borde externo de un anillo, entonces el ángulo que forma con un radio que la intersecta debe ser parecido al ángulo formado por el mismo radio y el borde externo del anillo exterior. Este filtro recibe el anillo exterior del tronco como parámetro, por lo que podría ser calculado automáticamente, o bien seleccionado manualmente por el usuario. Luego se compara cada arista de la triangulación con el anillo exterior en la manera antes descrita utilizando un valor umbral definido por el usuario; y si es lo suficientemente similar, es conservado.

Transformada de Hough generalizada adaptada a aristas

En este paso, se recibe un conjunto de aristas de la malla con una buena probabilidad de pertenecer al borde externo de un anillo, y se las quiere utilizar para construir las polilíneas correspondientes a los mismos. Para esto, se utilizará una modificación de la Transformada de Hough Generalizada presentada en [12].

La transformada de Hough generalizada es un algoritmo que sirve para encontrar ocurrencias de un patrón dado de antemano en una imagen, variando los llamados *grados de libertad* del patrón. Por ejemplo, si lo que se busca son circunferencias, sus grados de libertad son: la posición del centro en los ejes x e y, y el radio de las mismas. Sin embargo, si se conoce el centro de las circunferencias, queda sólo un grado de libertad, lo que simplifica mucho los cálculos. El algoritmo programado consiste de los siguientes pasos:

1. Filtrado de aristas Corresponde a los pasos de filtrado de aristas descritos anteriormente, aplicados en forma secuencial. En conjunto, recibe la malla geométrica que representa la imagen del corte del tronco, y entrega como resultado final un conjunto de aristas que tienen una alta probabilidad de pertenecer al borde externo de un anillo; sin embargo, este conjunto no está totalmente libre de ruido.

2. Acumulación de posibles anillos En este paso, se utiliza como patrón a buscar el anillo externo, que se recibe como parámetro en forma de una polilínea junto con su centro en la etapa de filtrado de aristas (los cuales podrían ser obtenidos automática o manualmente). Como se acepta que cada anillo se puede expresar como una versión escalada del anillo más grande, y que todos los anillos comparten el mismo centro, se puede concluir que se tiene sólo un grado de libertad, el que será expresado como la razón entre el tamaño del anillo en cuestión y el tamaño del anillo mayor. Se llamará k a esta razón, que varía entre los valores 0 si el anillo corresponde a un solo punto (el centro), y 1 si el anillo es el anillo exterior. De esto se desprende que cualquier anillo se puede expresar en función del anillo exterior, el centro de los anillos y la razón k; por lo tanto, también se cumple que el punto central la iésima arista de cada anillo (que será llamado A_i) se puede escribir en función del punto central de la iésima arista del anillo exterior b_i, el centro C, y la razón k, de la siguiente manera:

$$A_i = C + k(C - R_i) \tag{3.4}$$

La acumulación consiste en iterar el conjunto de aristas obtenidas del filtrado, que corresponden a aristas con alta probabilidad de pertenecer al borde exterior de los anillos, que es lo que se intenta encontrar. Para cada arista, se expresa su punto central en la forma descrita en 3.4, de lo que se obtiene un valor de k; a este valor se le asigna un voto. Una vez finalizada

la iteración, se tiene un *acumulador de k*; es decir, un conjunto de posibles valores de k, cada uno de ellos con un cierto número de votos, los que corresponden a la cantidad de aristas que resultaron estar más cerca de ese valor de k que de cualquier otro.

3. Selección final de anillos Éste es el último paso del algoritmo. En él, se recibe como entrada el acumulador de k antes descrito, y se seleccionan los valores de k que finalmente corresponden a anillos de la imagen. Para esto, se ordenan los valores de k y se seleccionan los máximos locales entre los valores acumulados. Finalmente, para cada valor de k elegido, se calcula la polilínea correspondiente a escalar en esa razón la polilínea del anillo exterior, y se retorna como un anillo.

3.3. Diseño y programación de la herramienta

Parte del objetivo de esta memoria es diseñar e implementar una herramienta en Java que permita utilizar todos los algoritmos planteados anteriormente para el filtrado de imágenes, creación de mallas a partir de las mismas y extracción de anillos, y sea extensible a nuevos algoritmos. Para esto, se creó y trabajó con un diseño orientado a objetos basado en lo descrito en [8]; en él, se dio importancia a establecer una jerarquía adecuada y mantener una modularización que permita implementar nuevos algoritmos y criterios en el futuro; para esto, se utilizaron patrones de diseño apropiados.

3.3.1. Modelamiento de la imagen

Las imágenes con las que se trabaja consisten en una matriz de pixeles, cada uno de los cuales tiene asociado un tono que va de 0 a 255 si están en escala de grises (que es como se manipulan las imágenes en esta memoria), o tres componentes en el mismo rango si están en colores (con representación RGB). No fue necesario crear una nueva clase para modelar una imagen, ya que la biblioteca de Java provee la clase `BufferedImage`, que provee la representación y métodos adecuados para ello. Sin embargo, sí se creó una interfaz llamada `HandleImage` para proveer métodos ad-hoc relacionados con las imágenes, y también para poder extender los formatos de imágenes soportados por la aplicación (ver figura 3.6).

3.3.2. Modelamiento de la malla

Como fue explicado en secciones anteriores, la malla geométrica es una representación de una imagen, y se compone de puntos, aristas y triángulos. Para representarla, se crearon las siguientes clases, cada una de las cuales contiene los métodos necesarios para su creación y manipulación (ver figura 3.7):

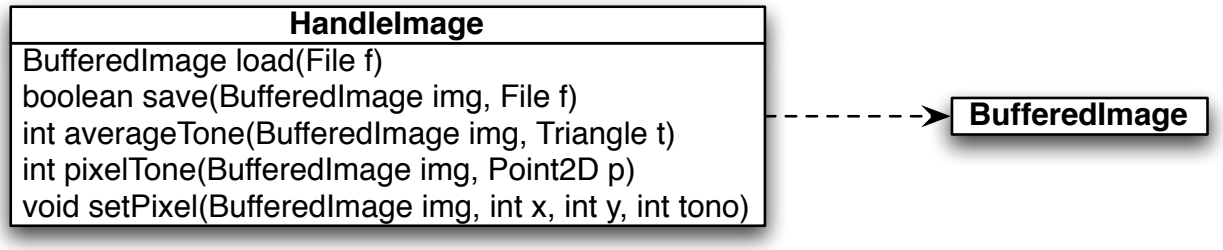


Figura 3.6: Diagrama de clases para la imagen.

`Point2D` es la representación de un punto en el plano.

`Points` es un conjunto de puntos.

`Edge` es la representación de una arista formada por dos puntos, con punteros a los dos triángulos de los cuales forma parte.

`Edges` es un conjunto de aristas.

`Triangle` es la representación de un triángulo formado por tres puntos (vértices) y un tono, que es el tono promedio de los pixeles de la imagen original que corresponden al triángulo en cuestión, y punteros a las tres aristas que forman sus lados.

`Triangles` es un conjunto de triángulos.

`Mesh` contiene toda la información concerniente a la malla; es decir, la triangulación que representa una imagen. Se compone de una instancia de `Points`, una de `Edges` y una de `Triangles`.

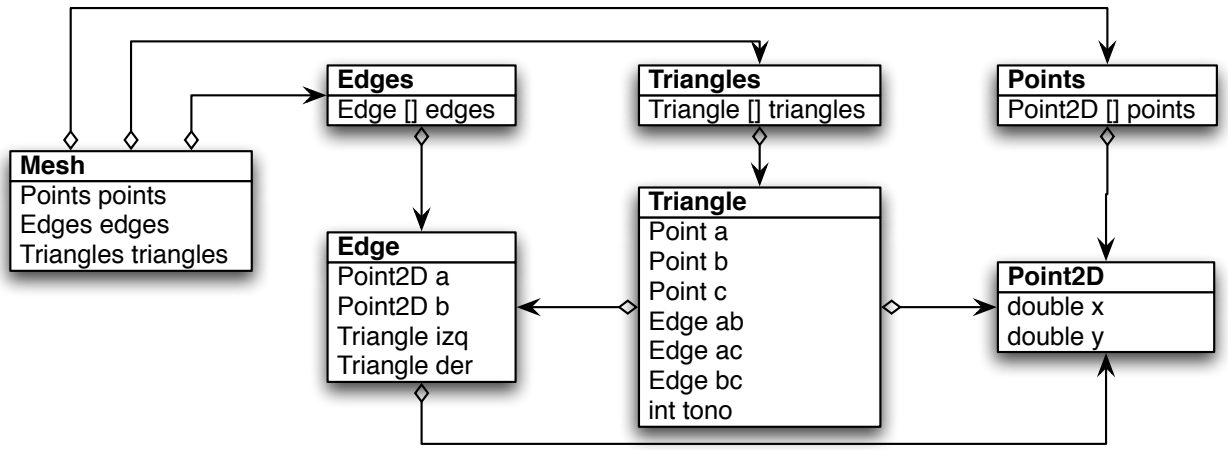


Figura 3.7: Diagrama de clases para la malla geométrica.

3.3.3. Modelamiento de los anillos

Uno de los objetivos de la memoria es representar como polilíneas los anillos de los árboles que aparecen en las imágenes. Por lo tanto, se crearon las siguientes clases, cada una de las cuales contiene los métodos necesarios para su creación y manipulación (ver figura 3.8):

`Polyline` es un conjunto de aristas.

`Polylines` es un conjunto de polilíneas.

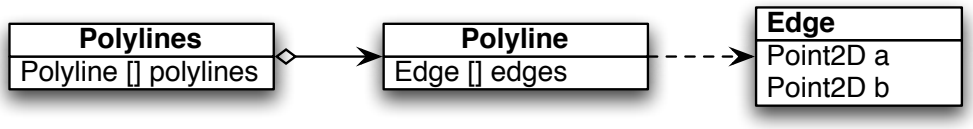


Figura 3.8: Diagrama de clases para las polilíneas que representan anillos.

3.3.4. Modelamiento de la imagen como un todo

Como ha sido explicado en secciones anteriores, cada imagen de un tronco de árbol se asocia a una malla y a un conjunto de polilíneas que corresponden a sus anillos. Es por esto que se creó una clase llamada `VisibleMesh`, la que a través de agregación contiene una instancia de la clase `BufferedImage`, una de `Mesh` y una de `Polylines`, además de métodos apropiados para aplicar los diversos algoritmos necesarios para extraer los anillos de una imagen (ver figura 3.9).

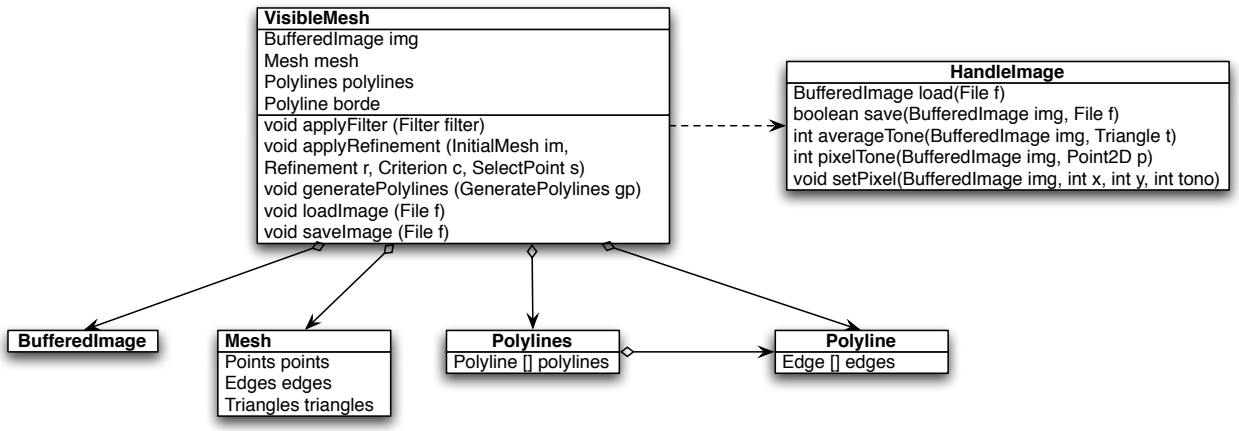


Figura 3.9: Diagrama de clases para la imagen junto con su malla y anillos.

3.3.5. Generación de la malla y las polilíneas

Un importante problema de diseño es decidir cómo generar la malla geométrica y las polilíneas. Ya que se requiere que los algoritmos de creación de malla inicial, refinamiento y generación de polilíneas, así como los criterios de refinamiento y de selección de puntos sean intercambiables y extensibles sin modificar el código existente, se decidió utilizar el patrón de diseño *strategy* tal como se sugiere en [8]. Se identificaron seis interfaces a implementar, las cuales son fácilmente heredadas para cumplir con los requisitos antes descritos. Éstas son:

`Filter`

Corresponde a los algoritmos de filtrado de imágenes. Cada uno de sus hijos debe implementar el método `apply`, que recibe una imagen y una región donde aplicar el filtro, y lo aplica a la imagen (ver figura 3.10).

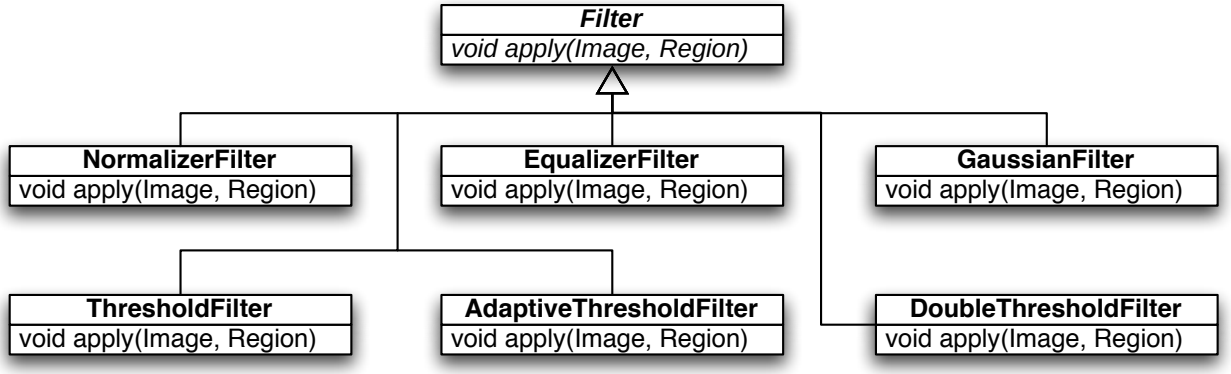


Figura 3.10: Diagrama de clases para los filtros de imágenes.

`InitialMesh`

Corresponde a los algoritmos de creación de una malla inicial a partir de una imagen. Sus hijos deben implementar el método `apply`, que recibe una malla (vacía) y una imagen, y agrega a la malla sus primeros triángulos (ver figura 3.11).

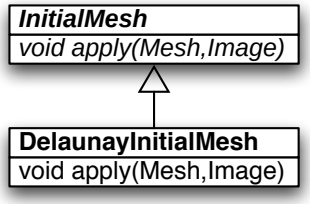


Figura 3.11: Diagrama de clases para los algoritmos de malla inicial.

`Refinement`

Corresponde a los algoritmos de refinamiento de la malla geométrica. Sus hijos deben implementar el método `apply`, que recibe una malla (inicializada), una imagen, un criterio, un algoritmo de selección de puntos y una región, y refina los triángulos de la malla que están dentro de la región especificada utilizando el algoritmo de selección de puntos dado hasta que todos los triángulos cumplen el criterio dado (ver figura 3.12).

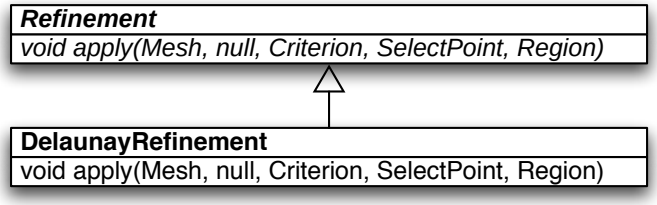


Figura 3.12: Diagrama de clases para los algoritmos de refinamiento de mallas.

`SelectPoint`

Corresponde a los algoritmos para seleccionar un punto de un triángulo. Sus hijos deben implementar el método `selectPoint`, que recibe una malla (inicializada), una imagen y un triángulo de la malla, y retorna un punto completamente contenido por el triángulo (ver figura 3.13).

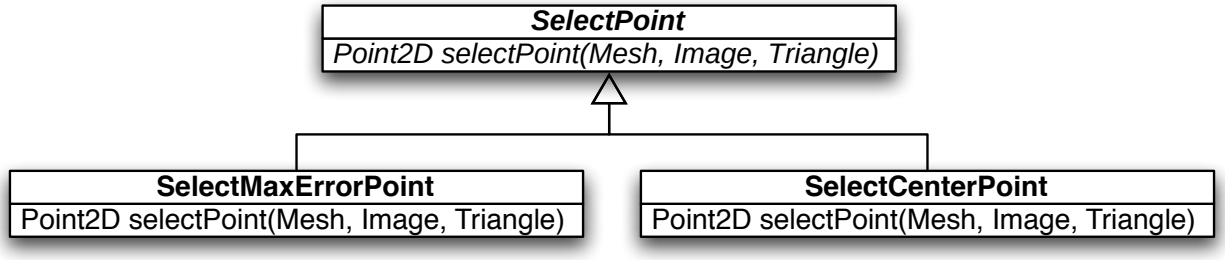


Figura 3.13: Diagrama de clases para los algoritmos de selección de puntos.

`Criterion`

Corresponde a los criterios para decidir si un triángulo debe o no ser refinado. Sus hijos deben implementar el método `meetsCriterion`, que recibe una malla (inicializada), una imagen y un triángulo de la malla, y retorna un boolean que indica si el triángulo cumple o no con el criterio (ver figura 3.14).

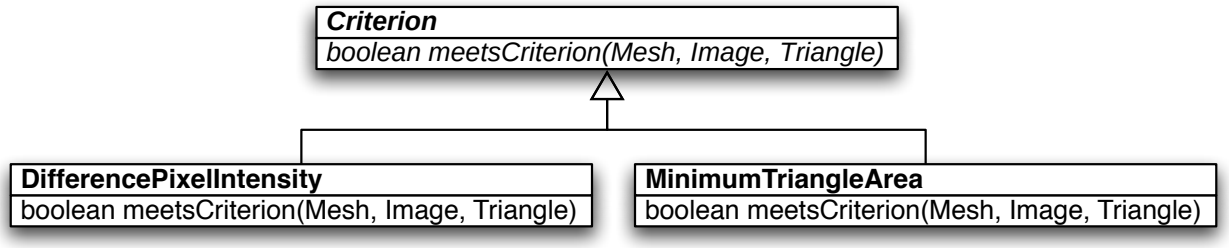


Figura 3.14: Diagrama de clases para los criterios de refinamiento de triángulos.

`Region`

Corresponde a la región de la imagen donde se debe aplicar un cierto algoritmo. Sus hijos deben implementar el método `intersects`, el cual recibe un triángulo y retorna un boolean que indica si ese triángulo intersecta o no la región dada (ver figura 3.15).

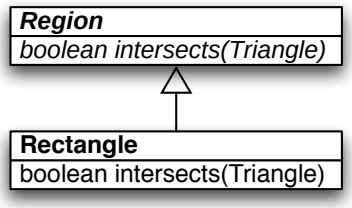


Figura 3.15: Diagrama de clases para las regiones.

3.3.6. Interfaz Gráfica

En la interfaz del programa se crearon dos áreas: una para cargar y guardar imágenes y definir y aplicar los algoritmos a utilizar, y otra para visualizar la imagen, la malla y las polilíneas obtenidas, y definir manualmente el centro y el anillo exterior de la imagen (ver figuras 3.16 y 3.17).

3.3.7. Implementación

Todo el programa fue implementado en el lenguaje Java 1.5, utilizando el entorno de desarrollo NetBeans 5.5.1. Este lenguaje fue escogido ya que se presta muy bien para el desarrollo de complejos diseños orientados a objetos, y también como una forma de probar su idoneidad para el desarrollo de un programa que involucra cálculos intensivos.

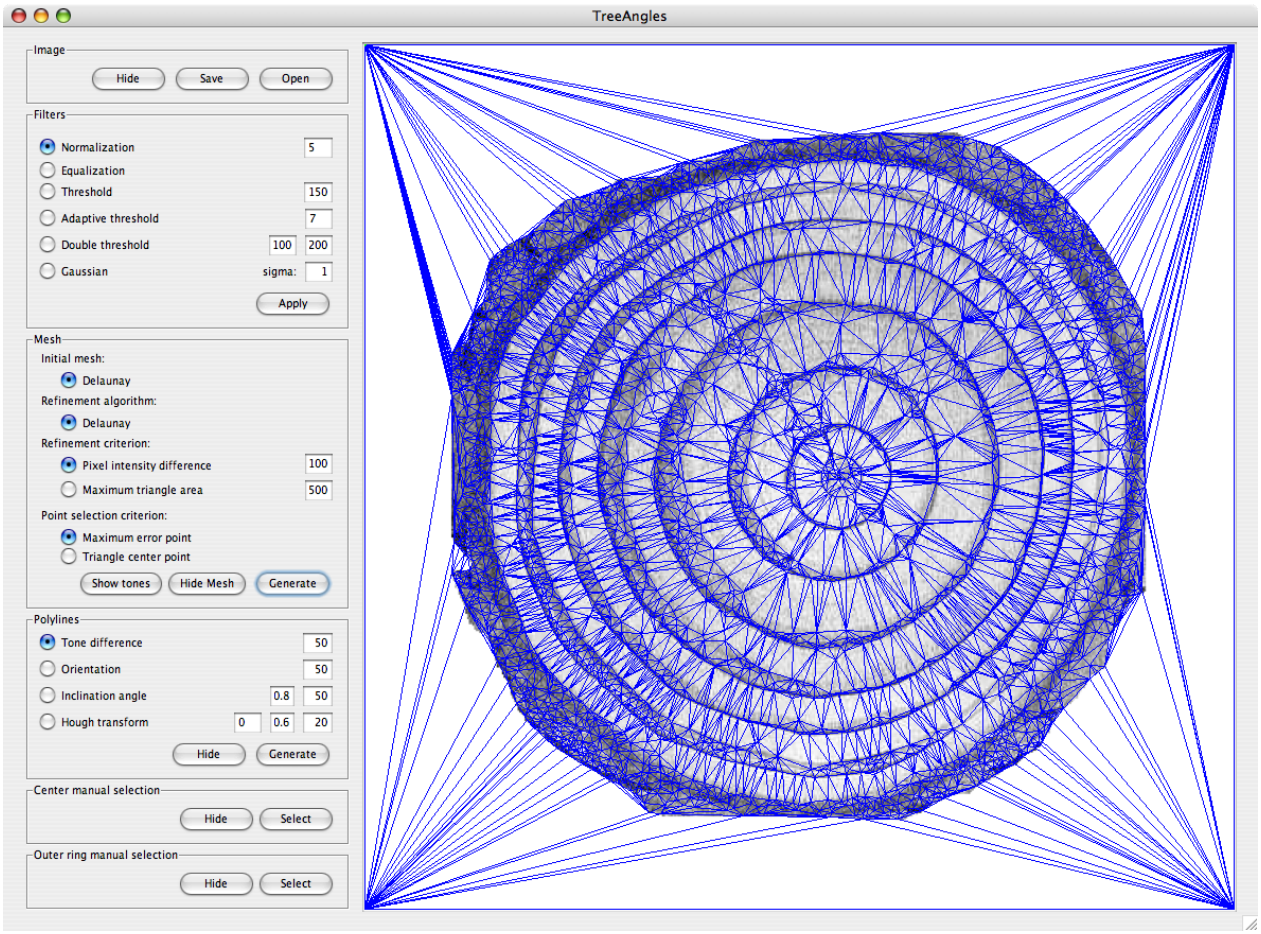


Figura 3.16: Interfaz gráfica del programa.

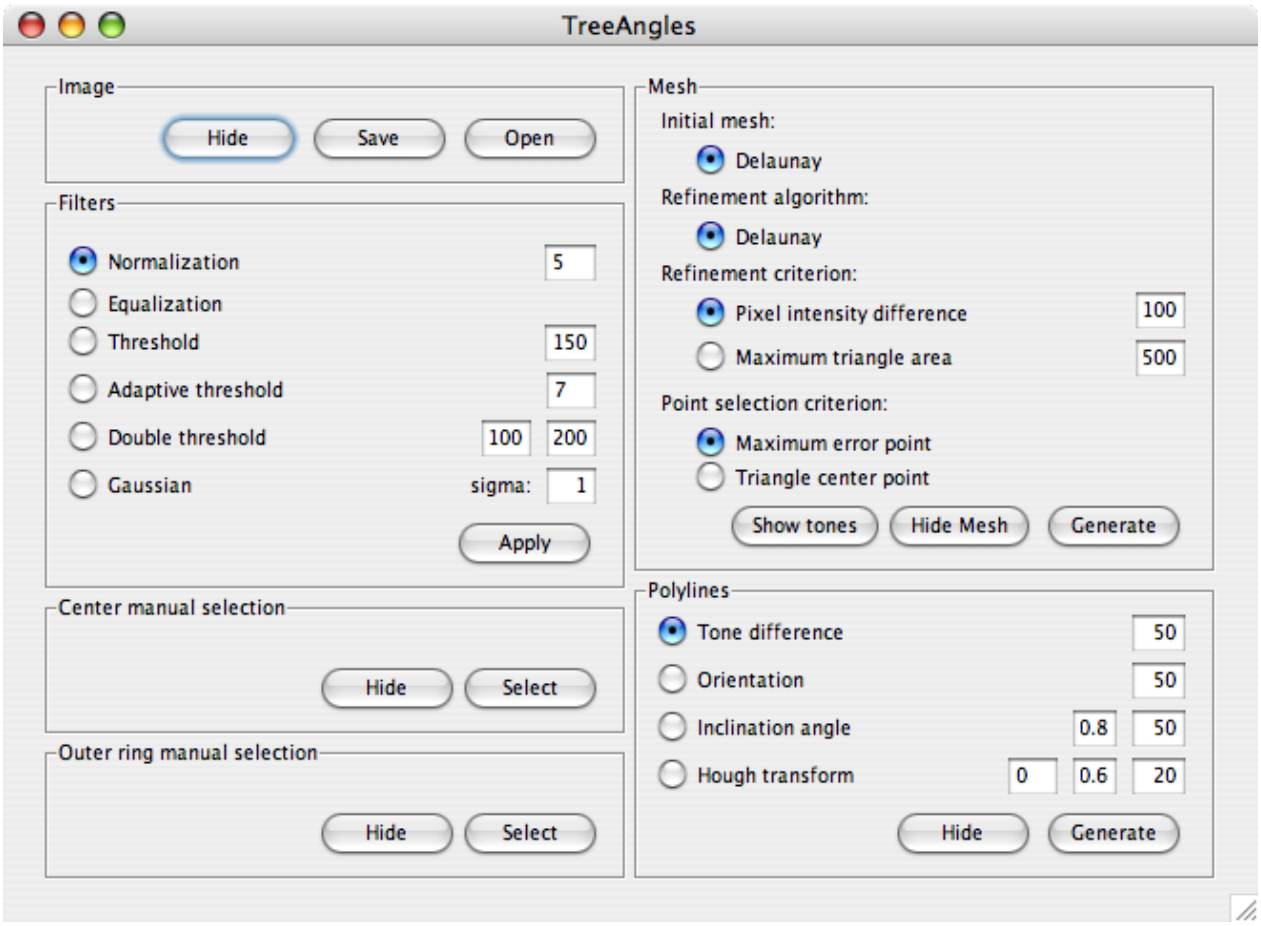


Figura 3.17: Detalle del área de comandos de la interfaz gráfica del programa.

Capítulo 4

Resultados

En el capítulo anterior se revisaron los algoritmos utilizados para detectar anillos en imágenes. En este capítulo se presentan y analizan los resultados obtenidos por los mismos, tanto para imágenes fabricadas como para imágenes reales.

4.1. Aplicación de los algoritmos de generación de mallas geométricas y polilíneas a imágenes fabricadas

En esta sección se mostrarán los resultados de aplicar pruebas a una imagen fabricada, de tamaño 650 pixeles x 650 pixeles, que representa la imagen "ideal" del corte del tronco de un árbol (ver figura 4.1). En esta imagen los anillos son perfectamente circulares, hay una transición muy marcada entre anillos, y los mismos son concéntricos y están totalmente separados entre sí. La finalidad de utilizar esta imagen ideal es poder comparar la efectividad de los distintos algoritmos de forma controlada.

Dado que la imagen de prueba que se está utilizando no tiene ruido ni imperfecciones, resulta innecesario aplicar filtros para mejorarla; por lo tanto, en este paso se probará a efectividad de la generación de mallas geométricas, utilizando los distintos criterios de refinamiento y de selección de puntos implementados; y de la construcción de las polilíneas correspondientes a los anillos utilizando el método incremental implementado.

Como se vio en la sección anterior, se implementaron dos criterios de refinamiento de triángulos, que son la diferencia de intensidad de pixeles y el área máxima de triángulo, y dos criterios de inserción de puntos, que son el punto de mayor error y el punto central del triángulo. En esta prueba, se refinó la imagen antes mencionada para cada combinación posible de un criterio de refinamiento y uno de inserción de puntos, lo que da cuatro casos posibles, que se detallan a continuación.

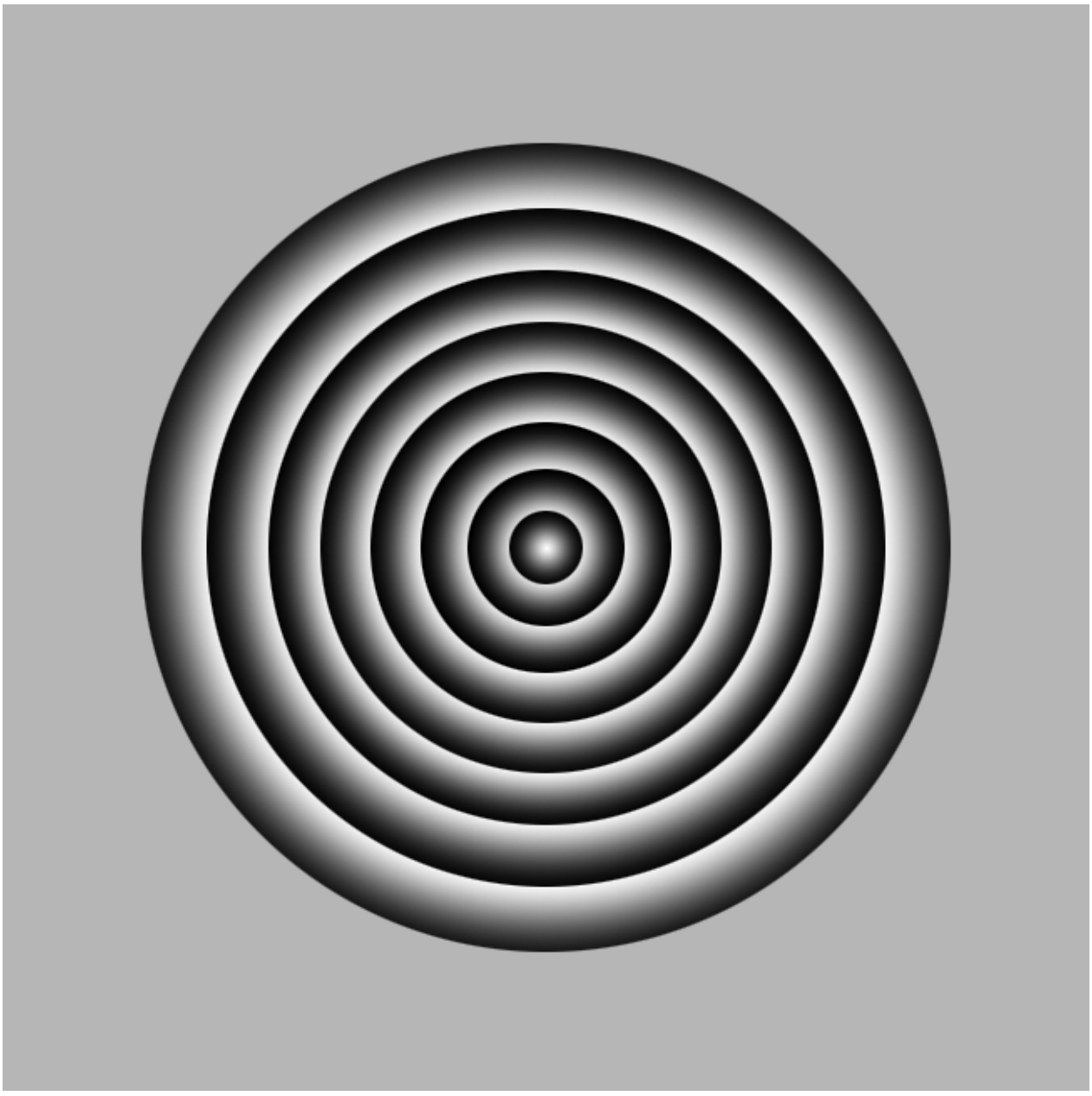
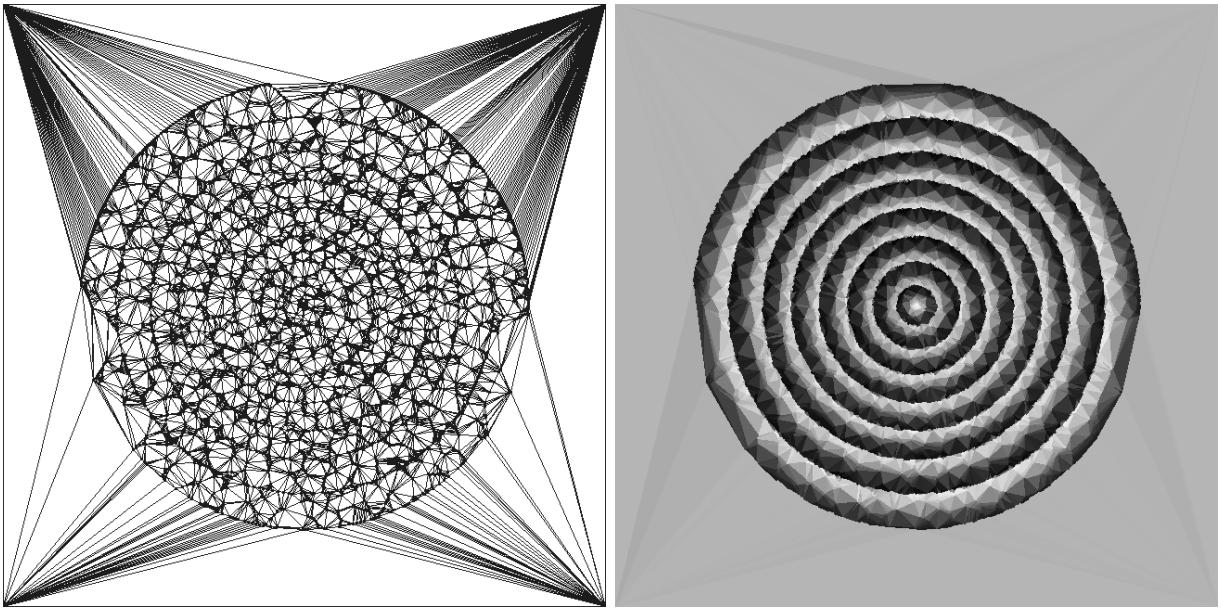


Figura 4.1: Imagen fabricada que representa anillos de árboles sin ningún tipo de imperfecciones.

4.1.1. Diferencia de intensidad de pixeles y punto de mayor error (Malla PID-MEP)

Generación de la malla geométrica

En este caso, se aplicó el criterio de diferencia de intensidad de pixeles con una diferencia de intensidad mínima de 100 para refinar los triángulos de la malla, y para refinar los triángulos que no cumplen con este criterio, se escogió el punto a insertar utilizando el criterio del punto de mayor error. Como resultado, se obtuvo una malla de buena calidad y bastante ajustada a la imagen, muy refinada alrededor de los anillos y más gruesa en el fondo, con bordes muy bien definidos. Esto implica que se tiene un alto nivel de detalle para describir los anillos y mucho menor en los demás lugares, lo que es un resultado óptimo ya que se describe adecuadamente la información importante de la imagen y se descarta el resto. Esta malla está compuesta de un total de 10568 triángulos (ver figura 4.2).



(a) (b)

Figura 4.2: Triángulos de la malla PID-MEP (a) y tonos de la misma (b).

Generación de las polilíneas

Una vez obtenida la malla geométrica, hay que extraer anillos de la misma. El método implementado en esta memoria consiste en aplicar sucesivamente tres filtros de aristas a la malla, cuyo objetivo es conservar las aristas que tienen una alta probabilidad de pertenecer a los bordes de los anillos y descartar el resto, y luego aplicar a esas aristas una versión modificada de la Transformada de Hough Generalizada. Estos pasos se detallan a continuación.

Cabe destacar que tanto el punto central como el anillo exterior, que son necesarios como parámetros de algunos de estos pasos, se ingresaron manualmente.

Filtro por diferencia de tonos En este paso se decidió utilizar una diferencia muy ligera entre tonos (el parámetro se fijó en 5), para eliminar sólo aristas que separan triángulos de tono muy similar, y mantener todas las aristas que tengan probabilidad de pertenecer a un borde. Como resultado, se filtraron casi todas las aristas que corresponden al fondo de la imagen, y menos de la mitad de las restantes; de éstas, casi todas pertenecientes a un solo anillo, y no al borde entre dos de ellos (ver figura 4.3 a).

Filtro por orientación respecto al centro En este paso no es necesario escoger un parámetro, ya que se mantienen sólo las aristas cuyo triángulo vecino de tono más oscuro está más cerca del centro de la imagen que el otro. Como resultado, se filtró una gran cantidad de aristas que no están sobre el borde de un anillo, pudiéndose distinguir claramente los mismos en las aristas resultantes (ver figura 4.3 b).

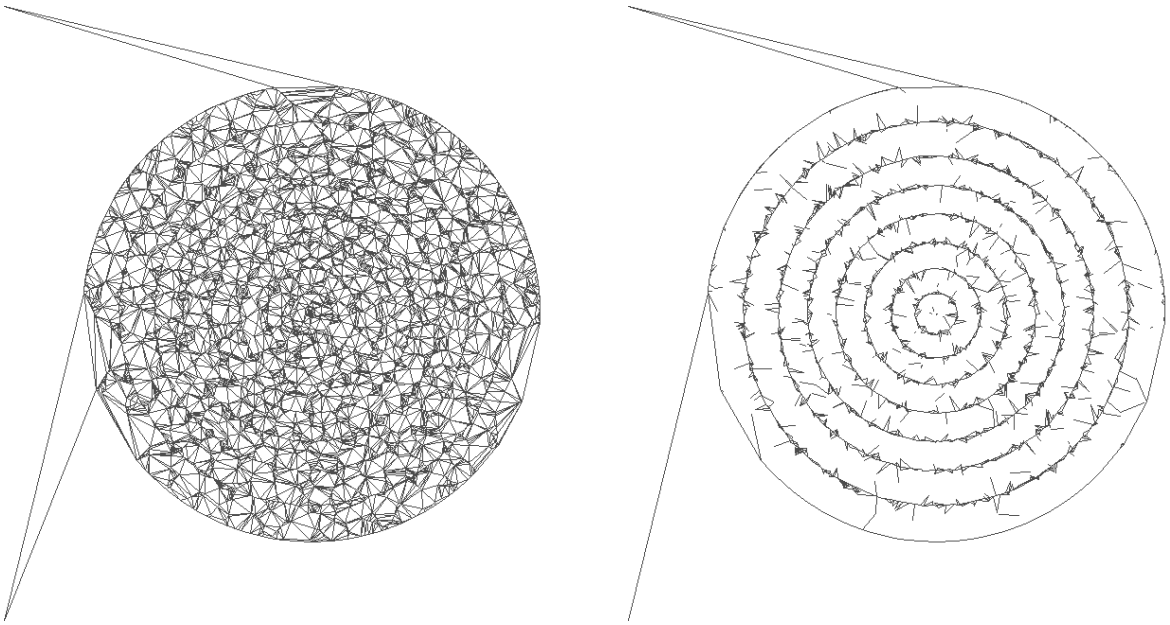
Filtro por ángulo de inclinación En este paso correspondiente al último filtro se desea mantener sólo aquellas cuyo ángulo de inclinación es similar al del anillo exterior. Para aplicarlo, es necesario ingresar otro parámetro que mide esta similaridad; en este caso se escogió el valor 0.4, donde 1 representa líneas paralelas, y -1, perpendiculares. Como resultado se obtuvo una imagen casi perfecta de los anillos, pero con la particularidad de que se borraron todas las aristas que pasaban por los ejes cartesianos. Este problema se dio por un error de truncación, ya que alrededor de esos puntos las aristas tienen pendientes muy cercanas a 0 y $\pm\infty$; queda como trabajo pendiente el corregirlo (ver figura 4.3 c).

Transformada de Hough Generalizada adaptada a aristas Finalmente, en este paso se utiliza adaptación a aristas de la transformada de Hough generalizada para generar versiones escaladas del anillo exterior. El resultado obtenido es una aproximación casi perfecta de los bordes de los anillos de la imagen (ver figura 4.3 d).

4.1.2. Diferencia de intensidad de pixeles y punto central del triángulo (Malla PID-TCP)

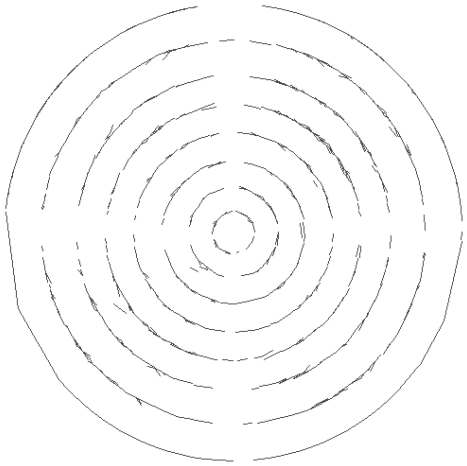
Generación de la malla geométrica

En este caso, al igual que en el anterior, se utilizó el criterio de diferencia de intensidad de pixeles con una diferencia de intensidad mínima de 100 para refinar los triángulos de la malla; pero para refinarlos se aplicó el criterio de inserción del punto central. El resultado obtenido es una malla con características bastante similares a la anterior; de buena calidad

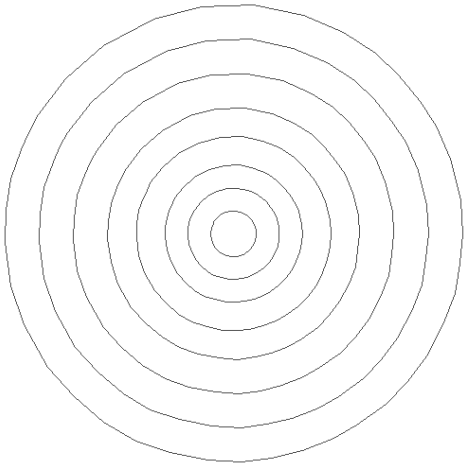


(a)

(b)



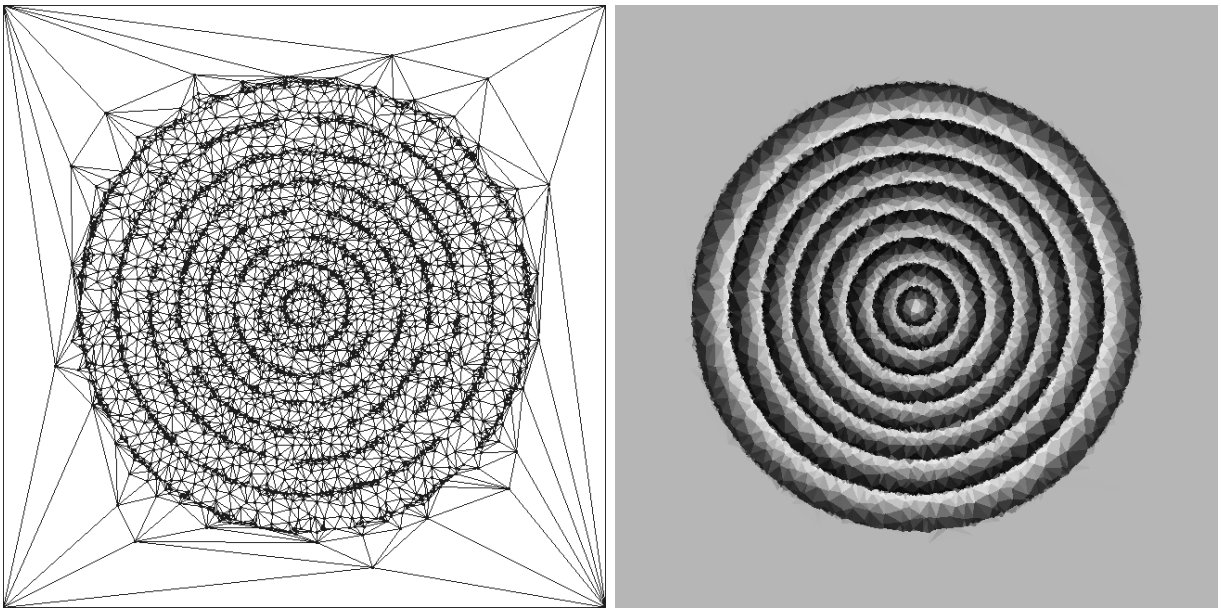
(c)



(d)

Figura 4.3: Filtro por diferencia de tonos (a), filtro por orientación respecto al centro (b), filtro por ángulo de inclinación (c), y transformada de Hough generalizada (d) aplicados a la malla PID-MEP.

y más refinada cerca de los anillos que en el fondo (pero en menor medida que la malla PID-MEP), con bordes y gradientes de tono bien definidos. Una ventaja con respecto a la malla anterior es que se utilizaron menos triángulos, 8118 en total (ver figura 4.4).



(a) (b)

Figura 4.4: Triángulos de la malla PID-TCP (a) y tonos de la misma (b).

Generación de las polilíneas

Para generar polilíneas en todas las mallas siguientes, se utilizarán los mismos parámetros que en el caso de la malla anterior, para poder hacer una comparación objetiva de las mismas.

Filtro por diferencia de tonos Al igual que en el caso anterior, y por las mismas razones, se utilizó una diferencia de tonos de 5 para eliminar aristas cuyos triángulos vecinos son de tonos muy similares. El resultado también es similar al del caso anterior, e incluso mejor: acá desaparecieron todas las aristas del fondo, además de algunas de las que pertenecen al interior de los anillos (ver figura 4.5 a).

Filtro por orientación respecto al centro Como resultado de aplicar este filtro, y al igual que en el caso anterior, desapareció una gran cantidad de aristas que no están sobre el borde de un anillo, pudiéndose distinguir claramente los mismos en las aristas resultantes (ver figura 4.5 b).

Filtro por ángulo de inclinación Como resultado de la aplicación de este fitro, se obtuvo un imagen muy buena de los anillos, casi sin ruido fuera de ellos, pero con lineas algo

más gruesas que en el caso anterior, ya que en algunos puntos se mantuvieron varias aristas muy juntas y paralelas que representan el mismo borde (ver figura 4.5 c).

Transformada de Hough Generalizada adaptada a aristas El resultado obtenido del paso final de la generación de polilíneas es una aproximación casi perfecta de todos los anillos de la imagen, con la excepción de uno (el segundo, contando desde dentro). Como en el paso anterior se observa claramente que ese anillo está representado por una buena cantidad de aristas, se puede concluir que hubo un problema en el algoritmo de selección de anillos finales; es decir, el contador correspondiente a ese anillo no fue seleccionado pese a que tiene una buena cantidad de aristas ya que no resulta ser un máximo local. Este problema queda propuesto para ser resuelto posteriormente (ver figura 4.5 d).

4.1.3. Área máxima de triángulo y punto de mayor error (Malla MTA-MEP)

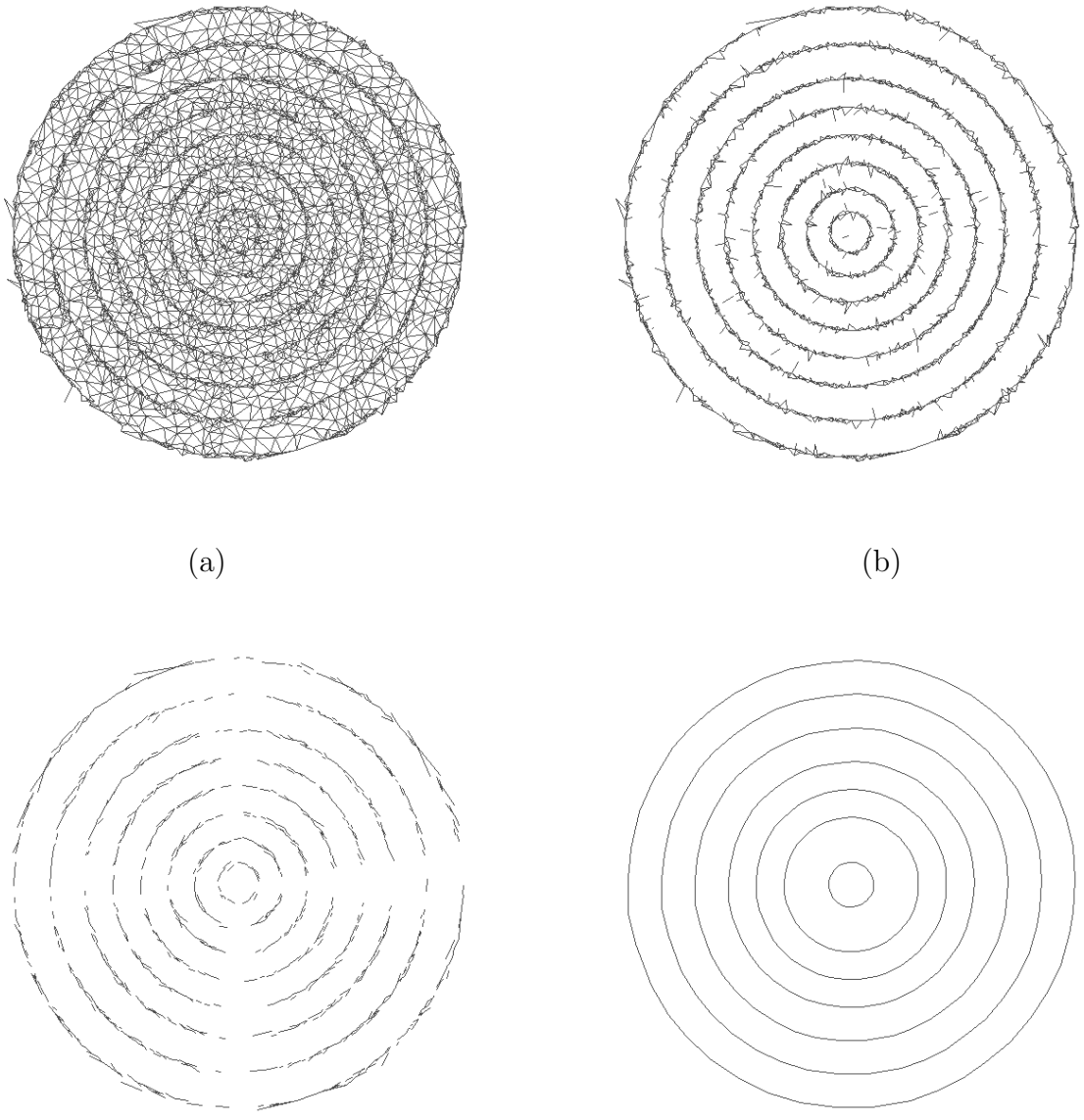
Generación de la malla geométrica

En este caso se utilizó el criterio de área máxima de triángulo con un área máxima de 100 pixeles para refinar los triángulos de la malla, y para refinar los triángulos que no cumplen con este criterio, se escogió insertar su punto de mayor error. Como resultado, se obtuvo una malla bastante ajustada a la imagen, bien refinada alrededor de los anillos, pero demasiado refinada en el fondo, por lo que se desperdician recursos (triángulos) detallando zonas de tono uniforme. Los bordes y gradientes de tono están razonablemente bien definidos, pero menos que en los dos casos anteriores. Además, se utilizaron bastantes más triángulos que en los casos anteriores, 14786 en total; esto sumado a su calidad inferior, hacen a esta malla una peor elección que las anteriores para describir una imagen de anillos de árboles (ver figura 4.6).

Generación de las polilíneas

Filtro por diferencia de tonos Al aplicar este filtro se obtuvo un resultado muy similar al anterior, con la excepción de que no desaparecieron las aristas externas al anillo exterior, pero unidas a él por un vértice. De todas maneras, es un resultado bastante bueno (ver figura 4.7 a).

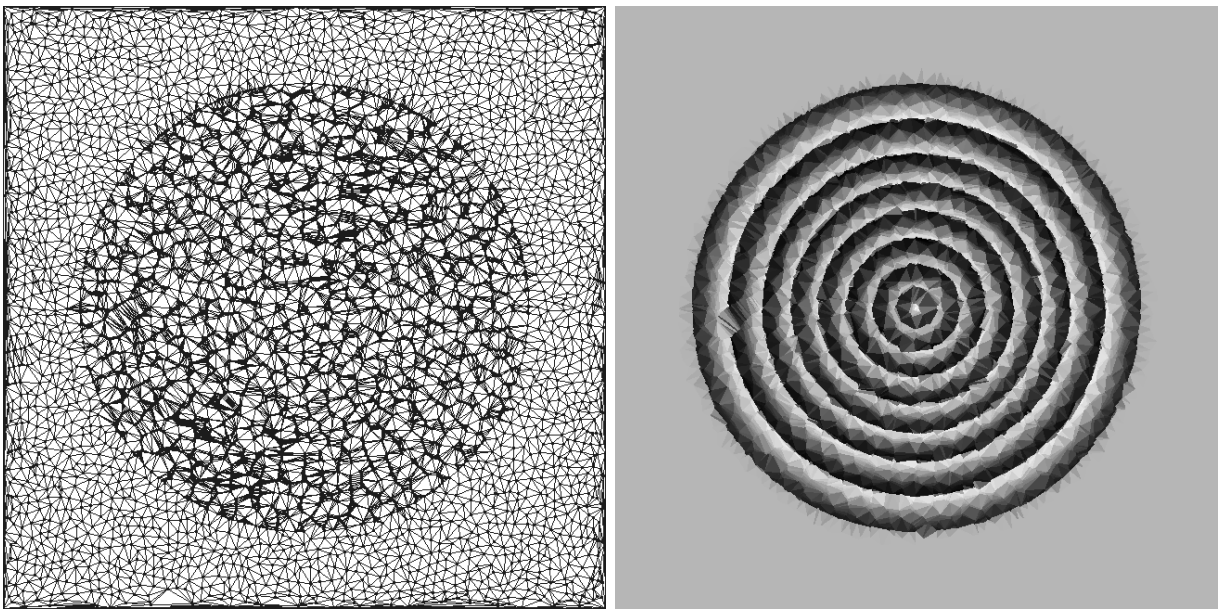
Filtro por orientación respecto al centro Al igual que en el caso anterior, desaparecieron muchas de las aristas que no están sobre el borde de un anillo, pero se mantuvo un poco más de ruido que antes. Además, no desaparecieron las aristas exteriores al anillo externo (ver figura 4.7 b).



(a) (b)

(c) (d)

Figura 4.5: Filtro por diferencia de tonos (a), filtro por orientación respecto al centro (b), filtro por ángulo de inclinación (c), y transformada de Hough generalizada (d) aplicados a la malla PID-TCP.



(a) (b)

Figura 4.6: Triángulos de la malla MTA-MEP (a) y tonos de la misma (b).

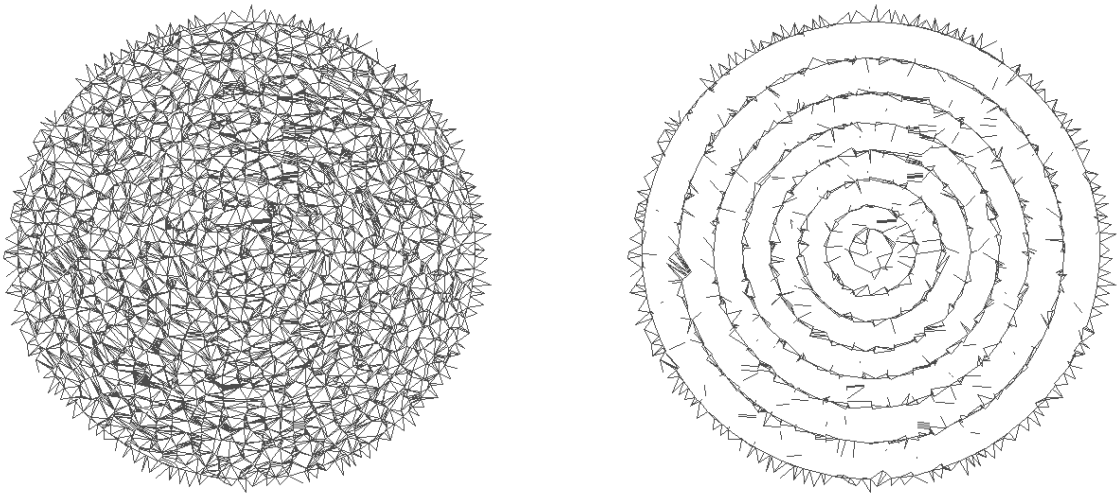
Filtro por ángulo de inclinación El resultado de aplicar este filtro, al igual que en los casos anteriores, es una imagen muy clara de los anillos, casi sin ruido fuera de ellos, y habiendo desaparecido casi todas las aristas exteriores al anillo externo. La imagen no es de tan buena calidad como en el caso de la malla PID-MEP, pero sí comparable a la de la malla PID-TPC (ver figura 4.7 c).

Transformada de Hough Generalizada adaptada a aristas Finalmente se obtiene una aproximación casi perfecta de todos excepto dos de los bordes de los anillos (el segundo y el séptimo, contando desde dentro). Como en el paso anterior se observa claramente que ambos anillos están representados por una buena cantidad de aristas, nuevamente podemos concluir que el problema está en el algoritmo final de selección de anillos (ver figura 4.7 d).

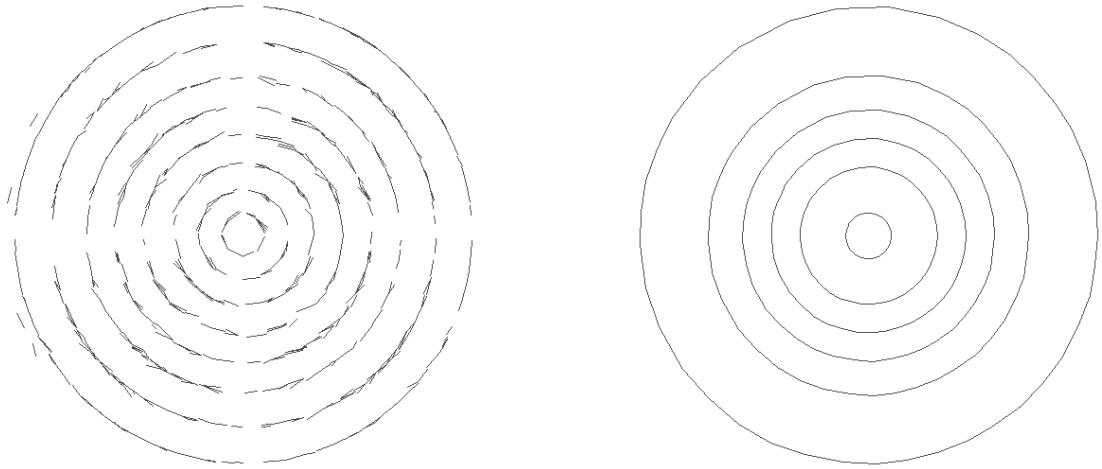
4.1.4. Área máxima de triángulo y punto central del triángulo (Malla MTA-TCP)

Generación de la malla geométrica

En este caso, como en el anterior, también se utilizó el criterio de área máxima de triángulo con una área máxima de 100 pixeles para refinar los triángulos de la malla; pero para refinarlos se aplicó el criterio de inserción del punto central. Cabe destacar que de los cuatro casos estudiados, éste es el único que no tiene ningún criterio que dependa de la imagen que se quiere representar como una malla, sino que exclusivamente del área y forma de los triángulos



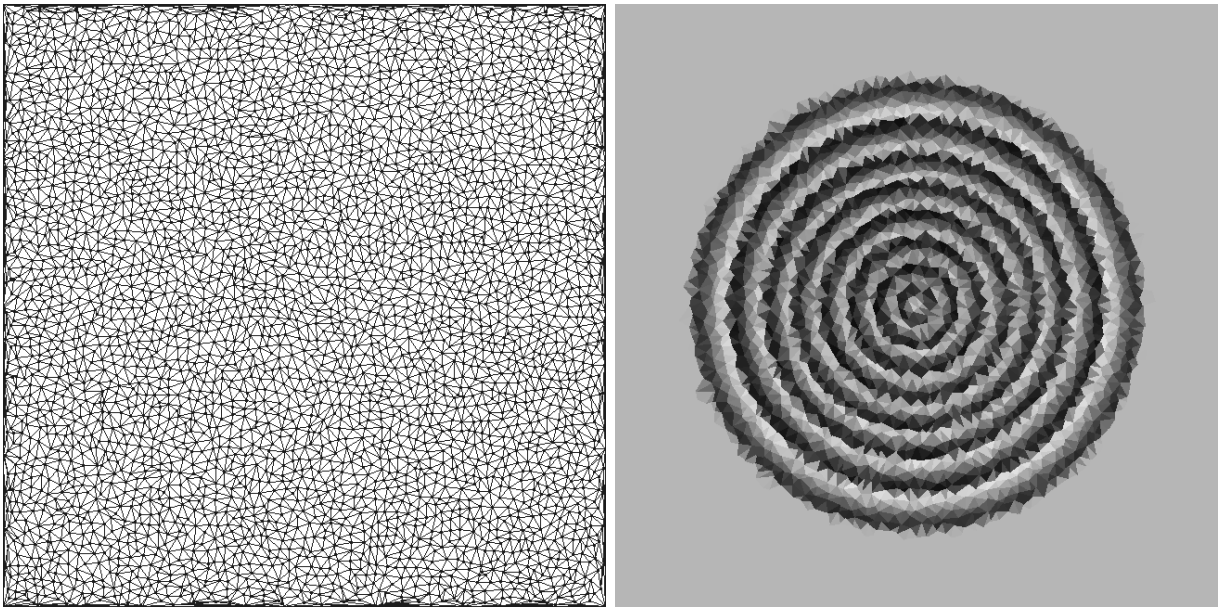
(a) (b)



(c) (d)

Figura 4.7: Filtro por diferencia de tonos (a), filtro por orientación respecto al centro (b), filtro por ángulo de inclinación (c), y transformada de Hough generalizada (d) aplicados a la malla MTA-MEP.

que la componen. El resultado, como es de esperarse por esta razón, es una malla que no presenta ninguna diferencia entre las zonas cercanas a los anillos y el fondo, por lo que la definición de los bordes es nula, y el gradiente de tonos es bastante pobre. El número de triángulos que componen esta malla es el menor de los cuatro casos, 7552 en total; lo que también era de esperarse ya que no hay ningún refinamiento alrededor de los bordes. Esto junto a la rapidez con que se genera, son las únicas ventajas de esta malla frente a las demás, ya que su calidad es muy inferior (ver figura 4.8).



(a) (b)

Figura 4.8: Triángulos de la malla MTA-TCP (a) y tonos de la misma (b).

Generación de las polilíneas

Filtro por diferencia de tonos La aplicación de este filtro sobre la malla produjo un resultado relativamente similar al de los casos anteriores: desaparecieron todas las aristas del fondo, pero muy pocas de las correspondientes al tronco; y las que desaparecieron lo hicieron sin ninguna diferenciación entre las que pertenecen a bordes y las que no (ver figura 4.9 a).

Filtro por orientación respecto al centro Al igual que en casos anteriores, al aplicar este filtro desaparecieron muchas de las aristas que no están sobre el borde de un anillo, pero las que se mantuvieron se ven mucho más separadas entre sí que en los otros casos, y se mantienen una mayor cantidad de aristas que están unidas a un borde por sólo un vértice que en los casos anteriores (ver figura 4.9 b).

Filtro por ángulo de inclinación El resultado de aplicar este filtro es una imagen muy difusa de los anillos, de mucho peor calidad que en los otros casos. Las aristas que se conservan tienen una inclinación parecida a los mismos, pero están muy disconexos entre sí, y no se adivina a simple vista dónde están (ver figura 4.9 c).

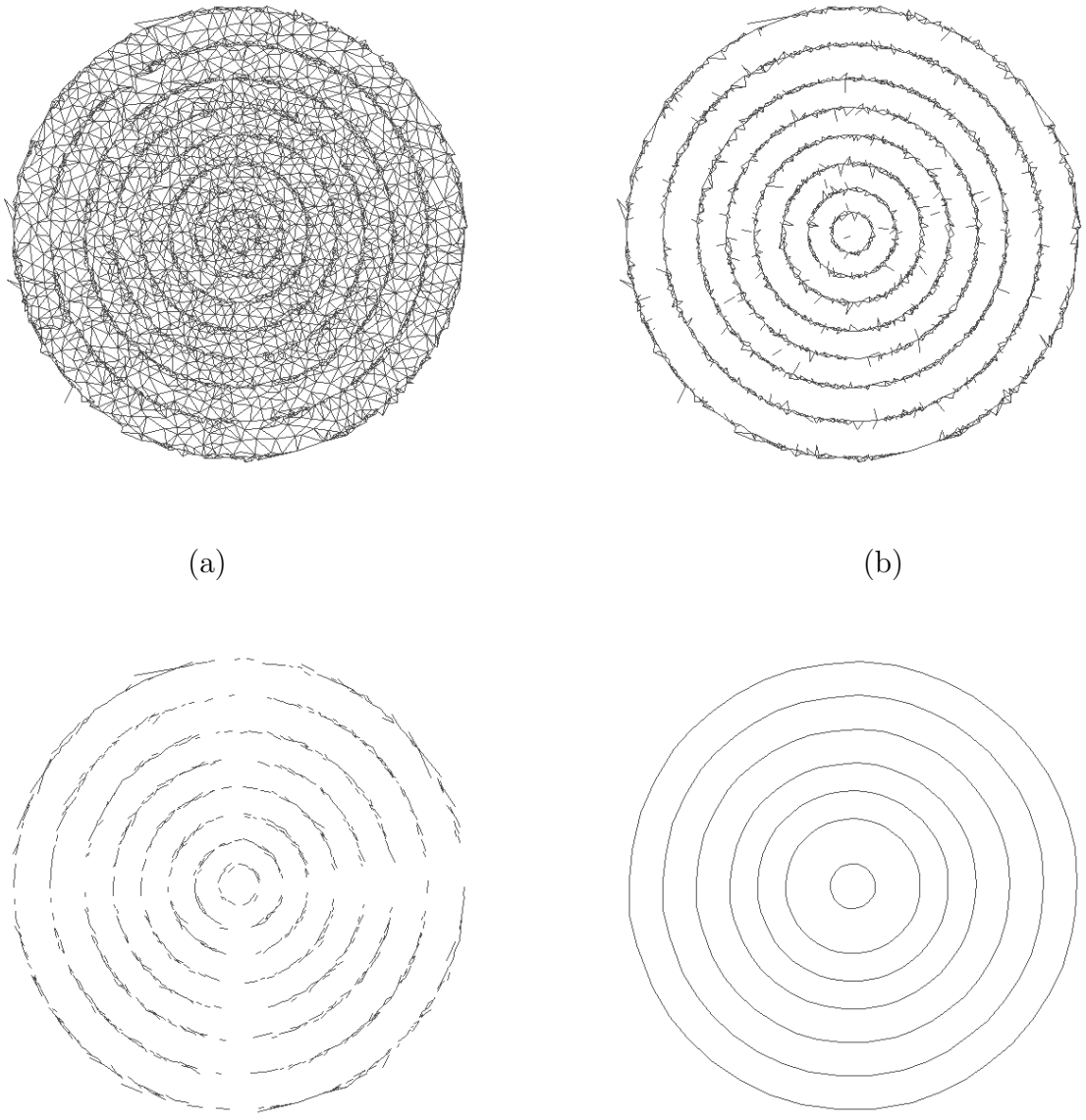
Transformada de Hough Generalizada adaptada a aristas Finalmente, el resultado de aplicar el generador de polilíneas es una muy buena aproximación de todos los anillos excepto uno (el primero contando desde dentro), lo que se explica por la robustez de la transformada de Hough, ya que este método se basa en buscar un patrón predefinido, y dado un cierto conjunto de aristas , determina dónde tiene una mayor probabilidad de estar, independiente de la calidad del conjunto. En este caso, el anillo que no apareció está representado en el conjunto de aristas, pero de manera tan pobre que es posible que simplemente pasara por ruido y por esa razón no fuera considerado como un anillo. En caso contrario, sería causa del mismo probema de selección final de anillos que ocurrió en los dos casos anteriores (ver figura 4.9 d).

4.2. Aplicación de los algoritmos de filtrado de imágenes a imágenes fabricadas

En esta sección se mostrarán los resultados de aplicar filtros a la misma imagen fabricada de la sección anterior, que representa la imagen ideal del corte del tronco de un árbol (ver figura 4.1), pero con diversas imperfecciones que dificultarían extraer sus anillos. La finalidad de aplicar estos filtros a una imagen es neutralizar el efecto de dichas imperfecciones, para pemitir a los algoritmos de generación de mallas y polilíneas trabajar correctamente. Para comparar el efecto de los diversos filtros en la generación de mallas, se generó una malla con iguales criterios y parámetros a algunas de las imágenes utilizadas en esta sección. Los criterios y parámetros escogidos fueron: criterio de diferencia de intensidad de pixeles con una diferencia de intensidad mínima de 100 para refinar los triángulos de la malla, y el criterio del punto de mayor error para escoger el punto a insertar en un triángulo. Se eligió esta malla porque fue probada y dio muy buenos resultados en la sección anterior.

4.2.1. Normalización

En este caso, ya que el propósito de la normalización es mejorar el contraste de una imagen, se tomó una imagen con bajo contraste y se generó una malla a partir de ella utilizando los criterios antes descritos. El resultado fue una malla poco detallada (la cantidad total de triángulos es de 3018), con anillos relativamente bien delineados en la parte interior del tronco, pero con un mal borde externo del mismo; además, los triángulos resultaron tener

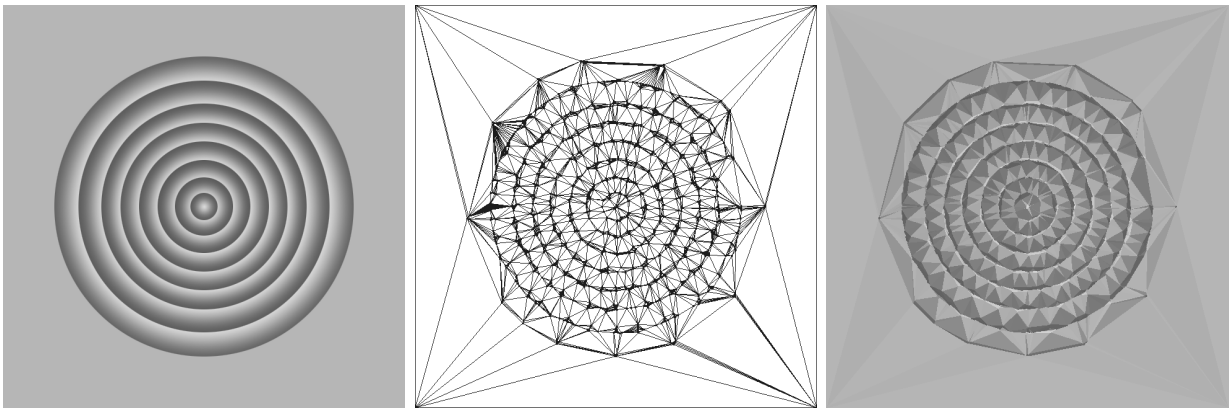


(a) (b)

(c) (d)

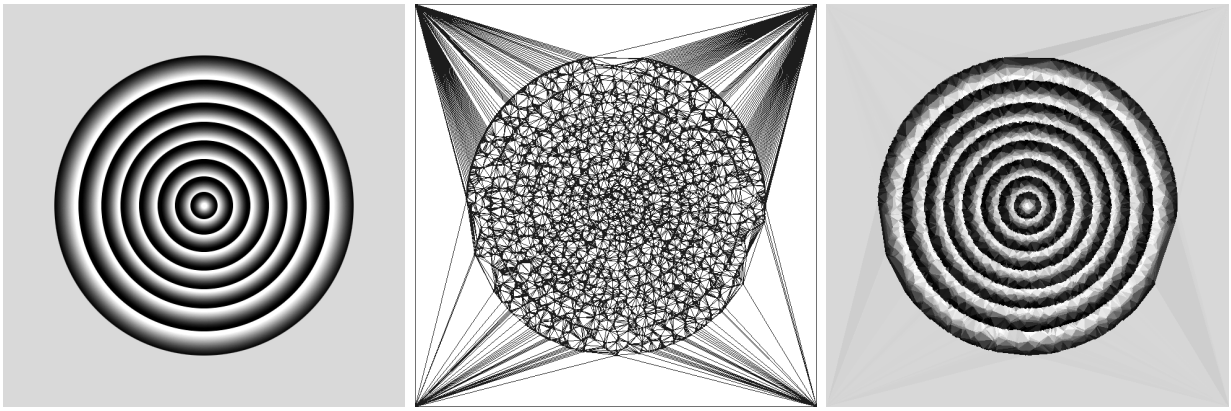
Figura 4.9: Filtro por diferencia de tonos (a), filtro por orientación respecto al centro (b), filtro por ángulo de inclinación (c), y transformada de Hough generalizada (d) aplicados a la malla MTA-TCP.

tan poca diferencia de tono que podrían causar problemas en la generación de polilíneas (ver figura 4.10). A continuación se aplicó un filtro de normalización considerando el 90 % central del histograma a la imagen, y se generó nuevamente la malla. El resultado obtenido fue una malla mucho más detallada, compuesta de 12206 triángulos, con una mucho mejor definición de bordes y diferenciación de tonos entre triángulos de diferentes anillos (ver figura 4.11).



(a) (b) (c)

Figura 4.10: Imagen con poco contraste (a) y su malla resultante (b) y (c).



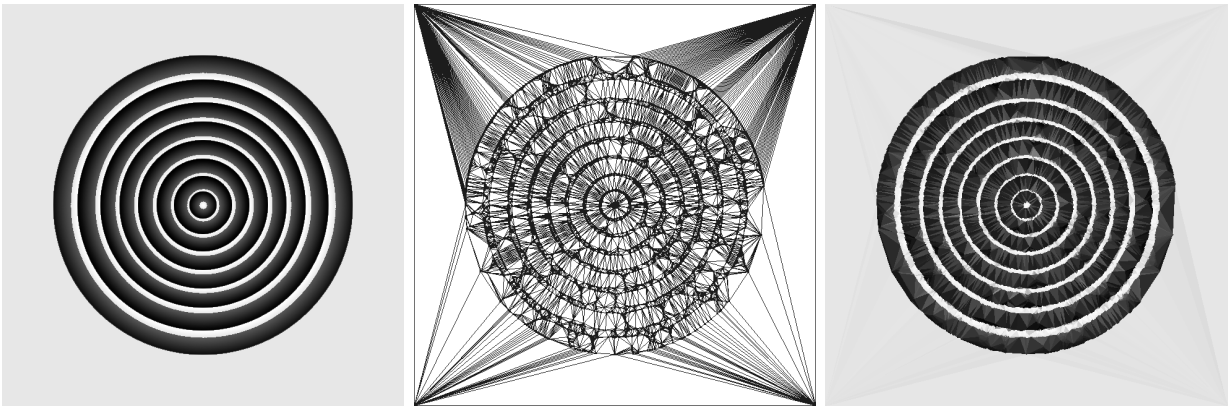
(a) (b) (c)

Figura 4.11: Imagen normalizada (a) y su malla resultante (b) y (c).

4.2.2. Ecualización

Ya que el objetivo de este filtro, al igual que el de la normalización, es mejorar el contraste de la imagen, se utilizó la misma imagen de prueba que en el caso anterior (ver figura 4.10). La imagen obtenida luego de aplicar este filtro tiene un excelente contraste, pero pierde detalle del gradiente de tono de cada anillo, lo que podría hacer pensar que esta imagen generaría una malla de inferior calidad a la anterior; sin embargo, el resultado obtenido de la generación de

la malla a partir de la imagen ecualizada es una malla de muy buena calidad, con un número de triángulos similar a la anterior (con un total de 12254) pero más detallada y suave en los bordes, y con una excelente diferenciación de tonos entre triángulos de diferentes anillos (ver figura 4.12).



(a) (b) (c)

Figura 4.12: Imagen ecualizada (a) y su malla resultante (b) y (c).

4.2.3. Umbral

El objetivo de aplicar este filtro a una imagen de anillos de árboles es clasificar los pixeles de la misma en dos categorías: los que pertenecen a un borde, y los que no. Cuando la imagen no presenta imperfecciones, este filtro funciona muy bien, pero es demasiado dependiente del valor umbral; si éste está mal escogido, la imagen puede quedar casi completamente blanca o negra. El filtro se probó sobre la imagen de prueba original (ver figura 4.1) utilizando tres valores de umbral: 50, 5 y 240 (ver figura 4.13).

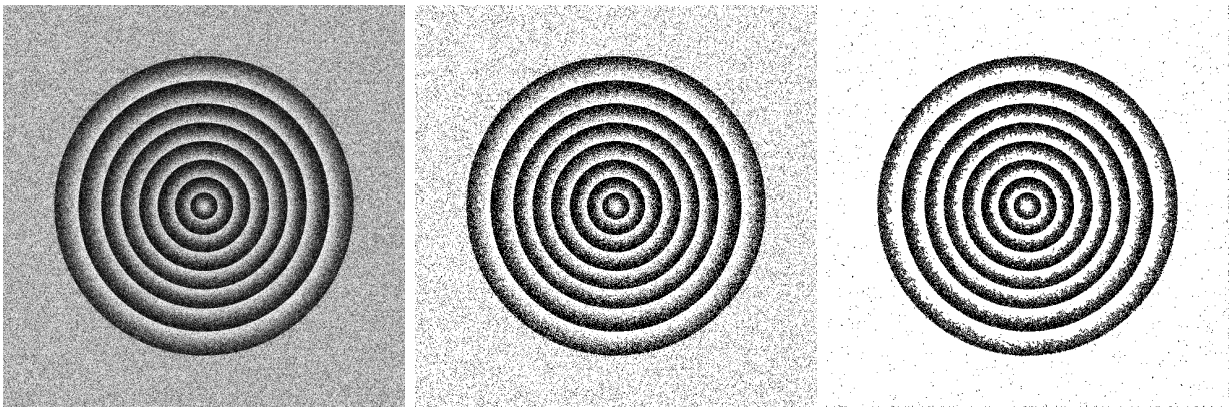
(a) (b) (c)

Figura 4.13: Filtro de umbral aplicado a la imagen, con un umbral de 50 (a), 5 (b) y 240 (c).

Sin embargo, este filtro tiene otros problemas aparte de su sensibilidad al valor umbral: funciona bastante mal en caso de que la imagen tenga mucho ruido, o tenga luminosidad desigual. Para estos casos se pueden utilizar las siguientes modificaciones del filtro.

Doble umbral

En el caso en que la imagen tiene ruido del tipo salt & pepper (ver figura 4.14 a), al aplicar el filtro de umbral con un umbral adecuado da como resultado una imagen también ruidosa (ver figura 4.14 b). Si, en cambio, se aplica el filtro de doble umbral sobre la misma imagen, se puede eliminar gran parte de ese ruido, manteniendo la calidad de los anillos (ver figura 4.14 c).



(a) (b) (c)

Figura 4.14: Imagen original con ruido (a), con filtro de umbral de 100 (b), y con filtro de doble umbral de 10 y 100 (c).

Umbral adaptativo

En el caso en que la imagen tiene grande diferencias de luminosidad, como por ejemplo un gradiente de luminosidad (ver figura 4.15 a), ni el filtro de umbral ni el de doble umbral funcionan correctamente, ya que puede ocurrir que el umbral tenga un valor entre el tono máximo y mínimo de los anillos, en cuyo caso sólo se identificará parte de los mismos (ver figura 4.15 b); o bien el umbral estará completamente sobre o bajo los tonos máximo y mínimo de los anillos, en cuyo caso éstos se confundirán con el fondo. Además, si el umbral está entre los tonos máximo y mínimo del fondo, parte de éste también será identificado como perteneciente al borde de un anillo (ver figura 4.15 c).

El umbral adaptativo, en cambio, no sufre de ninguno de estos problemas, ya que para cada pixel selecciona su valor umbral en forma local, por lo que los cambios de luminosidad entre distintas zonas de la imagen no lo pueden afectar (ver figura 4.15 d). Además, su único

parámetro es el radio del área que se considera para calcular el umbral de cada pixel, y el algoritmo resulta ser mucho menos sensible a variaciones en su parámetro que los otros filtros de umbral. El único problema de este filtro es su gran costo en términos de cálculo, que resulta ser varios órdenes de magnitud superior al de los otros dos; sin embargo, sus excelentes resultados muchas veces ameritan su utilización.

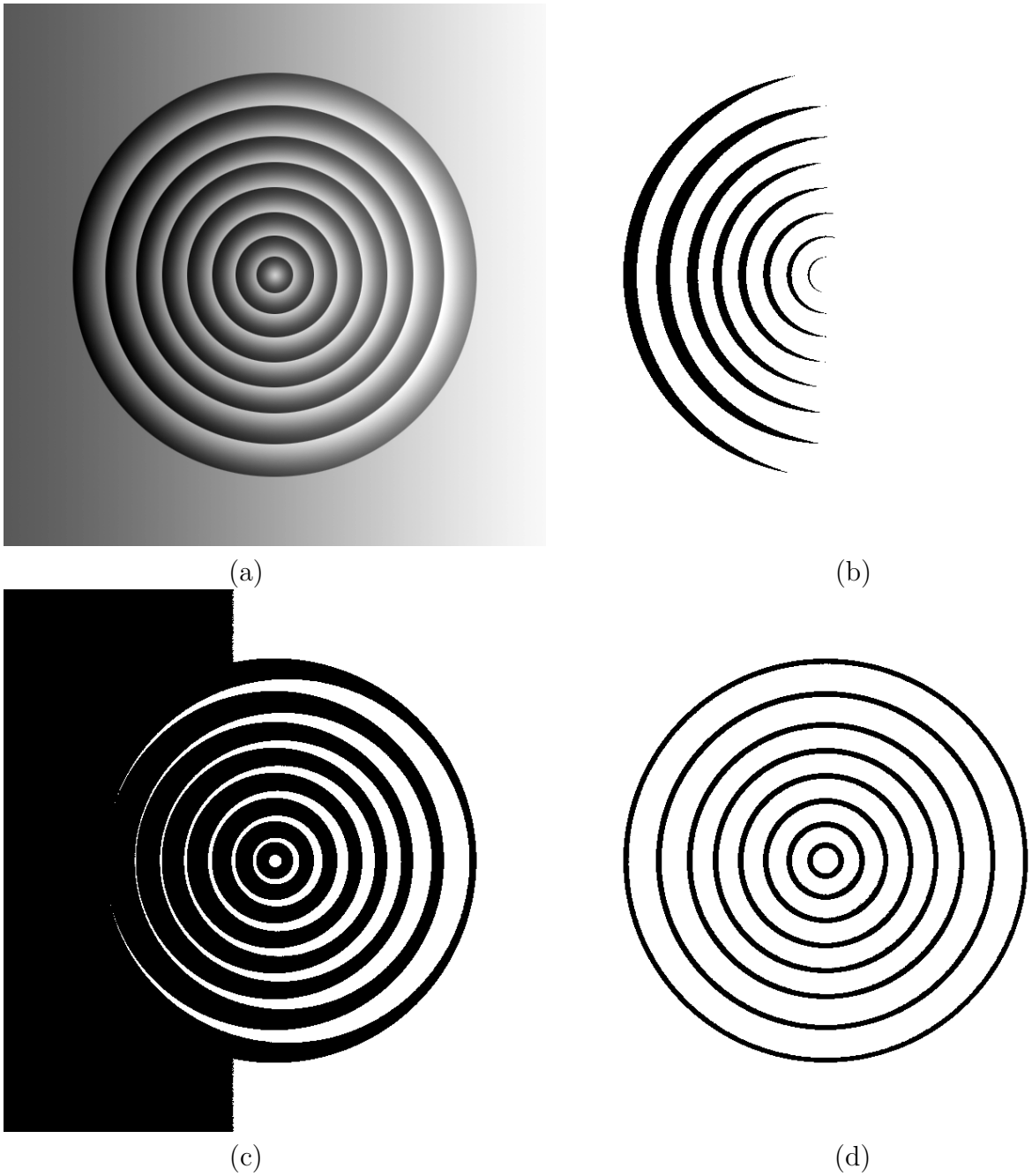
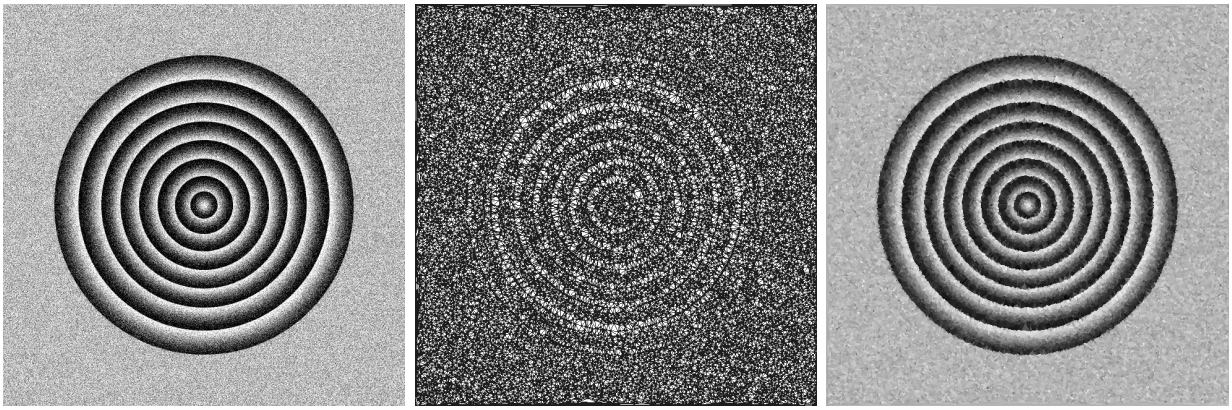


Figura 4.15: Imagen original con gradiente de luminosidad (a), con filtro de umbral de 50 (b), con filtro de doble umbral de 50 y 150 (c), y con filtro de umbral adaptativo de 7 pixeles de radio (d).

4.3. Filtro Gaussiano

El objetivo de este filtro es reducir los niveles de ruido de una imagen. En el caso de las imágenes de anillos de árboles que son las utilizadas en esta memoria, esto es importante para posteriormente generar una malla y polilíneas a partir de la misma, ya que la presencia de ruido genera una malla innecesariamente refinada y con imperfecciones. Sin embargo, se debe tener cuidado con la intensidad con la que se aplica, ya que el filtro gaussiano tiene el problema de disminuir la nitidez de la imagen, lo que en casos extremos confunde los bordes de la misma.

Para probar este filtro, primero se utilizó la imagen original con ruido y con un filtro de normalización ocupando el 90 % central del histograma (ver figura 4.16 a), a partir de la cual se generó una malla utilizando los mismos criterios que en el resto de los casos de esta sección. Se puede observar que la malla generada tiene un nivel innecesario de detalle, especialmente en el fondo; está compuesta de 53826 triángulos (ver figuras 4.16 b y c).

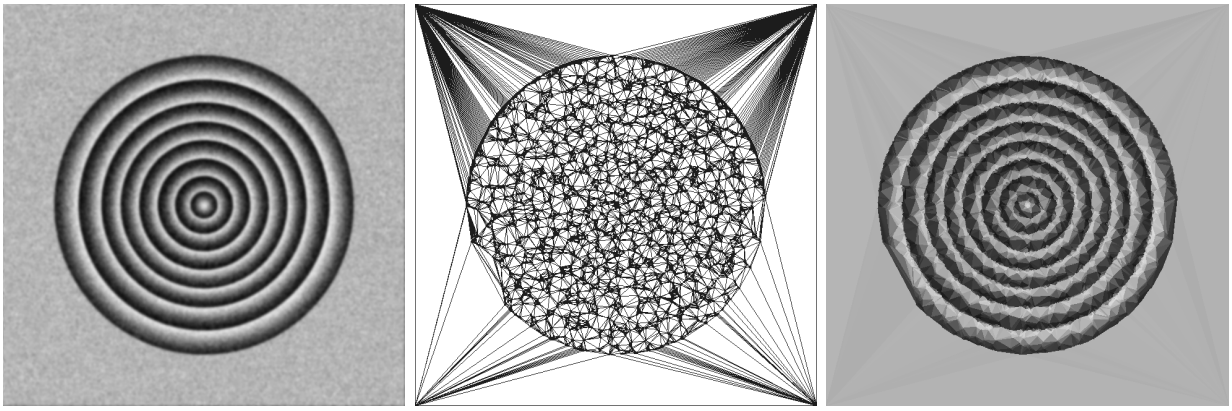


(a) (b) (c)

Figura 4.16: Imagen original con ruido (a) y la malla que genera (b) y (c).

A continuación, se probó aplicar el filtro gaussiano a la imagen con tres valores de intensidad σ: 2, 4 y 8, que corresponden a niveles *ligero*, *medio* e *intenso*. Posteriormente, a todas las imágenes se les aplicó un filtro de normalización ocupando el 90 % central del histograma, ya que el filtro gaussiano tiende a desaprovechar gran parte del rango dinámico de la imagen.

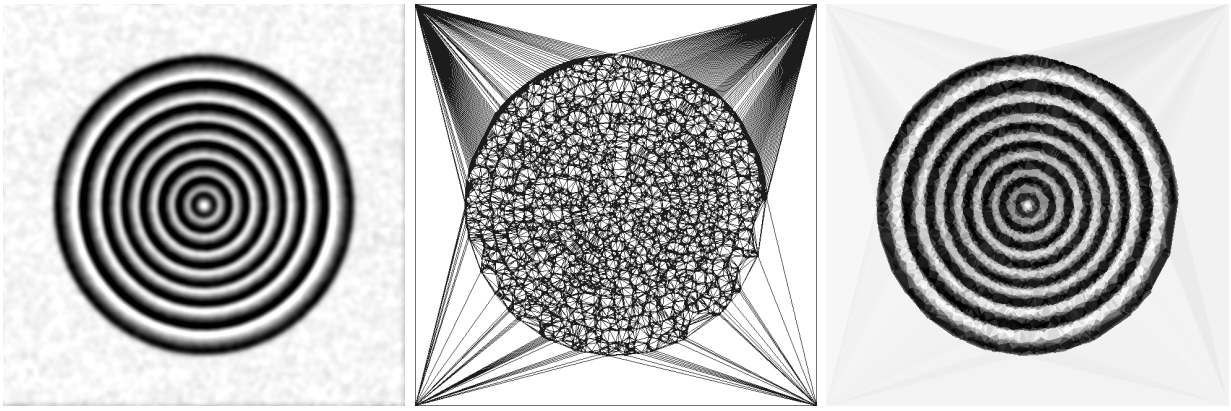
Al aplicar el filtro en el nivel ligero (con $\sigma = 2$), se observa una mejora substancial tanto en la imagen como en la malla que genera; el ruido casi desaparece por completo, y el efecto de pérdida de nitidez es lo suficientemente suave como para que se distingan bien los bordes. La malla generada tiene un muy buen nivel de detalle, estando compuesta de 11546 triángulos; y mantiene una diferencia de tonos adecuada entre los mismos (ver figura 4.17).



(a) (b) (c)

Figura 4.17: Imagen con ruido y filtro gaussiano con $\sigma = 2$ (a) y la malla que genera (b) y (c).

Al aplicar el filtro en el nivel medio (con $\sigma = 4$), se observa que el ruido en la zona del tronco desaparece completamente, y en el fondo queda reducido a manchas de bajo contraste, lo que es mucho mejor que el ruido tipo salt & pepper de la imagen original; hay una pérdida de nitidez más pronunciada que en el caso anterior, pero aún se distinguen adecuadamente los bordes. Además, la nueva imagen tiene un contraste mucho mayor que en el caso anterior. La malla generada tiene algo más de detalle que la anterior, ya que se compone de 17854 triángulos, y este detalle se concentra en los bordes. Los triángulos de la malla resultante tienen una diferencia de tonos aún mayor que en el caso anterior entre triángulos de distintos anillos, lo que ayuda a la detección de sus bordes. En general, se puede decir que esta imagen y la malla que genera es de superior calidad que en el caso anterior (ver figura 4.18).

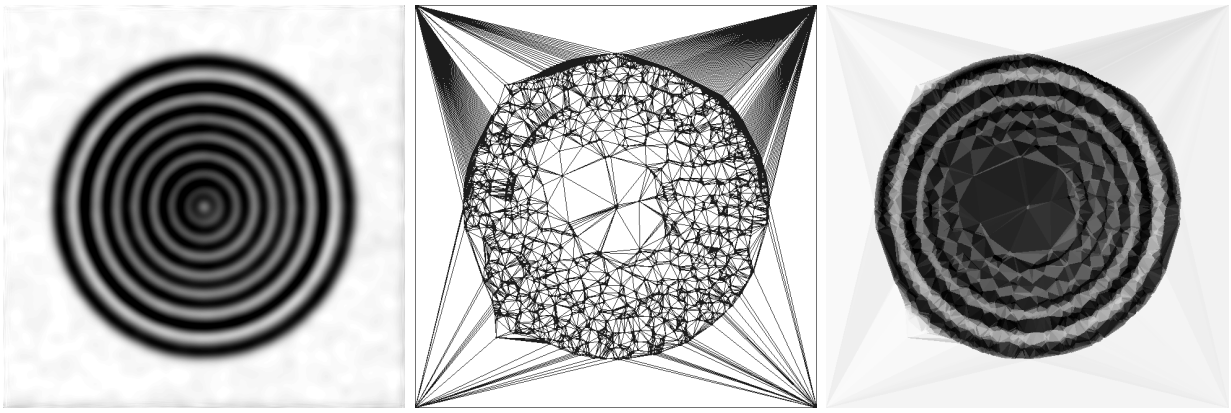


(a) (b) (c)

Figura 4.18: Imagen con ruido y filtro gaussiano con $\sigma = 4$ (a) y la malla que genera (b) y (c).

Finalmente, al aplicar el filtro en el nivel intenso (con $\sigma = 8$), se observa que el ruido del fondo caso ha desaparecido totalmente, pero la imagen está demasiado borrosa como para trabajar con ella. Los bordes de los anillos no se distinguen claramente, y el contraste entre

ellos disminuyó. La malla generada es de pésima calidad, ya que hay zonas dentro del tronco donde no se distingue entre anillos (en particular, alrededor del centro del tronco), sino que se ve una zona de tono uniforme (ver figura 4.19).



(a) (b) (c)

Figura 4.19: Imagen con ruido y filtro gaussiano con $\sigma = 8$ (a) y la malla que genera (b) y (c).

En general, se puede decir que el filtro gaussiano es una muy buena herramienta para la eliminación de ruido, pues aplicado en un nivel lo suficientemente bajo eliminará gran parte del mismo, sin afectar mucho la calidad de la imagen; y en particular, la nitidez de los bordes.

4.4. Aplicación de todo el procedimiento a una imagen fabricada difícil

En esta sección, se utilizarán los algoritmos antes vistos para detectar los anillos en una imagen fabricada. Esta imagen, al igual que las anteriores, representa los anillos de un árbol, pero además presenta muchos problemas: tiene bajo contraste, ruido del tipo *salt & pepper*, cortes en direcciones al azar dentro y fuera del área del tronco, e iluminación desigual (ver figura 4.20 a). El objetivo de esta prueba es detectar los anillos de la imagen; para esto, se utilizarán los algoritmos vistos en esta sección según sea conveniente.

Primero, se ataca el problema del ruido *salt & pepper*. Para esto, se utiliza el filtro gaussiano, con $\sigma = 4$ (ver figura 4.20 b). A continuación, se mejora el contraste de la imagen aplicando la normalización, utilizando el 90 % central del histograma (ver figura 4.20 c). El último filtro que se aplica es el umbral adaptativo con un radio de 6 pixeles, para eliminar el problema de la iluminación desigual en la imagen (ver figura 4.20 d).

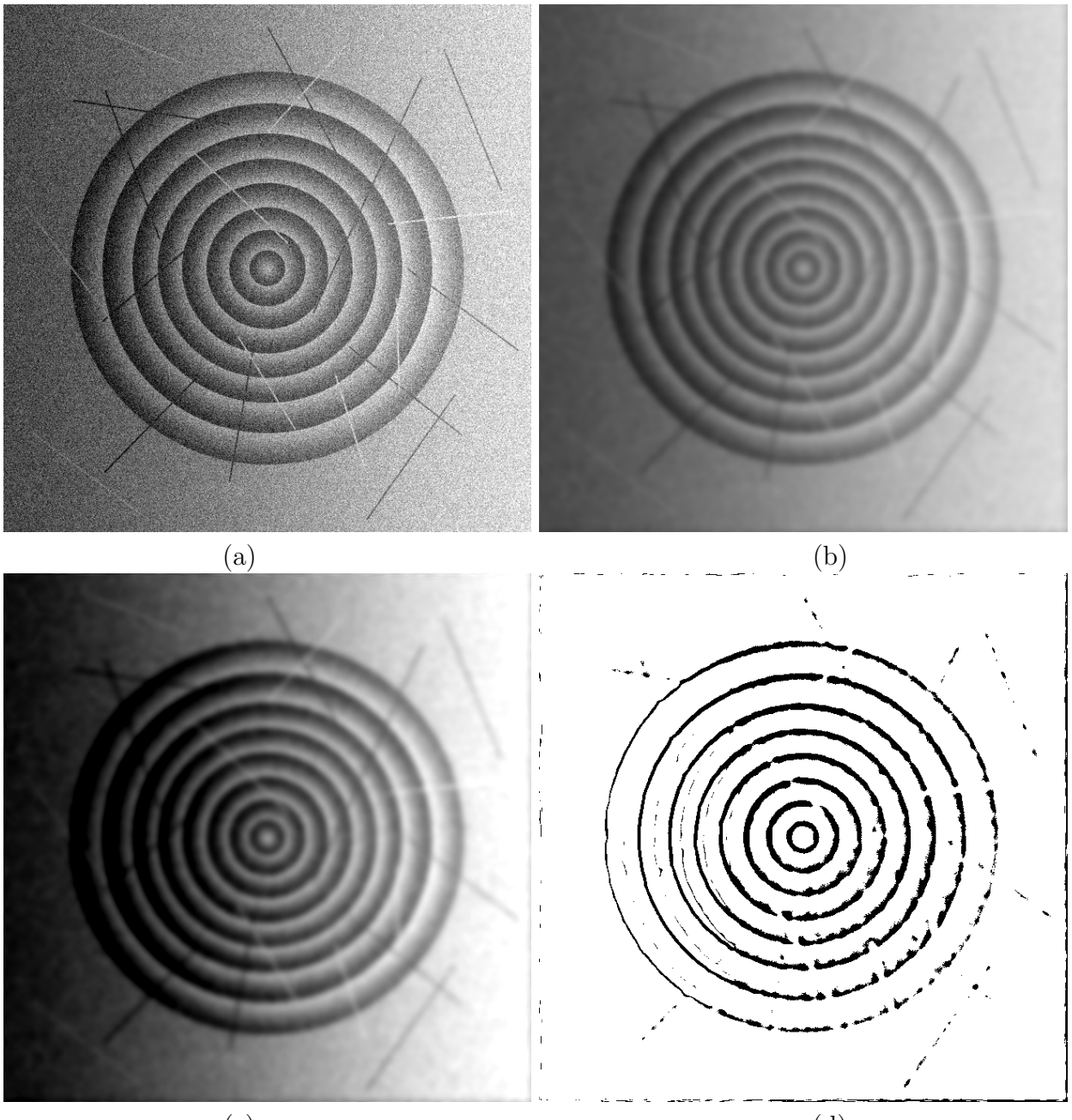
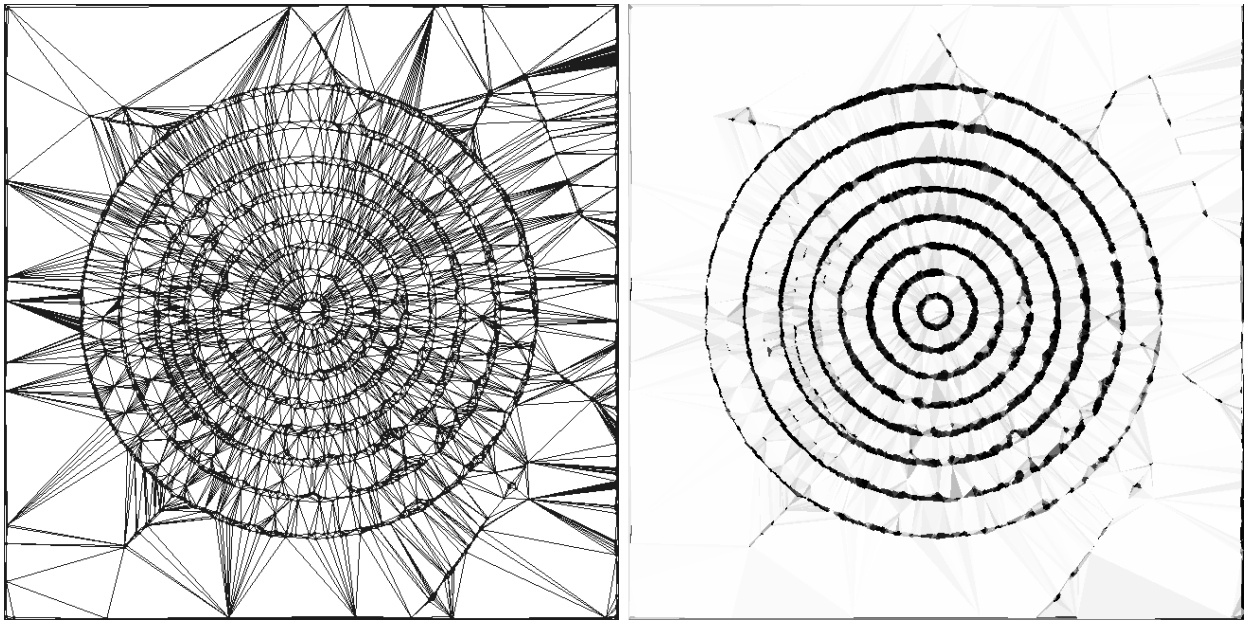


Figura 4.20: Imagen fabricada del corte de un árbol con bajo contraste, ruido *salt & pepper*, cortes e iluminación desigual (a); misma imagen con filtro gaussiano con $\sigma = 4$ (b), normalización utilizando el 90 % central del histograma (c) y umbral adaptativo con un radio de 6 pixeles (d).

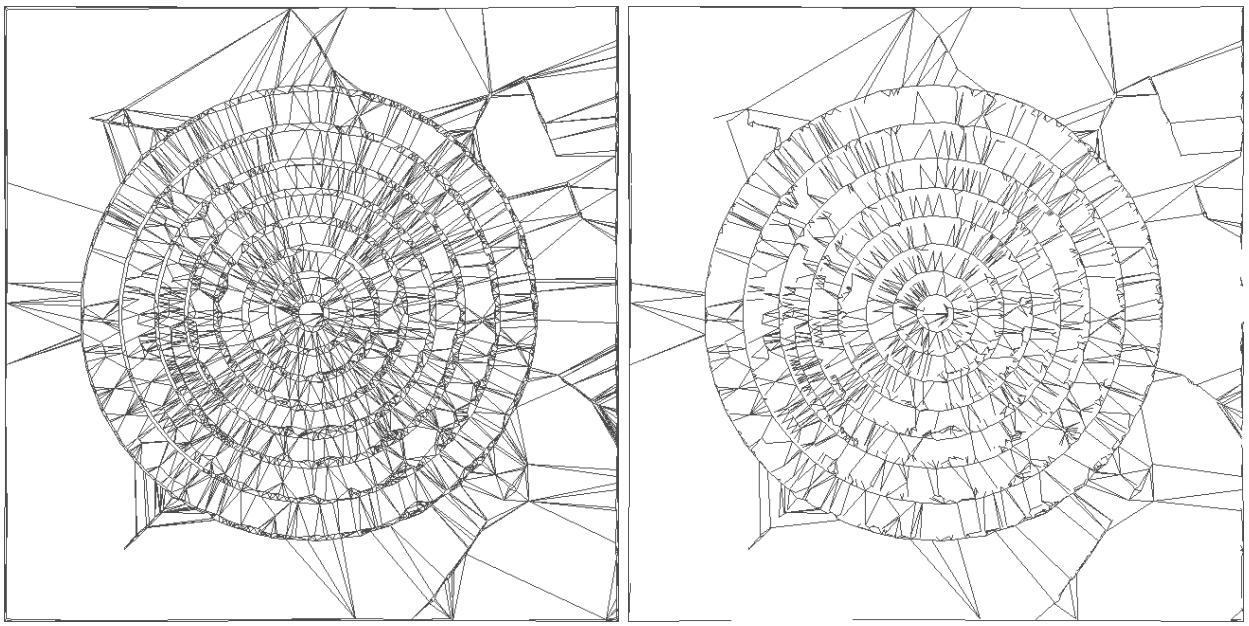
Se puede ver que la imagen generada es de excelente calidad, con muy poco ruido y bordes de anillos muy bien definidos, mientras que los cortes han desaparecido en gran medida. Una vez que la imagen se ha filtrado hasta dejarla lo mejor posible se debe generar la malla geométrica. Al igual que en todas las pruebas anteriores de esta sección, se utiliza el criterio de diferencia de intensidad de pixeles con una diferencia de intensidad mínima de 100 para refinar los triángulos de la malla, y el criterio del punto de mayor error para escoger el punto a insertar en un triángulo. El resultado obtenido es una malla de muy buena calidad, compuesta de 7766 triángulos, bien detallada alrededor de los anillos pero no mucho en el fondo, y con un excelente contraste entre triángulos de diferentes anillos (ver imágenes 4.21 a y b).



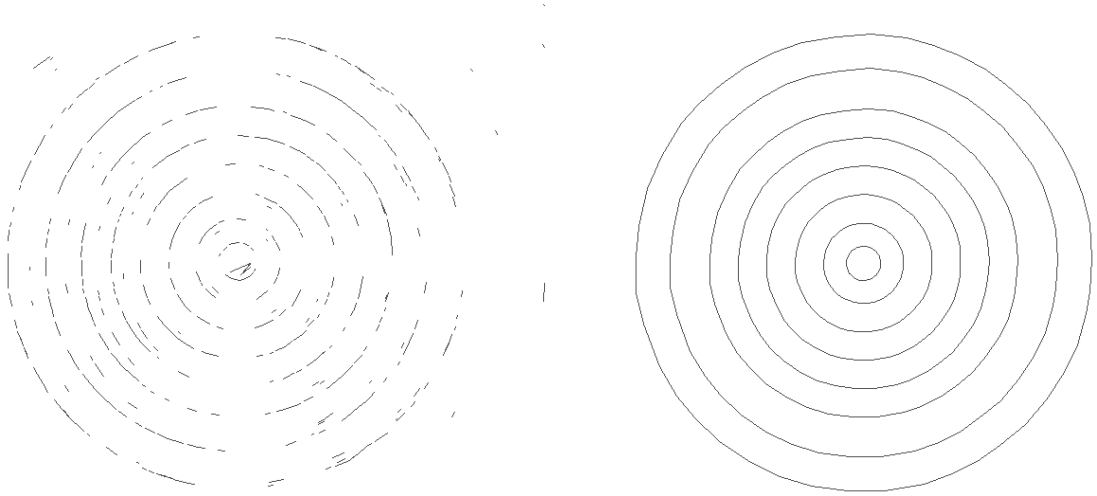
(a) (b)

Figura 4.21: Malla de la imagen 4.20 a (a), y sus tonos (b).

Finalmente, se generan las polilíneas a partir de la malla geométrica. Para esto, se aplican sucesivamente el filtro por diferencia de tonos con una diferencia de 5 (ver figura 4.22 a), el filtro por orientación respecto al centro (ver figura 4.22 b), el filtro por ángulo de inclinación con un parámetro de 0.8 (ver figura 4.22 c), y por último la transformada de Hough generalizada adaptada a aristas (ver figura 4.22 d). El resultado obtenido son 8 polilíneas que corresponden perfectamente a los 8 anillos de la imagen.



(a) (b)



(c) (d)

Figura 4.22: Filtro por diferencia de tonos (a), filtro por orientación respecto al centro (b), filtro por ángulo de inclinación (c) y transformada de Hough generalizada (d) aplicados a la malla 4.21 a.

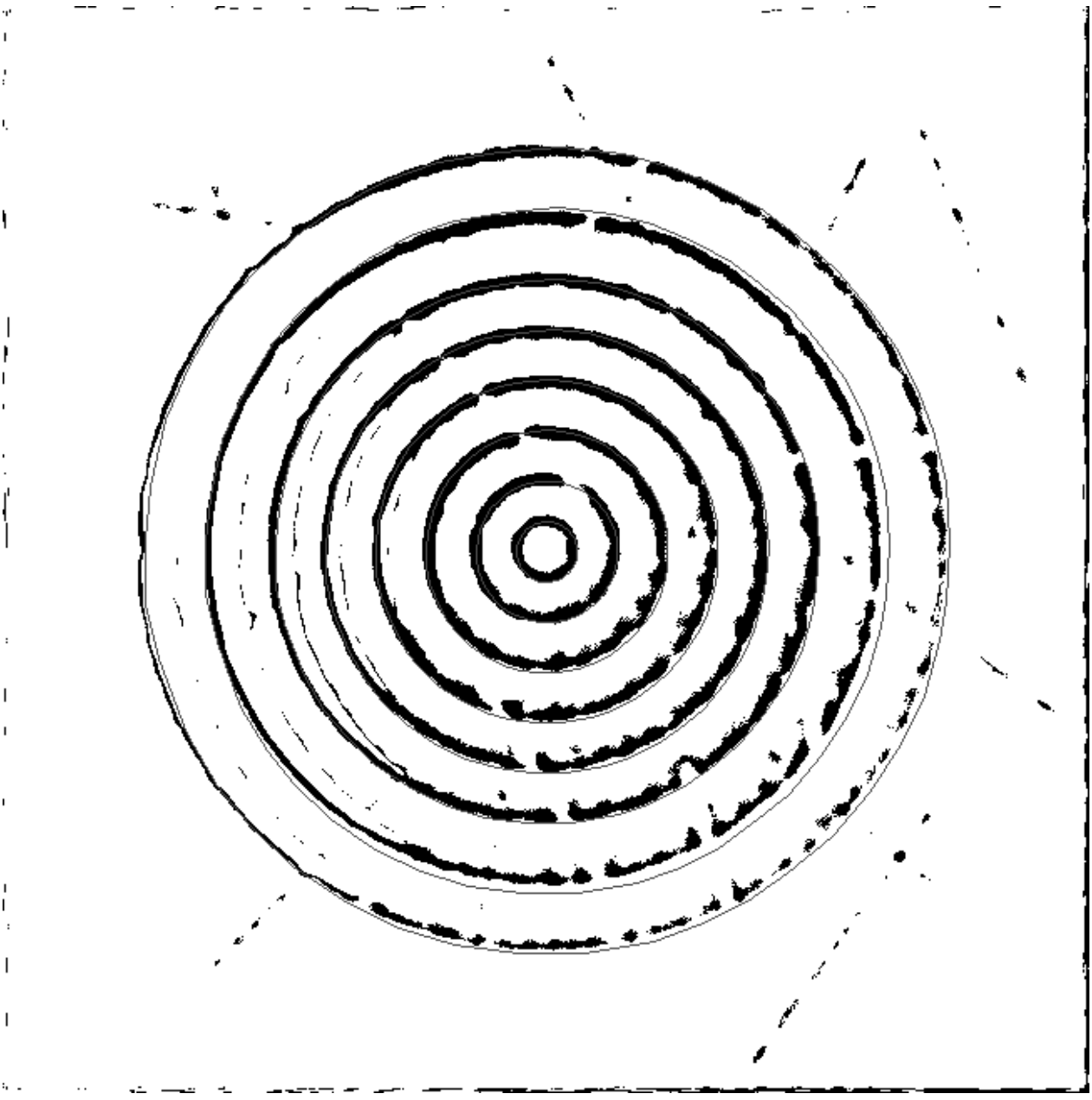


Figura 4.23: Superposición de la imagen difícil filtrada y sus polilíneas.

4.5. Aplicación del todo el procedimiento a imágenes reales

En esta sección, se utilizarán los algoritmos vistos en este capítulo para detectar los anillos de algunas imágenes reales de cortes de árboles, con diverso grado de dificultad. El objetivo de esta prueba es determinar qué tal útil es el procedimiento implementado en esta memoria para detectar anillos de árboles en imágenes reales.

Imagen de dificultad baja

La primera imagen es de dificultad baja, ya que tiene un buen contraste entre anillos, una luminosidad uniforme, poco ruido, y tiene anillos muy parecidos a una circunferencia, muy parecidos entre sí, bastante separados y sin grandes nudos que los deformen. El resultado de aplicar el procedimiento de detección de anillos da como resultado un conjunto de polilíneas muy cercanas a los bordes de los anillos, sin falsos positivos (líneas que no corresponden a ningún anillo) ni falsos negativos (anillos faltantes). Por lo tanto, el resultado del procesamiento de esta imagen es excelente (ver figura 4.24).

Imagen de dificultad media

Esta imagen tiene algo más de dificultad que la anterior, ya tiene un contraste menos (aunque sin llegar a ser bajo), y un gran nudo que atraviesa tres de sus anillos. Además, los anillos más cerca de la corteza son menos parecidos a los más cercanos al centro que en el caso anterior. Sin embargo, la robustez del algoritmo de Hough permitió una buena detección de los anillos; el resultado obtenido muestra que no hubo falsos positivos ni negativos, y que los anillos encontrados aproximan bastante bien a los reales, pese a que la calidad de la aproximación decae un poco en los anillos más cercanos al centro del tronco (ver figura 4.25).

Imagen de dificultad alta

La tercera imagen es bastante más dificil que las anteriores, ya que su contraste entre anillos es muy bajo y tiene una considerable cantidad de ruido en la madera, probablemente causado por la manera en que el tronco fue aserrado. Además, el tronco tiene algunas manchas en la madera; en particular, alrededor del centro del tronco, ya que esa zona es mucho más clara que el resto del mismo. En este caso la detección de anillos cometió dos errores. El primero es que hubo un falso negativo, ya que no se detectó el segundo anillo contando desde dentro; observando la imagen podemos deducir que esto ocurrió debido a que ese anillo se encuentra dentro de la mancha antes mencionada, lo que le dio un contraste extremadamente bajo. El segundo error es que la polilínea que representa al tercer anillo contando desde dentro es demasiado pequeño; nuevamente este error se puede atribuir a una mancha en la madera,

Figura 4.24: Imagen real de dificultad baja y sus anillos obtenidos.

Figura 4.25: Imagen real de dificultad media y sus anillos obtenidos.

esta vez oscura, que se encuentra por dentro de ese anillo. Sin embargo, pese a los errores encontrados, el procesamiento de esta imagen es satisfactorio (ver figura 4.26).

Figura 4.26: Imagen real de dificultad alta y sus anillos obtenidos.

Imagen de dificultad muy alta

La última imagen es mucho más difícil que todas las anteriores, ya que tiene un contraste relativamente bajo (aunque mayor que la imagen anterior), cortes muy profundos en la madera, manchas y grandes nudos; además, los anillos están muy juntos entre sí, y los más

cercanos a la corteza son muy diferentes a los cercanos al centro del tronco. Esta última dificultad es la que causa mayores problemas, ya que la modificación al algoritmo de Hough generalizado utilizado para detectar los anillos se fundamenta en que todos los anillos serán versiones escaladas del anillo exterior; como en este caso eso no se cumple, difícilmente la detección será buena. El resultado obtenido de esta imagen confirma este pronóstico, ya que sólo los anillos exteriores fueron detectados correctamente; los anillos medios, que no están cerca ni del centro y de la corteza, fueron detectados con grandes problemas de precisión; y los anillos interiores simplemente no fueron detectados. Por lo tanto, el resultado del procesamiento de esta imagen se considera insatisfactorio, pero esperable dado la dificultad de la imagen y la implementación del algoritmo de detección de anillos (ver figura 4.27).

Figura 4.27: Imagen real de dificultad muy alta y sus anillos obtenidos.

Capítulo 5

Conclusiones

Durante el desarrollo de este informe se ha mostrado que la aplicación combinada de algoritmos de filtrado de imágenes y generación de mallas geométricas funciona mejor que aplicarlos por separado, ya que la generación de una malla geométrica a partir de una imagen se basa en la extracción de determinadas características relevantes de ésta, las que pueden ser resaltadas mediante la utilización de filtros. Además, la utilización de diversos filtros pensados para atacar problemas específicos de la imagen (como por ejemplo, falta de contraste, presencia de ruido o luminosidad desigual) permiten relizar esta tarea con mayor precisión. Sin embargo, esto también acarrea algunos problemas, principalmente de automatización; ya que para distintas imágenes hay diferentes combinaciones de filtros y algoritmos de generación de mallas que resultan óptimos, y para cada uno de ellos, diferentes parámetros que determinar para su correcto funcionamiento. Pese a estos problemas, los buenos resultados obtenidos de la aplicación del método descrito a lo largo de esta memoria indican que es un paso en la dirección correcta.

Otro punto muy importante que se consideró es el diseño de la herramienta. La utilización de un buen diseño orientado a objetos junto con un lenguaje de programación robusto y enfocado a la orientación a objetos, permitió un desarrollo modular y extensible, al que fácilmente se le pueden agregar nuevos algoritmos, criterios y tipos de mallas. Esto permite realizar fácilmente tanto la extensión de la herramienta para continuar perfeccionando el método de detección de anillos de árboles desarrollado, como su utilización para otros problemas relacionados con la generación de mallas geométricas.

Una manera en que se puede seguir desarrollando este trabajo en el futuro es mejorar la automatización del método implementado. La selección de los filtros a aplicar a las imágenes y sus parámetros, los criterios de refinamiento e inserción de puntos, y los parámetros de los algoritmos de generación de polilíneas, incluyendo el punto central y el anillo exterior del tronco; todas éstas son tareas que actualmente se realizan manualmente por el usuario, por

lo que automatizarlas representaría una mejora substancial. También es importante seguir extendiendo la herramienta con nuevos filtros de imágenes, tipos de mallas (por ejemplo, campos de altura), algoritmos y criterios de refinamiento, y especialmente, algoritmos de generación de polilíneas. En el caso de éstos últimos, puede tratarse de una modificación al existente, basado en la transformada de Hough generalizada, para comportarse mejor en el caso de imágenes de troncos con anillos exteriores muy diferentes a los exteriores; y también algoritmos completamente nuevos. Finalmente, también es posible mejorar la interfaz de la herramienta desarrollada, para hacerla más fácil de utilizar e incluir en ella una mayor cantidad de información.

Bibliografía

[1] Steven L. Horowitz, Theodosios Pavlidis, "Picture segmentation by a tree traversal algorithm", *Journal of the Association for Computing Machinery*, vol.23, nº 2, pp. 368-388. 1976.

[2] Theo Pavlidis, Yuh-Tay Llow, "Integrating region growing and edge detection", *IEEE transactions on pattern analysis and machine intelligence*, vol. 12, nº 3, pp. 225-233. 1990.

[3] T. Gevers, A. W. M. Smeulders, "Combining region splitting and edge detection through guided Delaunay image subdivision", *Proceedings of the 1997 Conference on Computer Vision and Pattern Recognition*, pp. 1021-1026. 1997.

[4] Lifeng Liu, Stan Sclaroff, "Region segmentation via deformable model-guided split and merge", Computer Science Department, Boston University. 2000.

[5] Hayet Laggoune, "Tree Ring Analysis", Départament d'informatique, Université du Québec à Montreal. 2005.

[6] W. Steven Conner, "A computer vision based tree ring analysis and dating system", Master's thesis, Department of Electrical & Computer Engineering, University of Arizona. 1999.

[7] John Francis Canny, "Finding edges and lines in images", Master's thesis, Artificial Intelligence Laboratory , Massachusetts Institute of Technology, Cambridge. 1983.

[8] María Cecilia Bastarrica, Nancy Hitschfeld Kahler, "Designing a product family of meshing tools", *Advances in Engineering Software*, nº 37, pp. 1-10. 2005.

[9] Michael Garland, Paul S. Heckbert, "Fast triangular approximation of terrains and height fields", Computer Science Department, Carnegie Mellon University. 1997.

[10] Mauricio Cerda Villablanca, "Reconocimiento de bordes en imágenes, un enfoque aplicado", Memoria de Ingeniería Civil, Departamento de Ciencias de la Computación, Facultad de Ciencias Físicas y Matemáticas, Universidad de Chile. 2006.

[11] Cristián Maluenda Miranda, "Experimentación con mallas geométricas para el reconocimiento de anillos de árboles", Memoria de Ingeniería Civil, Departamento de Ciencias de la Computación, Facultad de Ciencias Físicas y Matemáticas, Universidad de Chile. 2004.

[12] Mauricio Cerda Villablanca, "Robust tree-ring detection", Departamento de Ciencias de la Computación, Universidad de Chile. 2007.

[13] Erich Gamma, Richard Helm, Ralph Johnson, John Vlissides, "Design Patterns: Elements of reusable object oriented sotware", Addison Wesley. 1995.