UNIVERSIDAD VERACRUZANA

INSTITUTO DE INVESTIGACIONES EN INTELIGENCIA ARTIFICIAL

PROCESAMIENTO MULTIESCALA DE IMÁGENES MEDIANTE KERNELS BASADOS EN ALGORITMOS EVOLUTIVOS PARA LA SEGMENTACIÓN DE TELARAÑAS Y LA CUANTIFICACIÓN DEL DAÑO ESTRUCTURAL

T E S I S

Para obtener el título de:

Maestro en Inteligencia Artificial

Presenta:

Ing. Jaime Rangel Ojeda

Director:

Dr. Horacio Tapia-McClung

22 de septiembre de 2025

Agradecimientos

Quisiera agradecer a mi familia por su apoyo incondicional, el cual me ha permitido alcanzar este importante logro en mis estudios.

A mi director de tesis, el Dr. Horacio Tapia-McClung, por su guía y el tiempo dedicado a la realización de este trabajo.

Al equipo de investigadores del Instituto de Biotecnología y Ecología Aplicada (INBIOTECA), por brindarme la oportunidad de realizar visitas, así como por su guía, orientación y explicaciones.

A mis profesores del instituto, por sus explicaciones y consejos.

Índice general

Resumen

El presente trabajo se centra en estimar de forma aproximada el daño estructural de una telaraña a partir de un registro fotográfico. Posteriormente, se procesarán las imágenes para segmentarlas mediante un algoritmo, lo que permitirá registrar, evaluar y cuantificar el daño sufrido por la telaraña. Esta información es de gran utilidad en los ámbitos biológico y ecológico.

Dada la complejidad que implica realizar un registro manual utilizando herramientas de medición como Fiji ImageJ, se hace necesaria la implementación de un algoritmo que automatice este proceso.

Los algoritmos que se proponen son los siguientes: un algoritmo de segmentación de imágenes basado en kernels que integra técnicas de cómputo evolutivo y visión por computadora. Se plantea iniciar con la detección de bordes en imágenes para posteriormente realizar un proceso de segmentación con operaciones morfológicas, seguido de una técnica basada en kernels usando una función de aptitud que maximiza la detección de píxeles que representan las regiones de interés (ROI).

La efectividad del algoritmo detector de bordes propuesto se evaluó empleando la base de datos BSDS500 como referencia y los resultados del algoritmo propuesto frente a los métodos clásicos de detección de bordes muestran que, en el análisis de F-score realizado sobre la base de datos BSDS500, el método no supera a los métodos tradicionales.

El segundo método se basa en la segmentación por umbralización local usando dos algoritmos Niblack y Feng ampliamente utilizados para la segmentación de texto en imágenes de baja calidad.

Como última propuesta, se utiliza una red neuronal convolucional (CNN) tomando como imágenes de muestra para el entrenamiento las imágenes segmentadas por los algoritmos de Niblack y Feng que tiene la ventaja de no necesitar parámetros para realizar una segmentación de una nueva imagen proporcionada al modelo, pero como desventaja es el tiempo y ajuste de parámetros de entrenamiento.

Los resultados muestran que el modelo entrenado basado en U-Net, utilizando las imágenes segmentadas con los algoritmos de Niblack y Feng que fueron analizadas y calificadas por el experto, alcanzó un rendimiento de F-Score del 90 % y 80 % respectivamente según los cálculos realizados a partir de la matriz de confusión.

Al realizar una comparación entre el algoritmo propuesto y los resultados previamente mencionados, se llevaron a cabo diversas pruebas ajustando parámetros como el tamaño del kernel y los umbrales. Estas pruebas arrojaron un rendimiento del algoritmo con un F-Score del 80 % y 65 %, respectivamente.

Esto evidencia que el algoritmo propuesto presenta una desventaja debido al ajuste de dos parámetros clave que influyen directamente en su rendimiento y tiempo de cómputo: la condición de paro y el tamaño de la población inicial al emplear cómputo evolutivo.

Capítulo 1

Introducción

La detección de bordes es un problema fundamental en el procesamiento de imágenes y visión por computadora. El ruido y la calidad de la imagen afectan significativamente la extracción de características y la detección de áreas de interés. Existen en la literatura diferentes métodos para la detección de bordes así como métodos para segmentación de imágenes (Ramesh Jain, 1995, Rafael C. Gonzales, 2002), pero estos métodos por sí solos no son suficientes para lograr un resultado óptimo que permita una interpretación aceptable en la mayoría de los casos.

Los métodos tradicionales de detección de bordes emplean distintas perspectivas y pueden agruparse en las siguientes categorías: Detectores basados en gradientes (aproximaciones a la primera derivada), detectores laplacianos (basados en cruces por cero) y algoritmos de aproximación de imágenes (Jain et al., 2016): Por ejemplo, los detectores basados en gradientes presentan la limitación de no capturar con precisión la localización real de los bordes.

Por ello, existen diferentes investigaciones que centran su enfoque en lograr detectar "bordes débiles" y estos consisten en bordes que aparecen en imágenes con bajo contraste y ruido (Galun et al., 2007, Cui et al., 2021, M. Zhang et al., 2022) ya que poder detectar la mayor cantidad de bordes presentes en una imagen permiten realizar una mejor interpretación del contenido de la misma, unas propuestas consisten en detección de espacio de curvas, umbrales óptimos y la construcción de filtros que se ajustan con las curvas trazadas por los bordes (Hutchison et al., 2010). También es útil la detección de características globales y locales mediante el uso de kernels para detectar regiones en las imágenes y obtener una representación de estructuras (J. Zhang et al., 2006). Otras investigaciones se centran en cambiar el enfoque de la detección de bordes, donde se buscan detectar estos bordes tratando de eliminar el ruido presente en la imagen, utilizando la representación multi-escala que proporciona por ejemplo la arquitectura U-Net entrenada para aplicar reducción de ruido (Ofir y Keller, 2021). Finalmente, otro enfoque aplica la detección de bordes combinando dos metodologías que consisten en aplicar la detección de bordes rectos y curvos en varias escalas con la ayuda de umbrales (Ofir et al., 2020).

En este trabajo se llevó a cabo una revisión de la literatura y el estudio de diferentes técnicas que proponen mejorar la detección de bordes y segmentación de imágenes en el área biológica, pero el alcance de este estudio también puede abarcar otras áreas donde sea

necesario detectar regiones finas con detalles difíciles de detectar en imágenes complejas donde las fronteras entre objetos pueden ser débiles debido a propiedades similares en ambos lados del borde, efectos de sombreado u otros factores. La aplicación de estas técnicas están enfocadas en las telarañas y se describe brevemente su diseño, estructura e importancia ambiental y ecológica. En colaboración con investigadores de la INBIOTECA que detallan los estudios actuales sobre la importancia de las telarañas y sus presas.

1.1. Definición del problema

El problema que se plantea consiste en desarrollar un algoritmo que permita detectar bordes de manera precisa para poder segmentar regiones de interés, en particular con un interés en imágenes de telarañas, con el objetivo de calcular su daño estructural. Esta información es relevante debido a que permite tener un registro físico del comportamiento de las arañas y sus presas.

La investigación se realiza en colaboración con la INBIOTECA de la Universidad Veracruzana, donde se cuenta con materiales y equipo para poder realizar estos estudios. La figura (1.1) muestra las áreas que son utilizadas para este fin.

(a) Área para la toma de fotografías de las cosechas de telarañas.

(b) Especímen de araña en almacenamiento.

Figura 1.1: Descripción de las áreas y especímenes.

Estimar el daño estructural de las telarañas puede proporcionar información importante para hacer una estimación de la densidad de presas. Cuantificar el daño de manera manual no es muy preciso, las técnicas actuales utilizan fotografías y realizan estimaciones manualmente (por vista) mediante un software especializado para realizar mediciones con base en puntos de referencia; distancia del punto de observación, tamaño de objetos en primer plano, etc.

Debido a lo anterior segmentar las imágenes de telarañas de forma automatizada por medio de un algoritmo de precisión aceptable constituye una mejora. La figura (1.2) muestra las imágenes originales de la base de datos, la figura (1.3) muestra un ejemplo en el que se han eliminado las regiones sin importancia de las fotografías tomadas antes y después

de un daño y la figura (1.4) muestra la aproximación utilizando técnicas de segmentación de esas fotografías utilizando umbralización local con los algoritmos de Niblack y Feng.

(a) Antes (b) Después

Figura 1.2: Imágenes originales extraídas de la base de datos que muestran los marcos de aluminio de referencia, la figura (a) es la foto antes del daño estructural y la figura (b) es la foto después del daño estructural.

(a) Antes (b) Después

Figura 1.3: Ejemplo del daño estructural en una telaraña, la figura (a) es la foto antes del daño estructural y la figura (b) es la foto después del daño estructural.

(a) Antes (b) Después

Figura 1.4: Aproximaciones a segmentación de áreas de interés utilizando el método de umbralización local de Niblack.

Debido a que no existen imágenes de referencia segmentadas de las áreas de interés (es decir una imagen objetivo), se define como aproximación al ejemplo mostrado en la figura (1.4).

Muchos ecólogos creen que la correlación de los insectos que son atrapados en las telarañas y la captura de las presas se debe a la relación entre el tamaño del insecto y la distancia entre las espirales en la superficie de la telaraña así como la orientación de las mismas (Chacon y Eberhard, 1980 , Eberhard, 1980). Esto refleja una falta de comprensión de los estímulos sensoriales específicos a los que responden los insectos y la maniobrabilidad de los insectos durante el vuelo libre. Estudios de laboratorio han demostrado que los insectos son capaces de detectar y evitar diferentes tipos de telarañas (Rypstra, 1982), observaciones preliminares muestran que las gotas viscosas que recubren la espiral de la telaraña son el componente más importante que afecta la visibilidad de la telaraña (Craig, 1990), sin embargo la visibilidad de una telaraña puede variar dependiendo de los detalles del diseño, el hábitat y la hora del día (Craig, 1988).

La función principal de las telarañas es actuar como elementos pasivos para captura de presas; las redes no actúan simplemente como filtros y existen al menos tres funciones principales en la captura de presas: interposición de la telaraña en el camino de las presas (interposición), absorción del impulso de la presa sin romperse (detención), adhesión y/o enredo de la presa para retenerla hasta que llegue la araña para atacar (retención).

Intercepción de presas: La selección de zonas de alimentación es fundamental para la mayoría de los animales. Estudios sugieren (Lubin et al., 1993, Mcnett y Rypstra, 2000, Rittschof y Ruggles, 2010) que las arañas mueven sus telarañas en respuesta a la baja disponibilidad de presas y las arañas reducen la creación de seda cuando construyen en un nuevo sitio. Así, la recolocación de telaraña está influenciada por factores del medio ambiente como presión de depredación, competencia por interferencia y daños en la telaraña.

El tamaño total de la telaraña juega un papel claro y directo en el número total de insectos que interactúan con ella, todo parece indicar que mientras más grande la telaraña parece ser mejor; sin embargo, el tejido de la telaraña está limitado por la cantidad total de captura que una araña puede producir (Eberhard, 1988).

Detención de presas: Las telarañas generalmente interactúan más con insectos voladores y saltadores que no pueden ser capturados hasta que su fuerza cinética sea disipada sin romper la telaraña. La importancia de detener el vuelo de los insectos se cita a menudo como un factor selectivo principal que favorece la evolución de las impresionantes propiedades materiales de las sedas en las telarañas (Eberhard, 1976 ,Kelly et al., 2011).

Retención de presas: Muchos insectos escapan antes de que sean atacados por las arañas, esto da una gran importancia a la capacidad de las telarañas para adherirse a los insectos y a la respuesta rápida de las arañas. El volumen de material invertido en los hilos de captura de una telaraña está relacionado con el peso de la araña, las rañas primitivas tejedoras de orbes producen hilos de captura cribelares y las tejedoras de orbes modernas producen hilos de captura adhesivos, como el hilo de captura adhesivo logra una mayor adherencia en relación con su volumen, las redes orbiculares adhesivas presentan una adherencia superior y un mayor potencial de captura de presas que las redes orbiculares cribelares (Galun et al., 2007). La espiral de captura de una telaraña moderna es una estructura notablemente compleja y es producida por una tríada de espiráculos en las hileras posteriores de las arañas (Nentwig, 1982, Blackledge y Zevenbergen, 2006).

La retención de presas está determinada en última instancia por una fuerte interacción

entre la fuerza adhesiva de los hilos viscosos individuales, la arquitectura de las telarañas, las características de las superficies de los insectos y los comportamientos de escape de los insectos, lo que hace difícil predecir cuán variados serán los tipos de insectos retenidos por las telarañas (Blackledge et al., 2011).

Las telarañas son catalogadas en tres categorías basadas solamente en la apariencia y el modo que la araña opera. La figura (1.5) ilustra una representación de las diversas estructuras construidas por las arañas, estas son:

1. **Las telarañas enredadas o telarañas de maraña:** Consisten en un laberinto irregular de seda tendida entre un sustrato tridimensional.

2. **Las telarañas de sábana:** Consisten en un laberinto de hilos de barrera dentro del cual hay una densa sábana de seda.

3. **Las telarañas orbiculares:** Son estructuras circulares bidimensionales. Una espiral de seda viscosa se teje alrededor de un área central desde la cual se extienden varios radios.

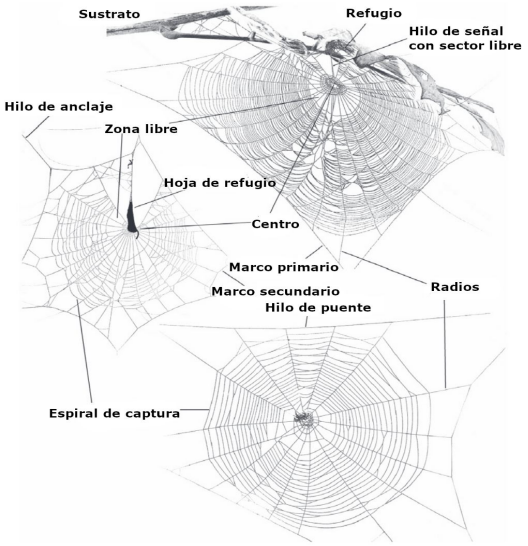


Figura 1.5: Representaciones de diversas telarañas (Blackledge et al., 2011, p. 180)

Algunos insectos toman acciones evasivas para evitar las telarañas y otras claramente realizan maniobras alrededor de las telarañas. Así la función de interposición en al menos las telarañas diurnas probablemente se vean influenciadas por su visibilidad. El incrementar la visibilidad de la telaraña puede ocasionar una reducción en la frecuencia del impacto de la presa y esta visibilidad está probablemente influenciada por el diámetro y densidades de las líneas, tamaños y densidades de las esferas adhesivas en la espiral y el fondo de la telaraña (Eberhard, 1990).

Los elementos de la estructura principal de una telaraña son:

- **Línea de puente**: Consisten en líneas ancladas al sustrato (superficie o material donde la telaraña es anclada).
- **Líneas de marco**: Líneas que se conectan directamente al sustrato u otras líneas en la red.
- **Radio**: También llamadas líneas de radio conectan las líneas de marco con el centro de la red llamada núcleo.
- **Espiral de captura**: que es un hilo continuo que se adhiere a los radios.

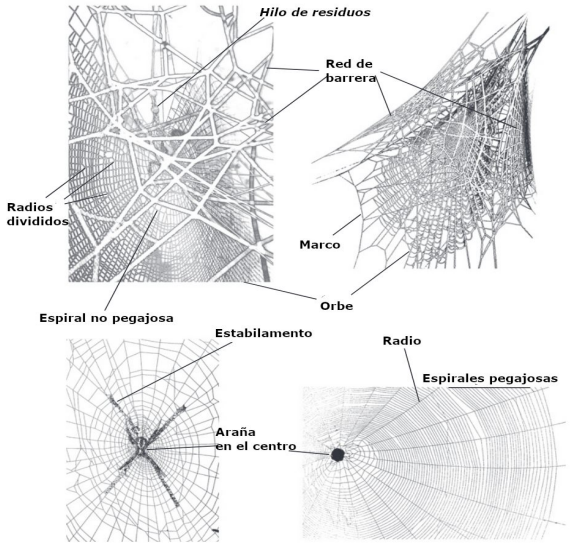


Figura 1.6: Terminología de telarañas (Blackledge et al., 2011, p. 181)

1.2. Objetivos

El procesamiento manual conlleva un esfuerzo significativo. Por ello, se requiere un enfoque automatizado y robusto es decir que permita detectar todas las áreas de interés y sea aceptable para el investigador, además de considerar la forma (variaciones en las mallas de las telarañas y decoraciones del stabilimentum), el tamaño y la estructura (elementos que conforman la telaraña, radio, red de barrera, etc.) de las imágenes de telarañas, así como los finos detalles de sus filamentos. Estos factores influyen en su correcta interpretación, por lo que procesar los bordes, incluso en imágenes ruidosas y de baja calidad, resulta fundamental para su posterior segmentación y así poder extraer información valiosa para su estudio biológico o ecológico.

1.2.1. General

Desarrollar y validar un algoritmo mediante técnicas de visión por computadora y métodos de cómputo evolutivo, sea capaz de procesar imágenes con ruido (variaciones en los niveles de contraste) y extraer características relevantes, específicamente los bordes presentes en la imagen. El algoritmo se diseñará para detectar y segmentar automáticamente las regiones de interés que corresponden al área de la telaraña para diferenciarlas del fondo. La aplicación del algoritmo a una base de datos especializada permitirá cuantificar el daño estructural a través de mediciones precisas de píxeles en imágenes binarizadas.

1.2.2. Específicos

- Explorar los métodos clásicos de detección de bordes en la literatura.
- Analizar los métodos de segmentación de imágenes basada en la detección de bordes.
- Búsqueda de métodos de segmentación global y local, así como métodos de umbralización adaptativos para segmentación de imágenes.
- Revisar el estado del arte con respecto al procesamiento de imágenes con ruido, que incluyan el uso de representaciones multi-nivel para identificar características específicas en cada representación y algoritmos evolutivos.
- Pre-procesar la base de datos de telarañas, esto consiste en eliminar las secciones irrelevantes de las imágenes originales y ajustar la orientación mediante inversión y rotación.
- Desarrollar un algoritmo que combine las áreas mencionadas (Cómputo evolutivo y Visión por computadora) para generar filtros o kernels y aplicar la representación multi-nivel para reducir el ruido y realizar la detección de bordes con convolución.
- Evaluar el algoritmo con la base de datos proporcionada para su segmentación comparando con métodos locales de detección de bordes.
- Comparar los resultados obtenidos del algoritmo propuesto contra los resultados clásicos en la literatura (Canny, Sobel, Roberts, Prewitt) utilizando la medida F-Score para evaluar su desempeño a partir de la base de datos BSDS500.
- Calcular el daño estructural de la base de datos proporcionada, mediante el registro de imágenes, alineando las imágenes originales, contando los píxeles blancos en las imágenes binarizadas y evaluando las diferencias en los píxeles.

1.3. Hipótesis

Se plantea que un algoritmo de detección de bordes que combine técnicas de visión por computadora y cómputo evolutivo mejorará significativamente la precisión en la segmentación de áreas de interés en imágenes ruidosas de telarañas, en comparación con los métodos tradicionales reportados en la literatura. Esto permitirá una cuantificación más precisa del daño estructural, facilitando un análisis más detallado y confiable del comportamiento de las telarañas y su interacción con las presas.

1.4. Justificación

La importancia de desarrollar un algoritmo capaz de segmentar de forma autónoma imágenes con elementos finos y complejos, como las telarañas para una detección precisa de bordes. Esta detección es fundamental para identificar características que son difíciles de localizar con métodos tradicionales en la literatura.

Se proporciona una metodología aplicable a imágenes con estructuras complejas al desarrollar una solución automatizada para poder realizar el cálculo de daño estructural de una manera más eficiente y confiable que los métodos manuales debido a que cuantificación manual del daño estructural de las telarañas a partir de imágenes fotográficas es un proceso laborioso y propenso a errores.

El algoritmo propuesto puede ser aplicado a otros contextos donde la segmentación de imágenes con estructuras finas y ruidosas sea requerida es decir, en aquellas imágenes que presentan variaciones en el contraste con rangos de intensidad reducidos.

Este trabajo surge como una alternativa que propone explorar el uso de cómputo evolutivo y visión por computadora mediante el ajuste y la optimización de parámetros específicos, es un esfuerzo por mejorar la precisión y la calidad en la segmentación de imágenes. Se espera que la integración de estas dos áreas permita desarrollar métodos más eficientes y robustos para el análisis y procesamiento de imágenes, haciendo uso de la capacidad de selección automática que ofrecen las técnicas de programación evolutiva para identificar las soluciones óptimas en comparación con los métodos clásicos descritos en la literatura.

Los resultados de usar este algoritmo pueden proporcionar datos sobre la funcionalidad de las telarañas definiendo dos puntos importantes:

1. Qué tan eficaces son las telarañas para capturar sus presas.

2. Analizar aquellos insectos que evaden o escapan de la telaraña y analizar el daño estructural.

1.5. Organización del documento

Este documento consta de seis capítulos organizados de la siguiente manera: En el capítulo dos se presenta una revisión de la literatura, detallando las cadenas de búsqueda

empleadas y las fuentes de información consultadas para obtener artículos relacionados con la problemática abordada.

El capítulo tres introduce el marco teórico, donde se exponen los fundamentos del procesamiento de imágenes, las técnicas utilizadas en la detección de bordes mediante el Laplaciano de la Gaussiana y la segmentación usando operaciones morfológicas, así como los principios del cómputo evolutivo, con énfasis en la programación evolutiva.

En el capítulo cuatro se describe, paso a paso, el algoritmo propuesto para la detección de bordes y la segmentación, integrando ambas áreas. Además, se exploran dos métodos adicionales como alternativas para abordar la problemática.

En el capítulo cinco se presentan los resultados obtenidos. Para ello, se proporcionaron al experto diversas imágenes previamente segmentadas y se calificó el resultado final mediante un cuestionario fundamentado en su criterio observacional. Además, se emplearon las métricas de precisión, sensibilidad y F-score con el fin de comparar los métodos evaluados, determinando así cuál de ellos se aproxima mejor a los resultados emitidos por el experto.

Finalmente, en el capítulo seis se ofrece una breve discusión sobre el trabajo futuro, destacando observaciones respecto a las propuestas planteadas y las posibles modificaciones que podrían mejorar su rendimiento y calidad.

Capítulo 2

Revisión de la literatura

En este capítulo se presentan los resultados de la búsqueda bibliográfica relacionada a trabajos anteriores en el área de detección de telarañas mediante análisis de imágenes, se muestra una revisión de la literatura para identificar antecedentes sobre el interés en abordar la detección de telarañas. La revisión de la literatura se realizó en bases de datos académicas como IEEE Xplore, Springer Link y ScienceDirect.

- IEEE Xplore: Proporciona publicaciones de ingeniería eléctrica, electrónica y computación.
- Springer Link: Ofrece una colección de libros y artículos científicos en diversas áreas como ciencia, tecnología y medicina.
- ScienceDirect: Ofrece una base de datos de revistas y libros editados por Elsevier en el área científica y médica.

Se construyó la siguiente cadena de búsqueda con términos relevantes para el área de estudio y se agregaron filtros específicos para los resultados. Se realizó en dos fuentes distintas con rangos de fechas de 2018 a 2023. Los detalles se encuentran en el cuadro (2.1).

(“image*” OR “detection*” OR “edge*”) AND (“genetic algorithm*” OR “evolutionary programming*” OR “evolutionary algorithm*” OR “image pyramid*” OR “multiresolution pyramid*” OR “multiscale pyramid*”)

Tabla 2.1: Resultados primera cadena de búsqueda

Fuente	Resultados	Filtros
IEEE Xplore	5,298	Conferencias, Articulos
Springer Link	6,883	Articulos

La siguiente cadena de búsqueda se utilizó exclusivamente en ScienceDirect debido a que no acepta los comodines (*) como criterio de búsqueda. Los detalles se encuentran en el cuadro (2.2).

("image") AND ("edge") AND ("detection") AND ("genetic algorithm" OR "evolutionary programming" OR "evolutionary algorithm" OR "image pyramid" OR "multiresolution pyramid" OR "multiscale pyramid")

Tabla 2.2: Resultados segunda cadena de búsqueda para ScienceDirect

Fuente	Resultados	Filtros
ScienceDirect	5,356	Ninguno

Con base en estas cadenas de búsqueda se eligieron los siguientes artículos basados en la relevancia y resúmenes relacionados con el tema:

- El artículo (Y. Zhang et al., 2017) propone un algoritmo donde combina los operadores Sobel, Canny y LoG el cual está enfocado en la identificación de paquetes de cigarros. Este algoritmo también hace uso de diferentes técnicas para mejorar estos operadores, en el cual extiende la detección de direcciones de los bordes para Sobel, el uso del algoritmo de Otsu para seleccionar el umbral apropiado, así como operaciones morfologías.

- El artículo (Radha. R, 2014) enfoca la detección de bordes a la detección de venas en hojas de plantas utilizando el método de Canny, implementa una representación multi-señal llamada scale-space y usa ajuste de umbrales para detectar correctamente tanto los bordes de las hojas y venas de las plantas.

- El articulo (M. Chen et al., 2023) Aborda una descripción del proceso de detección de bordes y compara varios operadores (Robers, Prewitt, Sobel, LOG, Canny) aplicados en MATLAB para imágenes de caña de azúcar, en el cual concluye que Canny es el operador más completo para esta tarea.

- El articulo (Cui et al., 2021) Propone un método llamado Multiscale Adaptive Edge detector (MAED) en el cual genera una pirámide multi-escala, en cada nivel calcula un mapa de características mediante el uso del gradiente, así como el cálculo de un mapa de desviación estándar.

  Obtiene los posibles bordes mediante la detección pixel a pixel (Pixel-by-pixel detection) basado en estos dos mapas y los bordes son reducidos con supresión no máxima (Nonmaximal suppression), aplica un mecanismo de fusión para los bordes obtenidos mediante votación y finalmente aplica enlace mediante histéresis adaptativa (Adaptive hysteresis linking).

Concluye que su método puede adaptarse rápidamente para la detección de bordes y compara la cantidad de parámetros fijos que usa su método contra SBED (Superpixel-based edge detection)

- El articulo (Shabankareh y Shabankareh, 2019) combina técnicas de detección de bordes, redes neuronales y algoritmos genéticos llamada (Dominated Sorting Genetic Algorithm NSGA-II) como en algunos artículos anteriores, el algoritmo procesa automáticamente un umbral para el procesamiento de la imagen, para la detección utiliza la información completa de la imagen y realiza una clasificación de píxeles como borde o no borde.

  Mediante el algoritmo genético realiza una población inicial con 10 métodos estándar, utilizando cromosomas los operadores son elegidos para aplicarlos en la red.

  La red neuronal consiste en tres capas, la capa de entrada consiste en N neuronas, la segunda capa es una capa oculta de 7 neuronas y la capa de salida que consiste en 2 neuronas, el cual contiene la clasificación de borde o no-borde.

- El articulo (Jain et al., 2016) da una breve introducción sobre las técnicas de detección de bordes las cuales complementa mediante el cómputo suave (Soft computing) en los cuales se encuentran: Computación neuronal, Lógica difusa, Sistema de Inferencia Neuro-Difuso Adaptativo, Algoritmos genéticos.

  Menciona que las técnicas de cómputo suave han sido ampliamente utilizadas para el procesamiento de imágenes, hace una revisión y da detalles de sus características, entre ellas las más notables son: Algoritmos Genéticos (Genetic Algorithms),Particle Swarm Optimization y Ant Colony Optimization.

En la búsqueda se ha encontrado que la mayoria de los artículos de investigación se pueden dividir en dos categorías: la primera son propuestas para mejorar los operadores actuales de detección de bordes y la segunda son propuestas de aplicación de diversas técnicas o métodos de diferentes áreas, las cuales suelen incluir un algoritmo para procesar las imagenes, este trabajo puede incluirse dentro de esta categoría.

Se generó la siguiente consulta de búsqueda avanzada con un rango de fechas de 2010 a 2023 para encontrar artículos y publicaciones científicas enfocado a trabajos relevantes que incluyan investigación enfocada en telarañas. Los detalles se encuentran en el cuadro (2.3).

(“spiderweb*” OR “spider web”) AND (“detection*” OR “image*” OR “evolutive*”)

Tabla 2.3: Resultados primera cadena de búsqueda

Fuente	Resultados	Filtros
IEEE Xplore	35	Ninguno
Springer Link	3,070	Ninguno

En esta búsqueda se eligió el siguiente artículo basado en la relevancia y resúmen relacionado con el tema:

- El artículo (Batista-Plaza, 2017) aborda la tarea de identificación y clasificación de arañas usando una base de datos de diferentes especies y usando su patrón de diseño para clasificarlas, utiliza filtros espaciales para mejoramiento de las imágenes, PCA (Análisis de componentes principales) y para clasificación SVM (Vector Support Machine).

  Este artículo se centra en la clasificación de arañas basadas en la telaraña construida, incluye una sección a la correcta segmentación de la telaraña y aborda que métodos obtienen mejores resultados. Hace una mención de los métodos de detección de bordes y menciona que método de la literatura obtiene mejores resultados para la detección en imágenes de telarañas.

Capítulo 3

Marco teórico

En este capítulo se explora la representación de imágenes digitales y se introducen las nociones y técnicas fundamentales para la detección de bordes, incluyendo métodos basados en primeras y segundas derivadas. Además, se abordan las operaciones morfológicas que permiten manipular la forma de los objetos en una imagen, permitiendo su segmentación. Posteriormente se describe el uso de la convolución y las operaciones de filtrado, herramientas clave para la reducción de ruido y mejora de características en las imágenes. Por último, se presentan técnicas de registro de imágenes para realizar análisis comparativos, así como las métricas utilizadas para evaluar el desempeño de los algoritmos.

3.1. Representación de imágenes digitales

En el tratamiento de imágenes tenemos la definición de pixel, este consiste en un elemento en una posición (f, c) fila y columna en una imagen digital. Un pixel representa el elemento más pequeño de una imagen, este es representado por un cuadrado, la figura (3.1) muestra esta representación en dos dimensiones.

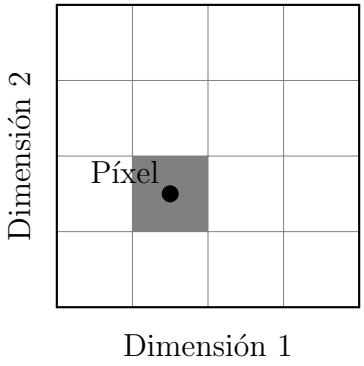


Figura 3.1: Ejemplo de representación de imagen digital.

Usando esta configuración de elementos, los píxeles pueden representar regiones que describen ciertas características dentro de la imagen, esto se logra mediante la representación de valores, como el espacio de colores RGB, escala de grises o una imagen binaria.

3.1.1. Resolución de imágenes

La resolución de imagen se refiere al número de píxeles en una imagen digital, una resolución alta significa mejor calidad de imagen, por ejemplo una resolución en 4K (donde la imagen tiene 3840 píxeles horizontales y 2160 píxeles verticales) es superior a una resolución SD (Standar definition) de 640 píxeles horizontales y 480 píxeles verticales.

3.1.2. Imágenes de 1-bit

Las imágenes de 1-bit consisten en píxeles donde la representación es de un solo bit (0 o 1) esta representación también es conocida como imagen binaria, la figura (3.2) ilustra esta representación donde los píxeles se catalogan en dos regiones (blancos y negros). Este tipo de imágenes también son llamadas monocromáticas debido a la falta de color (Li et al., 2014).

(a) Imagen original (b) Zoom de 900 %

Figura 3.2: Descripciones de una imagen de 1-bit.

3.1.3. Imágenes de 8 bits (Escala de grises)

En una imagen en escala de grises, los píxeles se representan en valores comúnmente dentro de los rangos de (0 a 255) estos valores miden la intensidad de luz en el pixel, donde 0 representa el color negro y 255 el color blanco y entre esos valores la escala de iluminación es variada, la figura (3.3) muestra esta variación.

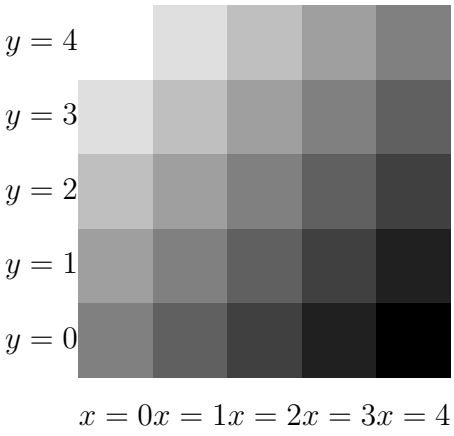


Figura 3.3: Ejemplo de representación de escala de grises.

Usando esta variación una imagen a color puede ser representada utilizando estos valores, la figura (3.4) muestra el resultado de aplicar la conversión de RGB a escala de grises.

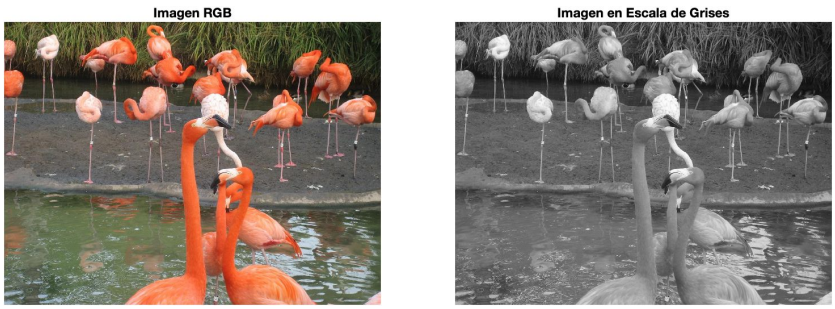


Figura 3.4: Ejemplo de conversión a escala de grises.

3.1.4. Imagen binaria

Las imágenes binarias a menudo son obtenidas extrayendo información de una imagen en escala de grises, estas pueden ser objetos en ciertas posiciones, límites o fronteras de objetos o la presencia o ausencia de propiedades en la imagen. A diferencia de una imagen en escala de grises, una imagen binaria solo un bit es asignado a cada pixel, $B = 1$ por lo que solo existen dos valores posibles 0 y 1. Estos valores usualmente son interpretados como boléanos, ya sea 0 y 1 o su equivalente “verdadero” o “falso”.

Los valores indican la presencia o ausencia de una propiedad asociada a la imagen, 1 indicando la existencia de cierta propiedad y 0 indicando lo contrario. La figura (3.5) muestra esta propiedad asociada a los bits representados con píxeles.

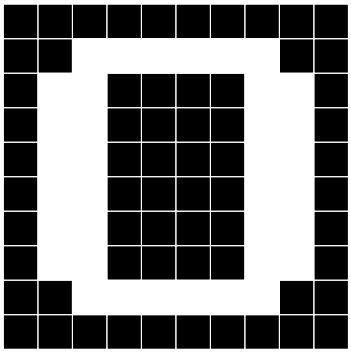
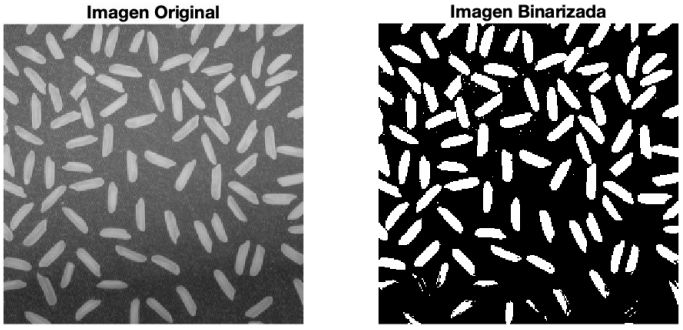


Figura 3.5: Ejemplo de representación del número 0 en una matriz de 10×10.

Usando esta representación binaria, las imágenes en escala de grises pueden ser convertidas a imágenes binarias (generalmente utilizando un umbral) en la figura (3.6) se puede observar este proceso de conversión.



(a) Imagen original (b) Imagen binaria

Figura 3.6: Ejemplo de binarización de una imagen en escala de grises.

3.1.5. Umbralización de imágenes

Una abstracción muy simple es la umbralización de imágenes, suponiendo que una imagen en escala de grises f puede tomar valores $0, 1, 2, 3, ..., K-1$, se define un umbral T que recae en los rangos de la escala de grises: $T \in \{0, 1, 2, \ldots, K-1\}$.

El proceso de umbralizar es una comparación de cada pixel en f con el valor T, basado en esta comparación se define la salida del pixel correspondiente en la imagen binaria.

$$g(n) = \begin{cases} 0 & \text{si } f(n) > T \\ 1 & \text{si } f(n) < T \end{cases} \tag{3.1}$$

El valor de T es crítico dado que controla la información obtenida, por ello diferentes valores de T producen diferentes abstracciones de la imagen resultante. Otros valores de T pueden no proporcionar información importante en el resultado final.

La variación del umbral permite de forma sencilla separar los objetos de interés en el resto de la imagen, para encontrar el valor óptimo generalmente se consigue mediante ensayo y error, aunque este valor también puede variar dependiendo las características presentes en la imagen (Sucar., 2011). En la figura (3.7) se observa los resultados de umbralizar una imagen con diferentes valores de T.

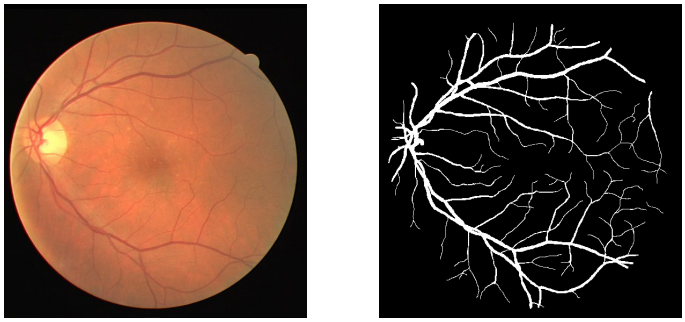
(a) Imagen original (b) Umbral 0.1 (c) Umbral 0.3 (d) Umbral 0.6

Figura 3.7: Resultados de umbralización utilizando imbinarize en MATLAB.

3.1.6. Segmentación

La segmentación es el proceso de dividir una imagen en regiones de interés, el caso más simple es tener dos regiones, un objeto de primer plano y la región de fondo, o pueden existir múltiples objetos de fondo.

Existen varios tipos de regiones, como por ejemplo al segmentar una imagen de la naturaleza, existen regiones del cielo, nubes, el terreno, edificios o árboles. En este caso las diferentes categorías se llaman clases, la figura (3.26) muestra un ejemplo que consiste en segmentación de vasos sanguíneos en imágenes de retina.



(a) Imagen original (b) Segmentación

Figura 3.8: Ejemplo de segmentación de DRIVE (Digital Retinal Images for Vessel Extraction) (ver Challenge, s.f.).

Como se explicó en la sección anterior (ver 3.1.5) existen umbrales fijos que permiten realizar una segmentación de una forma muy sencilla, la forma más común y confiable de encontrar un umbral es la selección manual. Las características más comunes para delimitar o segmentar regiones son: intensidad de los píxeles, textura, color, gradiente y profundidad

relativa. Existen varias técnicas de segmentación de regiones, estas se clasifican en tres tipos:

- **Locales**: Se basan en agrupar píxeles con base en sus atributos y los de sus vecinos (agrupamiento).
- **Globales**: Se basan en las propiedades globales de la imagen (división).
- **División y agrupamiento (Split & merge)**: Combinan técnicas locales y globales.

3.1.7. Ruido en imágenes

Las imágenes a menudo son degradadas por errores aleatorios, a esta degradación se le llama ruido. El ruido puede ocurrir durante la captura de la imagen, transmisión o procesamiento, el ruido suele describirse por sus características probabilísticas y a menudo se utiliza un tipo idealizado de ruido llamado ruido blanco. Un tipo especial de ruido blanco es el ruido Gaussiano, que consiste en una variable con una distribución Gaussiana (normal) con una función de probabilidad dada por la curva Gaussiana, en 1D la función de densidad se da por:

$$p(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{(x-\mu)^2}{2\sigma^2}} \tag{3.2}$$

Donde: μ es la media y σ la desviacion estandar de una variable aleatoria. El ruido Gaussiano es una buena aproximación al ruido que ocurre en la mayoría de los casos. La figura (3.9) muestra el efecto de aplicar ruido blanco a una imagen, este ruido se modela como variaciones aleatorias en los valores de los píxeles, donde cada variación está distribuida de manera normal alrededor del valor verdadero del píxel.

(a) Imagen original (b) Imagen con ruido blanco

Figura 3.9: Ejemplo de ruido blanco con media de 0.05 y varianza de 0.03.

3.1.8. Representación multi-escala

Consiste en la representación de una imagen en múltiples resoluciones (Adelson et al., 1984) el resultado es un conjunto de imágenes con resoluciones reducidas agrupadas en forma de pirámide, conforme vamos subiendo de nivel tanto el tamaño y resolución disminuyen. El nivel base J es de tamaño $2^J \times 2^J$ o $N \times N$, donde $J = \log_2 N$. El nivel 0 es de tamaño 1×1, y el nivel general es de tamaño $2^j \times 2^j$ donde $0 \leq j \leq J$

El total de píxeles en un nivel $P + 1$ para $P > 0$ es:

$$N^2 \left(1 + \frac{1}{4^1} + \frac{1}{4^2} + \ldots + \frac{1}{4^P}\right) \leq \frac{4}{3}N^2 \tag{3.3}$$

La figura (3.10) muestra estos cambios de resolución iniciando con una imagen de 256 $\times$ 256 píxeles.

Figura 3.10: La resolución de la imagen se reduce mediante un factor de 2, generando así distintos niveles de representación.

3.2. Detección de bordes

Al analizar y procesar imágenes digitales la detección de bordes es un paso principal si queremos descubrir características, el objetivo principal es detectar cambios sutiles (discontinuidades) en las imágenes y en algunos casos es necesario realizar pre-procesamiento para poder resaltar mejor estos detalles. Estas discontinuidades pueden representarse de la siguiente forma ilustradas en la figura (3.11) y para poder detectar estos cambios existen dos clases: basadas en gradientes (S. Chen y Yang, 2021) y segundas derivadas (M. Chen et al., 2023).

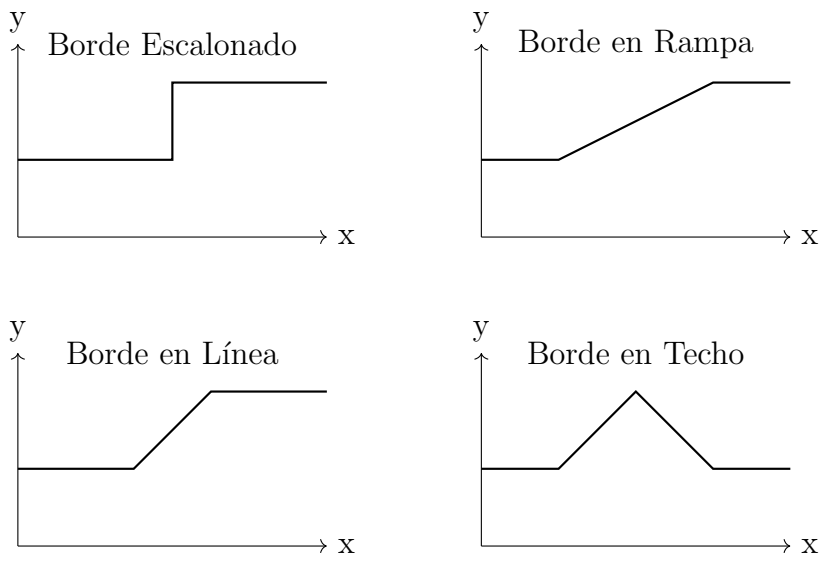


Figura 3.11: Tipos de bordes.

Estas discontinuidades pueden ser representadas en una función con un valor de gradiente o derivada alta, como se puede observar en la figura (3.12).

(a) (b)

Figura 3.12: Ejemplo de discontinuidad, la figura (a) muestra una discontinuidad en intensidad entre la parte izquierda y derecha, la figura (b) muestra el corte horizontal que corresponde a la discontinuidad.

La detección de bordes es una forma frecuentemente usada para segmentar imágenes basada en los cambios abruptos de intensidad. Actualmente existen diferentes operadores para la detección de bordes, entre los que se incluyen: Roberts, Sobel, Prewitt, Canny y Laplace of Gaussian (LoG) (Jain et al., 2016).

3.2.1. Operadores de gradiente (Primeras derivadas)

El operador gradiente es usado para extraer bordes en rango e intensidad en las imágenes, basada en diferenciar la imagen, es decir, encontrar la derivada con respecto a los

ejes x y y o gradiente (Sucar., 2011). Un gradiente puede ser representado en 2D de la siguiente forma:

$$\text{Vector gradiente:} \quad \nabla f = \begin{bmatrix} G_x(i,j) \\ G_y(i,j) \end{bmatrix} = \begin{bmatrix} \frac{\partial f(i,j)}{\partial x} \\ \frac{\partial f(i,j)}{\partial y} \end{bmatrix} \tag{3.4}$$

$$\text{Magnitud del gradiente:} \quad |\nabla f| = \sqrt{G_x(i,j)^2 + G_y(i,j)^2} \tag{3.5}$$

$$\text{Dirección del gradiente:} \quad \theta = \arctan\left(\frac{G_y(i,j)}{G_x(i,j)}\right) \tag{3.6}$$

La magnitud del gradiente (3.5) es igual al pico máximo por unidad de distancia en la dirección del gradiente (3.6). Así, si hay una discontinuidad en la imagen, la primera derivada da lugar al pico de respuesta, que es una indicación de un borde (Abdulghafour, 2003). El operador clásico Sobel hace uso de una plantilla en una imagen $f(i,j)$ para aproximar el valor absoluto de la magnitud del gradiente en una imagen en escala de grises, la figura (3.13) muestra estas plantillas que son utilizadas con convolución para detectar los bordes.

G_x

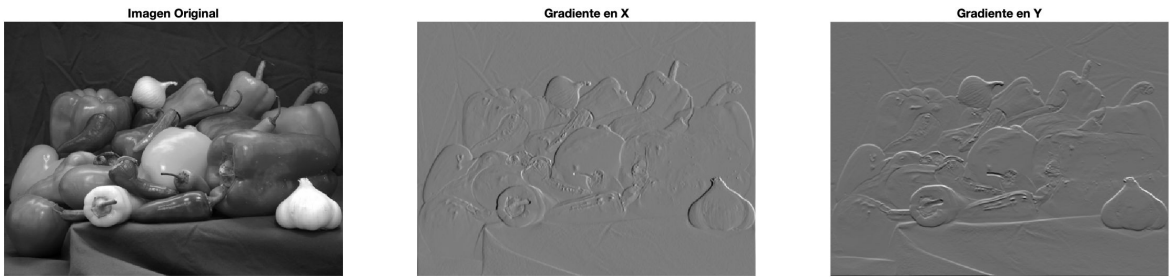
-1	0	1
-2	0	2
-1	0	1

G_y

1	2	1
0	0	0
-1	-2	-1

Figura 3.13: Aproximación para calcular el gradiente en los componentes G_x y G_y

El operador Sobel es utilizado por su simplicidad y buenos resultados, en la figura (3.14) se muestra los gradientes de la imagen en las direcciones x y y.



(a) Imagen original (b) Gradiente en X (c) Gradiente en Y

Figura 3.14: Resultado de aplicar los filtros con convolución en una imagen, estos muestran los cambios de los gradientes en las direcciones individuales (x, y)

Los resultados de estas aproximaciones son combinados para obtener la magnitud del gradiente usando la ecuacion (3.5) el resultado se observa en la figura (3.15) que muestra los bordes en la imagen.

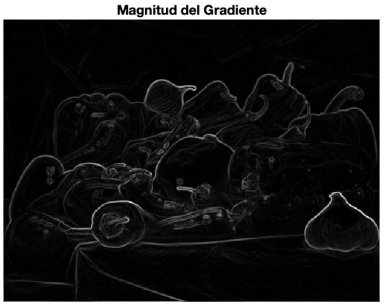


Figura 3.15: Imagen resultante de aplicar el operador Sobel.

3.2.2. Operador Laplace of Gaussian (Segundas derivadas)

Este operador aprovecha el filtro Gaussiano tradicional. Primero se realizan operaciones de reducción de ruido y suavizado para finalmente obtener los bordes mediante el diferencial de segundas derivadas para generar cruces por cero (M. Chen et al., 2023).

También llamado detector de bordes de Marr-Hildreth (Marr y Hildreth, 1980) debido a dos propiedades principales descritas por los autores que un operador detector de bordes debe tener: El operador debe ser capaz de calcular una aproximación de la primera o segunda derivada en cada punto de la imagen. El operador también debe ser capaz de ajustarse en cualquier escala deseada, de modo que puedan utilizarse operadores grandes para detectar bordes borrosos y operadores pequeños para detectar detalles finos, Marr y Hildreth argumentan que el operador Laplace of Gaussian cumple estas condiciones.

El algoritmo consiste en dos pasos clave, el primero consiste en explorar los cambios de intensidades usando un filtro especificado en una escala que es obtenido mediante la segunda derivada de la función Gaussiana para finalmente detectar los bordes usando el cruce por cero de la expresión (3.7) para una imagen $I(x, y)$.

$$\nabla^2 G(x, y) * I(x, y) \tag{3.7}$$

donde $G(x, y)$ representa una distribución Gaussiana en dos dimensiones, $(*)$ representa la operación de convolución y ∇^2 es el Laplaciano. La función Gaussiana en 2D (también llamado filtro Gaussiano o simplemente Gaussiano) está dado por:

$$G(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.8}$$

donde x, y son coordenadas en la imagen, σ (sigma) es la desviación estándar en ocasiones se utilizan las siguientes expresiones:

$$G(x, y) = \frac{e^{-\frac{x^2+y^2}{2\sigma^2}}}{2\pi\sigma^2} \quad \text{o} \quad G(x, y) = \frac{e^{-\frac{x^2+y^2}{2\sigma^2}}}{\sqrt{2\pi\sigma}} \tag{3.9}$$

Tomando en cuenta la ecuación (3.8) el objetivo es obtener la segunda derivada de la función en 2D. La figura 3.16 muestra esta representación usando un valor de $\sigma = 1$.

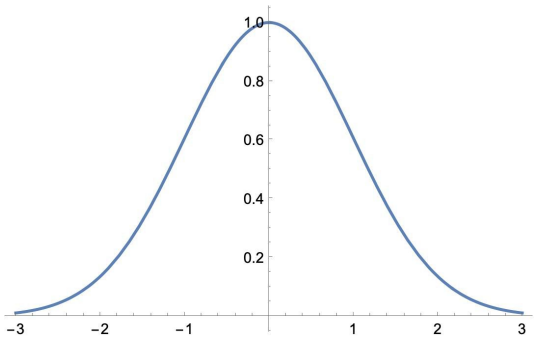


Figura 3.16: El valor de σ controla el ancho de la curva Gaussiana. Un valor mayor de σ hace que la curva sea más ancha, mientras que un valor menor de σ hace que la curva sea más estrecha.

La segunda derivada parcial con respecto a x:

$$\frac{\partial^2 G}{\partial x^2} = \frac{(x^2 - \sigma^2)e^{-\frac{x^2+y^2}{2\sigma^2}}}{\sigma^4} \tag{3.10}$$

la segunda derivada parcial con respecto a y:

$$\frac{\partial^2 G}{\partial y^2} = \frac{(y^2 - \sigma^2)e^{-\frac{x^2+y^2}{2\sigma^2}}}{\sigma^4} \tag{3.11}$$

El operador de Laplace ∇^2 da como resultado la segunda derivada y es no direccional (isotrópico) (Milan Sonka, 2014). Si tenemos el Laplaciano de una imagen $I(x, y)$ suavizado por una función Gaussiana (expresada usando convolución $*$) esta operación es comúnmente abreviada LoG (Laplace of Gaussian) representado por:

$$\nabla^2[G(x, y) * I(x, y)] \tag{3.12}$$

El orden de diferenciación y convolución puede intercambiarse debido a la linealidad de los operadores obteniendo:

$$[\nabla^2 G(x, y) * I(x, y)] \tag{3.13}$$

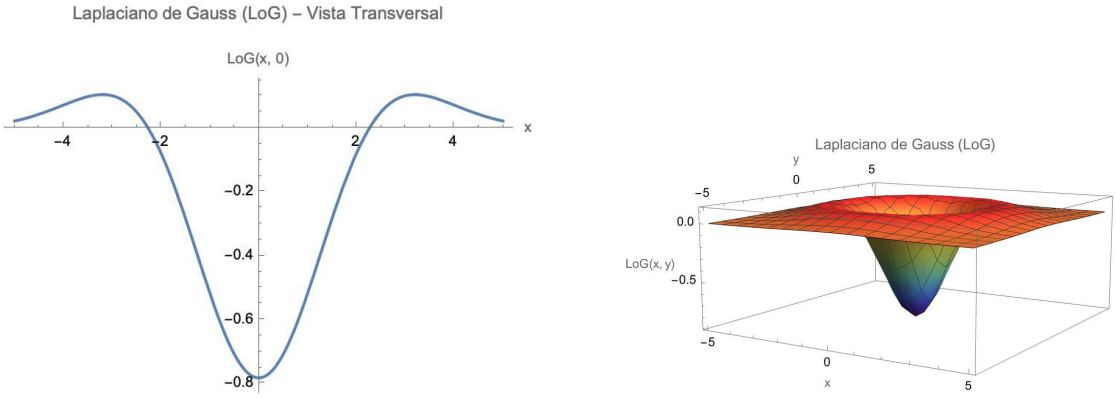
donde:

$$\nabla^2 G(x, y) = \frac{\partial^2 G(x, y)}{\partial x^2} + \frac{\partial^2 G(x, y)}{\partial y^2} \tag{3.14}$$

Sustituyendo obtenemos el siguiente resultado:

$$\text{LoG}(x, y, \sigma) = c \cdot (\frac{x^2 + y^2 - 2\sigma^2}{\sigma^4}) \cdot e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{3.15}$$

La constante c normaliza la suma de la máscara de sus elementos a cero. Debido a su forma invertida, este operador es comúnmente llamado Mexican hat, la figura (3.17) muestra la forma gráfica en 2D y en 3D.



(a) Vista en 2D (b) Vista en 3D

Figura 3.17: Forma invertida del operador LoG.

Similar a las primeras derivadas (3.2.1) las segundas derivadas pueden ser estimadas con un conjunto de filtros lineales (Burger y Burge, 2016), por ejemplo la plantilla horizontal y vertical se pueden representar usando.

$$\frac{\partial^2 f}{\partial^2 x} \approx H_x^L = \begin{bmatrix} 1 & -2 & 1 \end{bmatrix} \quad \frac{\partial^2 f}{\partial^2 y} \approx H_y^L = \begin{bmatrix} 1 \\ -2 \\ 1 \end{bmatrix} \tag{3.16}$$

Al combinar las segundas derivadas en el eje x y el eje y obtenemos un filtro Laplaciano en 2D esto se muestra en la ecuación (3.17).

$$H_L = H_x^L + H_y^L = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{3.17}$$

El efecto de aplicar un filtro Laplaciano puede apreciarse en la figura 3.18 asignando un valor de $\sigma = 1$ y con un tamaño de 7×7 usando la siguiente ecuación:

$$\mathrm{k}(i,j) = LoG(i - \frac{n-1}{2}, j - \frac{m-1}{2}, \sigma) \tag{3.18}$$

donde i, j representan posiciones de fila y columna del kernel y n, m el tamaño del kernel en filas y columnas.

Debido que los resultados contienen números negativos y valores decimales, la imagen fue normalizada con rangos de 0 a 255 para su visualización. Esta operación puede realizarse con la siguiente ecuación:

$$P_{out} = (P_{\text{in}} - C)\left(\frac{b-a}{d-c}\right) + a \tag{3.19}$$

donde a, b corresponden a los límites de representación en escala de grises con valores de 0 a 255 y c, d el valor mínimo y máximo respectivamente presentes en la imagen, P_{in} es el pixel de entrada y P_{out} el pixel de salida.

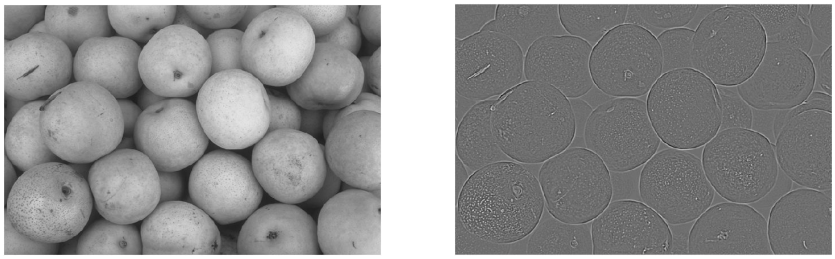


Figura 3.18: Aplicación del filtro por la ecuación (3.18), la imagen a la izquierda se encuentra en escala de grises y la imagen de la derecha es el resultado de aplicar el filtro LoG.

3.2.3. Cruce por cero

Los cruces por cero permiten detectar cambios abruptos en una función. El uso de un operador basado en las segundas derivadas ofrece la ventaja de localizar los bordes con mayor precisión (Sucar., 2011). Una técnica para identificar estos cruces consiste en definir una vecindad local de tamaño 3×3 alrededor de un punto p al procesar una imagen en escala de grises $I(x, y)$ mediante la convolución con el operador LoG. La figura (3.19) muestra esta vecindad centrada en el punto $p(x, y)$. Dentro de esta vecindad, se busca la presencia de dos píxeles con signos opuestos (Rafael C. Gonzales, 2002), lo cual puede ocurrir en cuatro direcciones: izquierda-derecha, arriba-abajo, y en ambas diagonales.

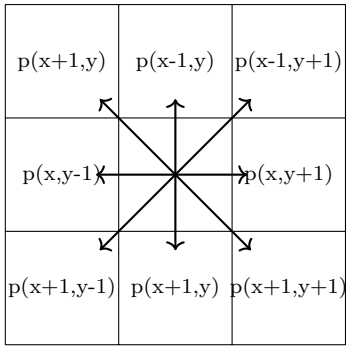


Figura 3.19: Al tener la imagen filtrada I usando LoG el cruce por cero es utilizado para buscar cambios de signo en al menos dos píxeles dentro de una vecindad.

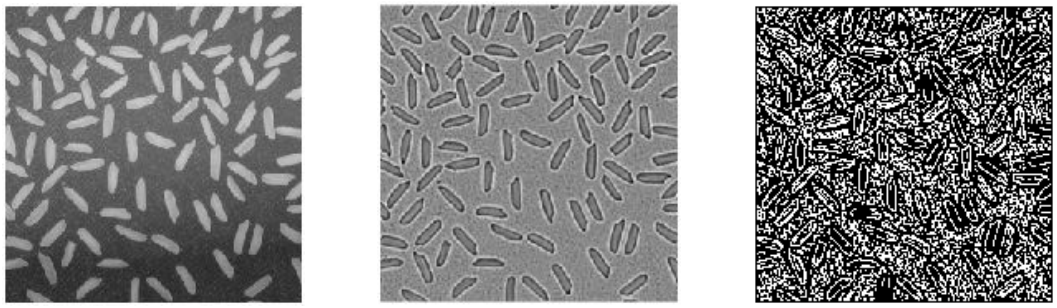
El algoritmo (1) ilustra el proceso de búsqueda de píxeles y sus correspondientes cambios de signo mediante la iteración de la imagen.

Algoritmo 1 Detección de Cruce por Cero

Requiere: I, Imagen filtrada con LoG
Requiere: T, Umbral para la detección de cambios significativos
Regresa: Z, Imagen binaria indicando las posiciones de los cruces por cero

```
function ZERO_CROSS_DETECTION(I, T)
    [rows, columns] ← size(I)
    Z ← zeros(rows, columns)
    for i ← 1 to rows − 1 do
        for j ← 1 to columns − 1 do
            if I[i, j] < 0 and I[i, j + 1] > 0 and |I[i, j] − I[i, j + 1]| > T then
                Z[i, j] ← 1
            end if
            if I[i, j] < 0 and I[i, j − 1] > 0 and |I[i, j] − I[i, j − 1]| > T then
                Z[i, j] ← 1
            end if
            if I[i, j] < 0 and I[i + 1, j] > 0 and |I[i, j] − I[i + 1, j]| > T then
                Z[i, j] ← 1
            end if
            if I[i, j] < 0 and I[i − 1, j] > 0 and |I[i, j] − I[i − 1, j]| > T then
                Z[i, j] ← 1
            end if
            if I[i, j] < 0 and I[i − 1, j − 1] > 0 and |I[i, j] − I[i − 1, j − 1]| > T then
                Z[i, j] ← 1
            end if
            if I[i, j] < 0 and I[i − 1, j + 1] > 0 and |I[i, j] − I[i − 1, j + 1]| > T then
                Z[i, j] ← 1
            end if
            if I[i, j] < 0 and I[i + 1, j − 1] > 0 and |I[i, j] − I[i + 1, j − 1]| > T then
                Z[i, j] ← 1
            end if
            if I[i, j] < 0 and I[i + 1, j + 1] > 0 and |I[i, j] − I[i + 1, j + 1]| > T then
                Z[i, j] ← 1
            end if
        end for
    end for
    return Z
end function
```

Cuando se utiliza un umbral no solo los signos dentro de la vecindad deben ser opuestos, el valor absoluto de la diferencia numérica debe exceder el valor del umbral. La figura (3.20) muestra el proceso completo para detectar los bordes en una imagen en escala de grises, un detalle a resaltar es que en la imagen resultante se observan "ciclos" en los bordes, este efecto es una seria desventaja del uso de este método y se puede considerar como información irrelevante en la búsqueda de características en la imagen, que se puede evitar definiendo un valor de umbral positivo. Si existe un cambio de signo en el pixel adyacente dentro de la vecindad y cumple con la condición del valor absoluto de la diferencia numérica excede el umbral especificado, entonces el resultado en la posición marcada en el pixel es 1 en caso contrario el valor asignado es 0.



(a) Imagen original (b) Operador LoG (c) Cruce por cero

Figura 3.20: Después de aplicar un filtro LoG con valor de $\sigma = 1$ y de tamaño 7×7 en una imagen I en escala de grises, el cruce por cero es utilizado para detectar los bordes en la imagen filtrada I haciendo uso también de un umbral manual $T = 10$, Nótese las características irrelevantes presentes en la imagen final.

En la figura (3.21) se pueden observar las variaciones en el resultado de aplicar cruce por cero con diferentes valores de σ y umbrales.

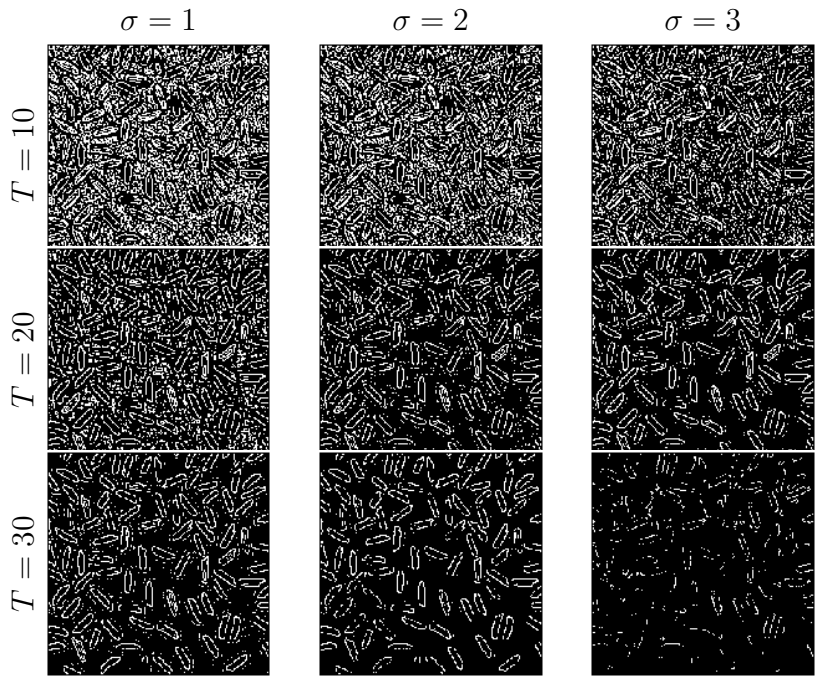


Figura 3.21: Umbrales T con valores bajos resulta en demasiados detalles en la imagen final que pueden no ser muy útiles para la extracción de características.

3.3. Operaciones morfológicas

Las operaciones morfológicas son una herramienta para la extracción de componentes en la imagen que son útiles para la representación de regiones. La morfología hace uso de teoría de conjuntos.

En imágenes binarias el conjunto se encuentra en una representación en 2-D en el espacio de enteros Z^2, donde cada elemento del set es una tupla (Vector 2-D) donde las coordenadas son las coordenadas de un objeto. Las Imágenes en escalas de grises pueden ser representadas como conjuntos donde sus componentes están en Z^3. En este caso dos componentes de cada elemento del conjunto representa las coordenadas del pixel, y el tercero corresponde los valores de intensidad.

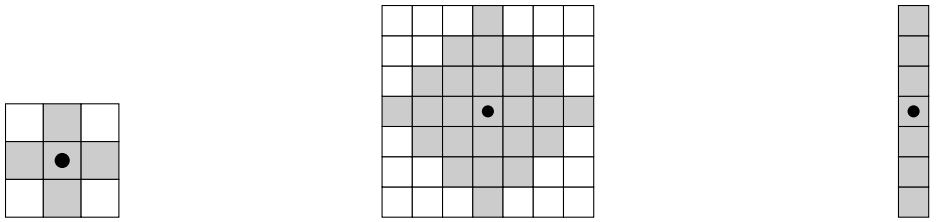
Dentro de las operaciones morfológicas se utilizan dos elementos: píxeles de primer plano y elementos estructurantes, los elementos estructurantes pueden especificarse en términos de píxeles tanto de primer plano como de fondo.

Continuación se presentan algunas definiciones:

- Píxeles de primer plano: En una imagen binaria se representan como el valor 1; al observar la imagen, los píxeles de primer plano aparecen en blanco.
- Píxeles de segundo plano: En una imagen binaria se representan como el valor 0; al observar la imagen, los píxeles de segundo plano aparecen en negro.
- Vecindad: Un conjunto de píxeles definidos por la localización relativa a los píxeles

de interés.

- Morfología: Conjunto de operaciones de procesamiento de imágenes que procesan imágenes basadas en formas.
- Elemento estructurante: una matriz usada para definir la forma y tamaño de la vecindad para aplicar operaciones morfológicas, incluyendo dilatación y erosión.



(a) (b) (c)

Figura 3.22: Ejemplos de elementos estructurantes, los puntos marcan el centro

La figura 3.22 muestra varios ejemplos de elementos estructurantes donde cada cuadrado sombreado representa un miembro que pertenece al elemento estructurante.

Dadas estas definiciones básicas, los conceptos de reflexión y traslación son ampliamente usados en morfología, la reflexión de un set B, se expresa $\hat{B}$ y se define como:

$$\hat{B} = \{w \mid w = -b, \text{ para } b \in B\} \tag{3.20}$$

Si B es el conjunto de píxeles en 2-D representando a un objeto en una imagen, entonces B es el conjunto de puntos donde las coordenadas (x, y) se reemplazan por $(-x, -y)$.

La reflexión consiste en rotar el elemento estructurante (SE) en 180° sobre su origen.

(a) (b)

Figura 3.23: Ejemplo de reflexión, los puntos marcan el origen

Por su parte, la traslación de un conjunto B por los puntos $z = (z_1, z_2)$, denotado $(B)_z$ se define como:

$$B_z = \{c \mid c = b + z, \text{ para } b \in B\} \tag{3.21}$$

Si B es el conjunto de píxeles que representan a un objeto en una imagen, entonces $(B)_z$ es el set de puntos en B donde las coordenadas (x, y) se reemplazan por $(x + z_1, y + z_2)$.

Tanto la reflexión y traslación se definen con respecto al origen de B.

Original B Desplazado $(B)_z$

(a) (b)

Figura 3.24: Ejemplo de traslación dado por z

3.3.1. Dilatación

Con los sets A y B en Z^2, la dilatación de A dado por B, se describe como $A \oplus B$ y se define como:

$$A \oplus B = \{z \mid (\hat{B}_z) \cap A \neq \varnothing\} \tag{3.22}$$

Esta ecuación se basa en reflejar B sobre su origen, y desplazando esta reflexión por z. La dilatación de A y B entonces es el conjunto de todos los desplazamientos en z, donde $\hat{B}$ y A se superponen en al menos un elemento. La ecuación 3.22 puede ser reescrita como:

$$A \oplus B = \{z \mid [(\hat{B}_z) \cap A] \subseteq A\} \tag{3.23}$$

Donde se asume que B es un elemento estructurante y A es el conjunto de objetos a dilatar. La operación dilatación "crece" o "engrosa" objetos en una imagen binaria.

Como ejemplo de la operación dilatación, se aplica un elemento estructurante que se muestra en la figura (3.25) en forma de disco de radio ($r = 2$) a una imagen binaria que representa texto y el resultado de aplicar dilatación se muestra en la figura (3.26).

$$\begin{bmatrix} 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 1 & 1 & 0 \\ 1 & 1 & [\mathbf{1}] & 1 & 1 \\ 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 1 & 0 & 0 \end{bmatrix}$$

Figura 3.25: Elemento estructurante (SE) con forma de disco.

(a) Antes (b) Despues

Figura 3.26: Resultado de aplicar dilatación a una imagen binaria

3.4. Convolución

El concepto de convolución consiste en combinar la intensidad de varios píxeles para obtener la intensidad resultante en un único píxel. Este proceso, que también se conoce como operador local, se implementa en su forma más simple mediante un filtro cuadrado de 3 x 3. Dicho filtro calcula los valores de los píxeles comprendidos en una región definida, denominada ventana, la cual puede tener forma cuadrada o adoptar otra configuración.

La convolución está definida mediante un kernel w de tamaño $m \times n$ y una imagen $f(x, y)$, siendo $w \star f(x, y)$ con la ecuación:

$$(w \star f)(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s, t) f(x - s, y - t) \tag{3.24}$$

Donde un filtro espacial de 3×3 se define como:

w(-1, -1)	w(-1, 0)	w(-1, 1)
w(0, -1)	w(0, 0)	w(0, 1)
w(1, -1)	w(1, 0)	w(1, 1)

Figura 3.27: Filtro espacial $w(s, t)$

El centro del kernel se encuentra en $w(0, 0)$, el kernel debe ser rotado 180 grados antes de realizar la suma de productos. Hay que recordar que debido que la convolución tiene la propiedad conmutativa, tanto w o f pueden ser rotados. Pero por convención el kernel es el elegido para rotación.

1	2	3
4	5	6
7	8	9

Entrada (a)

$*$

1	1
0	1

Kernel (b)

$=$

10	13
19	22

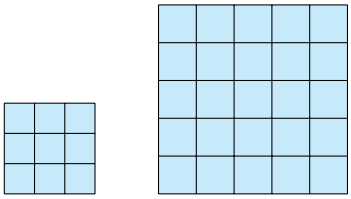
Salida (c)

$$(1 \times 1) + (0 \times 2) + (1 \times 4) + (1 \times 5) = 10$$
$$(1 \times 2) + (0 \times 3) + (1 \times 5) + (1 \times 6) = 13$$
$$(1 \times 4) + (0 \times 5) + (1 \times 7) + (1 \times 8) = 19$$
$$(1 \times 5) + (0 \times 6) + (1 \times 8) + (1 \times 9) = 22$$

Figura 3.28: Ejemplo de convolución con un kernel 2×2

3.4.1. Tamaño del kernel

El tamaño del kernel se refiere a las dimensiones del filtro para realizar la convolución, dentro de una CNN (ver 4.8.1) es uno de los hiperparámetros que se configuran cuando se construye una capa de convolución.



(a) 3x3 (b) 5x5

Figura 3.29: Ejemplo de tamaños de kernels.

3.4.2. Strides

Se refiere a la cantidad por la cual el filtro se desplaza sobre la imagen, es decir un (stride) de uno se desplaza 1 pixel sobre la imagen, para realizar un salto de dos (strides), se asigna un valor de 2.

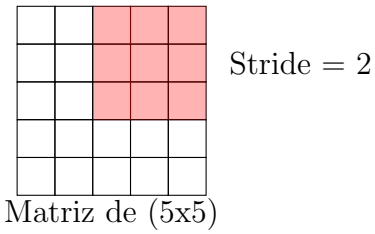


Figura 3.30: Ejemplo de *stride* en una matriz.

3.4.3. Zero-padding

Llamado zero-padding debido a que se agregan ceros al borde de una imagen, el padding es usado comúnmente para preservar el tamaño espacial del volumen de entrada, de modo que el alto y el ancho de la salida sean los mismos.

Padding = 2

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	194	152	32	18	185	0	0
0	0	9	42	51	44	138	0	0
0	0	146	139	172	200	234	0	0
0	0	125	47	78	153	39	0	0
0	0	236	127	150	243	2	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

Figura 3.31: Ejemplo de padding en una matriz.

3.4.4. Padding replicado

Los valores de los píxeles en el borde del marco activo se repiten para crear filas y columnas rellenando los espacios.

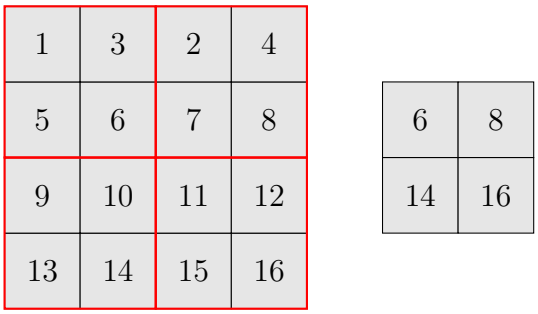
10	10	10	30	60
10	10	10	30	60
10	10	10	30	60
70	70	70	20	40

Figura 3.32: Ejemplo de padding replicado en una matriz, el contenido dentro del recuadro en rojo representa el marco activo, los valores fuera del recuadro representan el relleno de espacios.

3.4.5. Pooling

Al crear capas de convolución para las CNN se incrementan el número de parámetros que la red debe optimizar (aprender), la operación pooling redimensiona los datos de entrada al aplicar una operación matemática como lo es el máximo o el promedio de los valores para reducir la cantidad de parámetros que se pasan de una capa a otra.

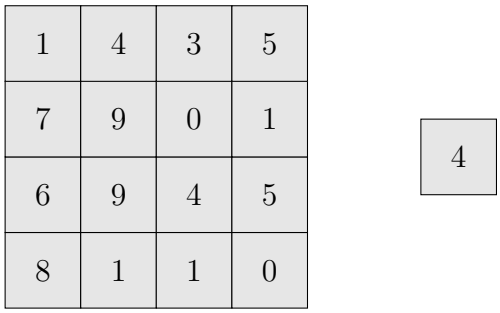
Para definir un **max pooling** podemos verlo de forma similar a un kernel convolucional, un max pooling es una ventana de cierto tamaño y un valor de stride que se desliza sobre la imagen, seleccionando el valor máximo de un pixel.



Matriz de entrada 4×4(Max Pooling 2×2)

Figura 3.33: Ejemplo de un max pooling de 2×2 con stride de 2 que reduce un mapa de características de 4×4 a 2×2.

Por otro lado, existe otro tipo extremo de reducción de dimensionalidad, en lugar de asignar un tamaño de ventana y strides, el **pooling de promedio global** calcula el promedio del valor de todos los píxeles en el mapa de características.

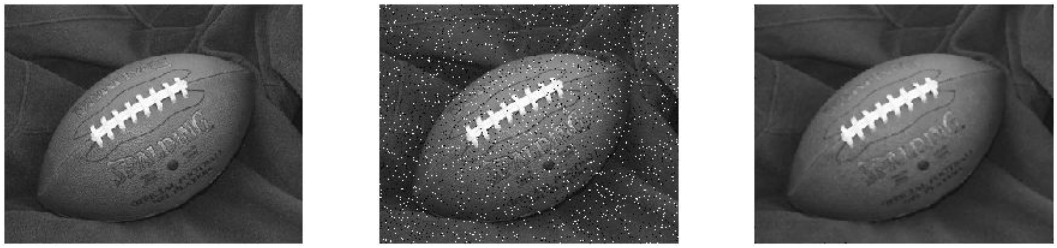


Matriz de entrada 4x4 (Pooling de promedio global)

Figura 3.34: Ejemplo de un Pooling de promedio global que reduce un mapa de características de 4×4 a un solo valor.

3.5. Filtro de mediana

El filtrado de mediana en imágenes es útil para reducir el ruido de sal y pimienta y preservar los bordes en la imagen, este ruido de sal y pimienta ocurre debido a cambios aleatorios. El filtro de mediana consiste en una ventana deslizante a través de una imagen y el valor de la mediana de la intensidad dentro de los valores de la ventana corresponde al valor de salida del pixel procesado. Este filtro preserva las discontinuidades en una función y puede suavizar algunos píxeles que difieren significativamente de los píxeles vecinos sin afectar a los demás píxeles en la imagen (Lim, 1990). La figura (3.35) muestra el resultado de aplicar el filtro de media en una imagen con ruido de sal y pimienta.



(a) Imagen original (b) ruido sal y pimienta (c) filtro de mediana aplicado

Figura 3.35: La imagen original (a) en escala de grises, la imagen en (b) corresponde al resultado de aplicar ruido de sal y pimienta con densidad de 0.05 y en (c) el resultado de aplicar filtro de mediana con ventana de 3×3.

3.6. Registro de imágenes

El registro de imágenes consiste en alinear dos o más imágenes de la misma escena tomando una imagen de entrada y una de salida pero la transformación requerida que produce la imagen de salida es desconocida. El problema consiste en estimar la función de transformación para registrar las dos imágenes (Rafael C. Gonzales, 2002).

Las imágenes pueden ser tomadas por un instrumento en diferentes momentos como las imágenes satelitales en una locación específica tomadas en intervalos de días, meses o incluso años. Para poder realizar un análisis comparativo entre ellas se requiere realizar una compensación geométrica causada por los ángulos, distancia, orientaciones y movimiento de los objetos. Una de las aproximaciones para resolver este problema es el uso de puntos de control donde los puntos de las locaciones y referencias de las imágenes son conocidas. La imagen (3.36) muestra la selección de puntos de control para su registro.

Figura 3.36: La primera imagen se encuentra distorsionada con respecto a la segunda, los puntos de control son usados para realizar una transformación geométrica que alinea las imágenes.

Los resultados del registro son observados en la figura (3.37) que coinciden con el tamaño y la posición de la segunda imagen.

Figura 3.37: Al superponer la imagen original y la transformación podemos ver que las imágenes se encuentran alineadas una de la otra.

3.6.1. Registro de imágenes basado en intensidad

En el registro de imágenes médicas, usando diferentes modalidades (CT, MR, PET) la obtención de información del estado de salud de los pacientes requiere combinar imágenes en diferentes resoluciones, posiciones u orientación de objetos, utilizando un sistema de coordenadas mediante un conjunto de imágenes de referencia estáticas.

La técnica del registro de imágenes consiste en un proceso automatizado utilizando registro rígido mediante la maximización de información mutua. El registro rígido consiste

en traslaciones y rotaciones. La maximización de información mutua consiste en la transformación del sistema de coordenadas de una imagen a otra de manera que se maximice la información mutua entre ambas imágenes (Raghunathan et al., s.f.).

El registro consiste en obtener la intensidad de dos imágenes $T1$ y $T2$ y considerarlas como variables aleatorias, la información mutua (MI) mide cuanta información de una variable aleatoria tiene sobre otra. La información mutua es calculada basada en la entropía de cada variable aleatoria, la entropía de la variable X es dada por la ecuación (3.25):

$$H(X) = -\int p_X(x) \log p_X(x)\, dx \tag{3.25}$$

La entropía de dos variables aleatorias, X y Y está dada por (3.26):

$$H(X,Y) = -\int p_{AB}(a,b) \log p_{AB}(a,b)\, da\, db \tag{3.26}$$

Si X y Y son variables aleatorias definiendo las intensidades de dos imágenes en un conjunto, entonces la información mutua de dos imágenes se define como (3.27):

$$I(X,Y) = H(X) + H(Y) - H(X,Y) \tag{3.27}$$

La probabilidad marginal y la conjunta de las intensidades de la imagen deben ser estimadas en la información de la imagen, esto se realiza mediante una ventana Parzen. En este método se toman muestras S de las imágenes y un kernel de superposición $K(\cdot)$ es centrado en los elementos de S. La función usada como un filtro de suavizado debe tener media de 0 y el valor de su integral debe ser 1. Si N es el número de ejemplos, la estimación de la variable aleatoria X está dada por la ecuación (3.28):

$$P(x) = \frac{1}{N}\left(\sum_{j \in N} K(x - s_j)\right) \tag{3.28}$$

El algoritmo de información mutua de Mattes (Mattes et al., 2001) es utilizado para el registro de la imagen, un conjunto de muestras de la imagen es extraído de la imagen, la probabilidad marginal y la función de densidad de probabilidad (PDF) es evaluada en posiciones discretas. La entropía es calculada sumando los intervalos, un kernel B-spline de orden cero es usado para calcular el PDF de la imagen estática, mientras que un kernel B-spline de tercer orden es utilizado para calcular el PDF de la imagen en movimiento, el optimizador usado por este algoritmo es el descenso del gradiente.

Parámetros

El registro de imágenes permite alinear mediante técnicas de correlación de intensidades, calcula la información mutua entre una imagen fija y una imagen en movimiento que deben ser registradas. Se utilizó la función imregister de MATLAB y utilizando el escenario de captura Multimodal el cual usa la métrica Mattes mutual information (Mattes et al., 2001) y asignado un optimizador con los siguientes parámetros:

- `optimizer.InitialRadius = 0.0005;`
- `optimizer.Epsilon = 0.000005;`
- `optimizer.GrowthFactor = 1.3;`
- `optimizer.MaximumIterations = 500;`

3.7. Cómputo evolutivo

Inspiradas en la teoría de evolución de las especies por Charles Darwin es el uso de una serie de metaheurísticas estocásticas (que también son llamados "Algoritmos Evolutivos"). El enfoque de estas metaheurísticas consiste en explorar un espacio de búsqueda de manera eficiente, estos algoritmos evolutivos son metaheurísticas estocásticas porque su comportamiento se basa en números aleatorios y no necesariamente generan el mismo resultado cada vez que se ejecutan (Coello, 2019).

El proceso de evolución se enfoca en la habilidad de sobrevivir de un organismo o sistema en un ambiente cambiante y competitivo, la idea básica consiste en tener individuos que se adapten y transmitan su descendencia, y esta descendencia debe continuar adaptándose (Dale y Schantz, 2007).

Haciendo una diferencia entre los términos computo evolutivo (CE) y algoritmos evolutivos (AE) se pueden definir de la siguiente manera: Computo evolutivo (CE) hace referencia a la resolución de problemas basados en modelos de evolución como la selección natural, supervivencia de los más aptos y reproducción (D. Fogel, 1993 ,Dale y Schantz, 2007). Un algoritmo evolutivo (AE) es una búsqueda estocástica óptima para un problema especificado, los componentes principales de un algoritmo evolutivo son:

- **Codificación** de las soluciones representadas mediante cromosomas.
- **Función de aptitud** para evaluar el comportamiento y la supervivencia de los individuos.
- **Inicialización** de la población.
- Operadores de **selección**.
- Operadores de **reproducción**.

Existen diferentes formas en las que los algoritmos evolutivos son implementados, resultando en diferentes paradigmas de cómputo evolutivo:

- **Algoritmos genéticos**: Modelan la evolución genética.
- **Programación genética**: Basada en algoritmos genéticos.
- **Programación evolutiva**: Se deriva de la simulación adaptativa del comportamiento en la evolución.

- **Estrategias evolutivas**: Modelan parámetros estratégicos de variación en la evolución.
- **Evolución diferencial**: Similar a los algoritmos genéticos, pero se diferencian en el uso de mecanismos de reproducción.
- **Co-evolución**: Donde individuos evolucionan a través de la cooperación o en competencias entre sí, adquiriendo características necesarias para sobrevivir.

3.7.1. Codificación (Cromosomas)

En el contexto de cómputo evolutivo (CE), cada individuo representa una solución para un problema de optimización, estas características se presentan como cromosomas (también llamadas genomas) y cada variable que necesita ser optimizada se le refiere como gen (la unidad más pequeña de información). Las características de los individuos pueden ser divididas en dos clases: genotipo y fenotipo.

Un genotipo es la información genética de un organismo, esta información puede o no ser manifestada u observada en el individuo. El fenotipo es un término que se usa en genética para denotar los rasgos físicos visibles de un individuo, dentro de los algoritmos evolutivos se refiere al valor decodificado de una solución candidata(Coello, 2019).

Un paso importante en los AE es encontrar una representación apropiada de las soluciones, la mayoría de las soluciones son representadas usando vectores en un tipo específico de datos, estos vectores (generalmente cadenas binarias) tienen la siguiente estructura: A cada cadena binaria que codifica una variable se le denomina gen y a cada bit dentro de un gen se le denomina alelo, a la concatenación de varios genes se le denomina cromosoma. La figura (3.38) muestra una representación usando vectores.

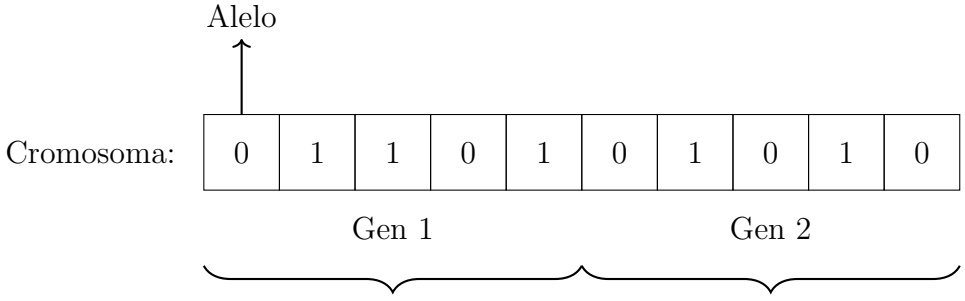


Figura 3.38: Representación de un cromosoma como un vector binario.

3.7.2. Población inicial

El primer paso de un AE es generar una población inicial. La forma estándar utilizada para generar dicha población consiste en asignar valores aleatorios dentro del espacio de soluciones de los genes para cada cromosoma. La selección aleatoria se asegura que la población inicial sea una representación uniforme del espacio de búsqueda. El tamaño de población afecta en términos de complejidad computacional.

3.7.3. Función de aptitud

Representa los requerimientos que debe tener la población para adaptarse. Es una función o procedimiento que asigna la calidad medida en el genotipo. Tomando como ejemplo la función un valor entero x que maximice x^2, la aptitud del genotipo 11001 se define como la decodificación ($11001 \rightarrow 25$) y tomando el cuadrado: $25^2 = 625$. Dependiendo el problema a tratar debido a que la función de aptitud se asocia con maximización, matemáticamente es trivial cambiar minimización en maximización (Eiben y Smith, 2015).

3.7.4. Operador de selección

El operador de selección consiste en distinguir a los individuos de la población basados en su calidad y así permitir que los mejores individuos se conviertan en padres de la siguiente generación. Este operador es responsable de mejorar la calidad de los individuos y generalmente de manera probabilística.

3.7.5. Selección por torneo

En la selección por torneo se seleccionan un grupo de individuos de forma aleatoria de la población $n_{ts} < n_s$, donde n_s es el número total de individuos en la población. El desempeño de los individuos n_{ts} es comparado y los mejores individuos del grupo es seleccionado y retornado por el operador. Si el tamaño del torneo n_{ts} no es muy grande esto previene que los mejores individuos dominen el espacio de busqueda, por otro lado si n_{ts} es muy pequeño, las probabilidades de seleccionar individuos con bajo rendimiento se incrementan.

3.7.6. Operador de reproducción y mutación

El operador de reproducción (también llamado de recombinación) es el proceso de generar descendencia de dos o más padres seleccionados. Este es un operador estocástico, la selección de las partes de cada padre para ser combinados depende de métodos aleatorios, el uso de más de dos padres es posible y fácil de implementar, pero no es comúnmente utilizado.

La idea principal consiste en combinar dos individuos con características diferentes pero deseables, dando como resultado descendencia que combina ambas características (Eiben y Smith, 2015). La figura (3.39) muestra los descendientes al aplicar el operador de cruza utilizado en algoritmos genéticos.

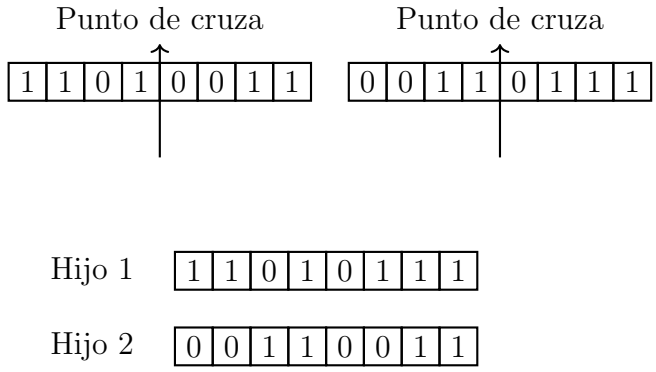


Figura 3.39: Resultado de aplicar la reproducción utilizando un punto de cruza. El hijo 1 tiene los primeros 4 bits del padre 1 y los últimos 4 bits del padre 2, mientras que el hijo 2 tiene los primeros 4 bits del padre 2 y los últimos 4 bits del padre 1.

El operador de mutación es siempre estocástico, es decir depende de una serie de decisiones aleatorias, por lo tanto la meta principal es realizar un cambio de manera aleatoria en los genes del cromosoma, generando así nuevo material genético en la población para incrementar la diversidad. La mutación dentro de CE ha tenido distintos roles; en programación genética no es muy utilizado mientras que en algoritmos genéticos tradicionalmente se ha utilizado como un operador base y en programación evolutiva es el único operador de variación.

Este operador no debe modificar la mayor parte del material genético en los individuos por esta razón la mutación debe aplicarse con cierta probabilidad. La figura (3.40) muestra un ejemplo en un vector binario donde se aplica mutación.

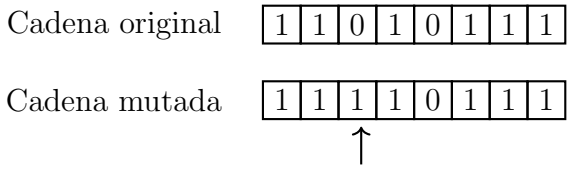


Figura 3.40: ejemplo de mutación, la posición 3 del vector es mutada cambiando de 0 a 1.

3.7.7. Programación evolutiva

Ideado por Lawrence J. Fogel (L. J. Fogel et al., 1966) se basa en cuatro componentes estándar dentro de los algoritmos evolutivos: Inicialización, variación, evaluación y selección. La programación evolutiva tuvo sus inicios como automatas de estado finito en un intento por emular el comportamiento inteligente. Estas máquinas operan sobre un conjunto finito de símbolos de entrada y posee un número finito de estados internos. La figura (3.41) muestra un ejemplo de un autómata de estado finito presentado originalmente por Fogel (D. Fogel, 1994).

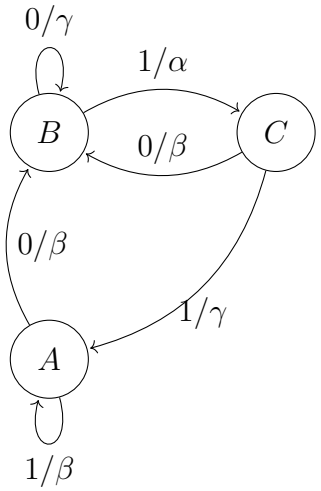


Figura 3.41: Autómata de estado finito con tres estados.

Fogel propuso la programación evolutiva usando automatas de estados finitos de la siguiente manera: Una población de padres es expuesta al medio ambiente, esto significa la observación de una secuencia de símbolos en un momento dado. A cada padre le es dado cada símbolo de entrada y cada símbolo de salida es comparado con el siguiente símbolo de entrada, el valor de la predicción es medido basado en la función de aptitud. Los hijos son creados mutando a los padres (por conveniencia cada padre genera un hijo). Esta mutación es generada con una distribución de probabilidad, y el número de mutaciones por hijo también es elegido con respecto a una distribución de probabilidad. Los hijos son evaluados en el medio ambiente de la misma manera que los padres.

Los automatas que obtienen la mejor función de aptitud son los seleccionados para convertirse en padres de la siguiente generación. La mitad de los autómatas son almacenados, así la población de padres se mantiene del mismo tamaño, este proceso se repite hasta que es necesario hacer una predicción actual del siguiente símbolo. Cada automata que es almacenado debe ser elegido en la mitad superior de la población en términos de puntuación.

La programación evolutiva se basa solamente en mutación, debido a la falta de un operador de cruza esta técnica puede considerarse como una búsqueda enumerativa (D. B. Fogel y Stayton, 1994). El método consiste de tres pasos (que se repiten hasta que se cumple la condición de paro o una solución adecuada es obtenida).

1. Se selecciona una población inicial con soluciones al azar (padres) y la representación de la población. En la figura (3.42) puede ser vista como una pila en estructura de datos, el tamaño de población es importante en términos de optimización.

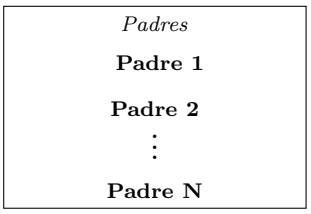


Figura 3.42: Se genera una población inicial N.

2. Se crea una nueva población (hijos) donde las soluciones creadas al azar son sometidas a mutación la figura (3.43) muestra el proceso de creación de población al azar y su mutación.

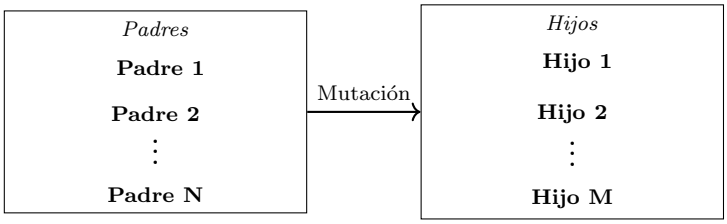


Figura 3.43: Se genera una población de padres y estos son mutados para generar hijos.

3. Cada elemento de los padres e hijos en la nueva población es sometido a un torneo estocástico donde son evaluados utilizando la función de aptitud con elementos elegidos aleatoriamente de la población para determinar N padres que se conservan para la siguiente generación. La figura (3.44) muestra el proceso de evaluación mediante un torneo.

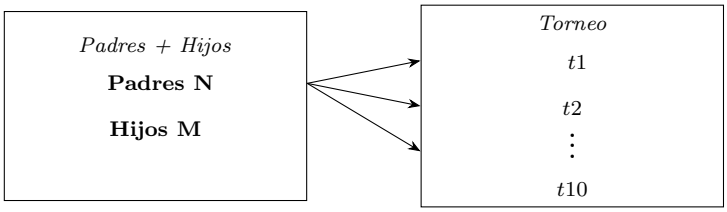


Figura 3.44: Cada elemento de la población de padres e hijos es evaluado con los elementos elegidos al azar (generalmente 10 elementos) de la poblacion mediante un torneo.

Al término de estos pasos, la población de padres e hijos S son ordenados en orden descendente y se selecciona la mitad superior para retener el tamaño de población original. La figura (3.45) ilustra este proceso.

La figura (3.46) muestra un resumen general que describe la programación evolutiva propuesta por Fogel.

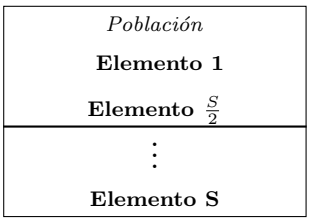


Figura 3.45: Una vez ordenada la población se selecciona la mitad superior para la siguiente generación.

Codificación: Vectores con números reales
Recombinación: Ninguna
Mutación: Perturbación Gaussiana
Selección de padres: Determinista (cada padre crea un hijo mediante mutación)
Selección de sobrevivientes: Probabilística
Especialidad: Auto-adaptación de los tamaños de paso de la mutación en la meta-EP

Figura 3.46: La principal característica de la programación evolutiva es que no utiliza el operador de recombinación.

3.8. Métricas

Cuando se dispone de un modelo o sistema, es necesario analizar qué tan bien funciona con los datos de prueba. Las predicciones realizadas se comparan con los valores reales de los datos, y se calcula su rendimiento. Las llamadas predicciones de clases nominales son una forma conveniente de medir el rendimiento basado en los datos actuales y predichos. Para ello, se utiliza una matriz de confusión, donde las instancias predichas se encuentran en las columnas, mientras que las filas representan las instancias de los datos reales. A partir de la matriz de confusión, se pueden obtener diferentes medidas de rendimiento que proporcionan diversas perspectivas sobre lo que constituye un buen modelo. En resumen, la matriz de confusión se emplea para evaluar las fortalezas y debilidades del modelo.

Dentro de estas métricas se encuentran la precisión, la sensibilidad (recall) y el f-score. Para calcular estas métricas, es necesario utilizar una matriz de confusión que describa el rendimiento del modelo.

3.8.1. Matriz de confusión

Esta tabla describe el rendimiento del modelo, el objetivo es describir este rendimiento desde varios ángulos.

Los términos que se utilizan son los siguientes:

- **Verdadero positivo (VP)**: El modelo predice correctamente Si (píxel donde existe telaraña).

- **Verdadero negativo (VN)**: El modelo predice correctamente No (fondo o segundo plano de la imagen).
- **Falso positivo (FP)**: El modelo predice erróneamente Si.
- **Falso negativo (FN)**: El modelo predice erróneamente No.

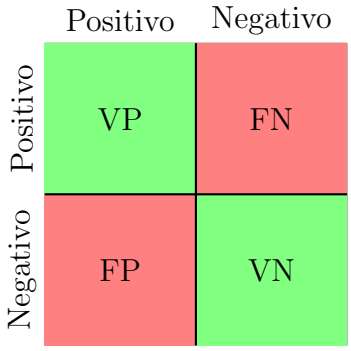


Figura 3.47: Ejemplo de matriz de confusión.

3.8.2. Precisión, sensibilidad y F-score

La precisión determina determina la proporción de verdaderos positivos (VP) respecto al número total de positivos.

$$\text{Precisión} = \frac{VP}{VP + FP} \tag{3.29}$$

La sensibilidad se puede definir como el proceso de determinar de forma correcta cuantos son los verdaderos positivos, es decir que tan bueno es el modelo identificando los casos positivos.

$$\text{Sensibilidad} = \frac{VP}{VP + FN} \tag{3.30}$$

El F-score es la media harmónica de la precisión y sensibilidad.

$$\text{F-score} = 2 \cdot \frac{\text{Precisión} \cdot \text{Sensibilidad}}{\text{Precisión} + \text{Sensibilidad}} \tag{3.31}$$

3.8.3. Medición del rendimiento del detector de bordes

El rendimiento de un detector de bordes se puede evaluar mediante el conteo de los bordes falsos y de aquellos que no han sido identificados, comparando estos resultados con la estimación esperada. Para ello, es posible generar una imagen sintética en la que los bordes son conocidos. Con esta imagen se puede determinar el número de bordes correctamente detectados, los no identificados y los falsos positivos, comparando los resultados

del detector con los valores reales de la imagen original. Cabe señalar que estos resultados pueden variar en función del umbral, el tamaño del filtro de suavizado y otros factores (Ramesh Jain, 1995).

3.8.4. Base de datos BSDS500

Esta base de datos es una extensión de BSDS300 e incluye 300 imágenes destinadas al entrenamiento y validación, junto con 200 imágenes adicionales anotadas manualmente por humanos. Se emplea como referencia estándar en pruebas comparativas de detección de contornos, la figura (3.48) muestra un ejemplo de los bordes predefinidos en una imagen, que se utilizan como referencia para evaluar el rendimiento de distintos algoritmos.

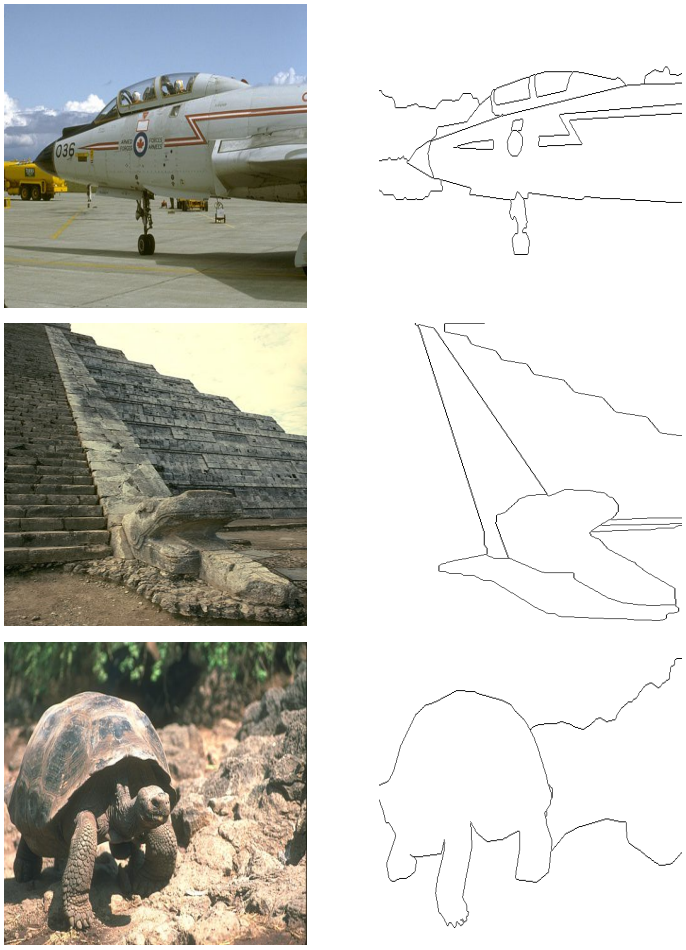


Figura 3.48: Imagen original y su correspondiente imagen binaria con anotaciones manuales.

Capítulo 4

Metodología

En la metodología se proponen tres métodos para poder segmentar las regiones de interés en las imágenes proporcionadas, cada método tiene sus ventajas y desventajas, aunque los resultados son subjetivos y depende de la perspectiva del investigador estas propuestas se enfocan en cubrir las necesidades y objetivos del proyecto, las métricas mencionadas son utilizadas para dar una idea del comportamiento y efectividad de detección de bordes basada en anotaciones humanas y diferenciar que tan lejano o cercano se encuentran las propuestas a un buen detector de bordes. Los métodos propuestos son los siguientes:

- **Segmentación basada en detección de bordes:** Se aplica un detector de bordes con el método de *Laplace of Gaussian (LoG)* mediante kernels y convolución, así como la ayuda de operaciones morfológicas para segmentar las imágenes, la función objetivo consiste en encontrar el valor óptimo de la ventana (σ) que maximice la cantidad de bordes detectados en la imagen. La búsqueda de los mejores kernels se realiza con técnicas de computación evolutiva.
- **Segmentación basada en umbralización local:** Se computa un umbral para cada pixel en la imagen basado en el contenido de los píxeles vecinos, mediante una ventana deslizante sobre una imagen en escala de grises. Los métodos propuestos son el algoritmo de Niblack y Feng.
- **Segmentación basada en un modelo de red neuronal convolucional:** Se propone utilizar las imágenes segmentadas con las propuestas anteriores para entrenar un modelo basado en U-Net y segmentar las áreas de interés.

4.1. Equipo Fotográfico

La base de datos de telarañas fue fotografiada con un equipo Canon modelo EOS 7D DS126251 con las siguientes especificaciones:

- **Tipo:** Cámara digital réflex de lentes intercambiables AF/AE con flash integrado

- **Medios de grabación:** Tarjeta CF Tipo I y II, tarjetas CF compatibles con UDMA, mediante medios externos (disco duro USB v.2.0, mediante un Transmisor Inalámbrico de Archivos WFT-E5A opcional)
- **Formato de imagen:** 22.3 x 14.9 mm (tamaño APS-C)
- **Lentes compatibles:** Lentes EF de Canon, incluyendo los lentes EF-S (la longitud focal equivalente a 35 mm es aproximadamente 1.6x de la longitud focal del lente)
- **Base del lente:** Montura EF de Canon

Las características del sensor de la imagen son las siguientes:

- **Tipo:** Sensor CMOS de alta sensibilidad, alta resolución y una sola placa grande
- **Píxeles efectivos:** Aprox. 18.0 megapíxeles
- **Unidad de píxel:** 4.3 micrones cuadrados
- **Total de píxeles:** Aprox. 19.0 megapíxeles
- **Relación de aspecto:** 3:2 (Horizontal: Vertical)
- **Sistema de Filtro de Color:** Filtros de colores primarios RGB
- **Filtro de paso bajo:** Posición fija en la parte frontal del sensor CMOS
- **Característica de remoción de polvo:** (1) Limpieza Automática del Sensor

4.2. Base de datos

Se obtuvo un total de 40 imágenes las cuales fueron procesadas con las metodologías mencionadas a continuación, la lista puede verse en la tabla (A.1). Esta es la lista de imágenes que fueron procesadas para generar las máscaras binarias ajustando tanto los parámetros del tamaño de ventana (kernel) como los valores de umbralización, la base de datos corresponde a imágenes donde la estructura de la telaraña no presenta algún daño (antes) y las imágenes que presentan algún daño en su estructura (después).

Las imágenes fueron nombradas con una nomenclatura para su estudio con una tabla de referencia que se muestra en (4.1) se presenta un ejemplo del uso de esta nomenclatura: `13-03-AL-CA-después` es fecha = `13-03`, tratamiento = *Sol arriba*, datos de araña = *Con araña*, datos de impacto = *Después*.

4.3. Pre-procesamiento

El pre-procesamiento de las imágenes consistió en editarlas y ajustarlas para aplicar el algoritmo, dado que las imágenes originales contenían elementos irrelevantes para la detección de bordes y presentaban errores en la captura, se realizaron ediciones que incluyeron recortes, reflejos y alineación.

Categoría	Descripción
Fecha	**Día-Mes**
Tratamiento	**S** = Sombra, **IL** = Sol de frente o detrás, **AL** = Sol arriba
Datos de la araña	**CA** = Con araña, **SA** = Sin araña
Datos de impacto	**Antes**, **Después**

Tabla 4.1: Nomenclatura del nombrado de imágenes.

4.3.1. Recorte de imágenes

Las imágenes originales fueron recortadas, esto debido a que al momento de tomar la fotografía de la telaraña estas se encontraban en sus marcos de aluminio, por ello se seleccionó solo el área de interés sin el marco, la figura (4.1) muestra un ejemplo de recorte de imágenes de la base de datos.

(a) Imagen original (b) Imagen recortada

Figura 4.1: Ejemplo de recorte de imágenes originales

4.3.2. Imágenes reflejadas

Algunas imágenes tuvieron que editarse para reflejarlas debido a que al momento de tomar la fotografía de la telaraña con el marco de aluminio fueron colocadas del lado equivocado con respecto a la primera toma de la fotografía (imágenes tomadas antes de impacto) de la telaraña.

Nombre de la Imagen
20-02-IL-SA-despues
18-02-S-SA-despues
17-03-S-CA-despues
17-03-IL-CA-despues
13-03-AL-CA-despues
18-02-AL-SA-despues
13-03-AL-SA-despues

Tabla 4.2: Lista de imágenes reflejadas.

4.3.3. Alineación de imágenes

Algunas imágenes no se encontraban alineadas en un ángulo recto de 90 grados con respecto a su marco de aluminio para aplicar un corte del área de interés, por ello se aplicó rotación, la tabla (4.3) muestra los ángulos de rotación que se aplicaron a las imágenes.

Nombre de la Imagen	Rotación	Dirección
18-02-AL-SA-despues	1°	Contraria a las agujas del reloj
18-02-IL-SA-despues	1°	Contraria a las agujas del reloj

Tabla 4.3: Lista de imágenes y su ángulo de rotación

4.4. Medición manual del daño

Los investigadores que trabajan en la INBIOTECA actualmente utilizan un software especializado para realizar mediciones de forma manual, este programa llamado (fiji) imageJ permite seleccionar una imagen y establecer una escala para calcular una distancia conocida en centímetros. Mediante la creación de segmentos dentro del programa pueden realizar estas mediciones pero la tarea resulta ser muy observacional y exhaustiva para poder realizar los cálculos.

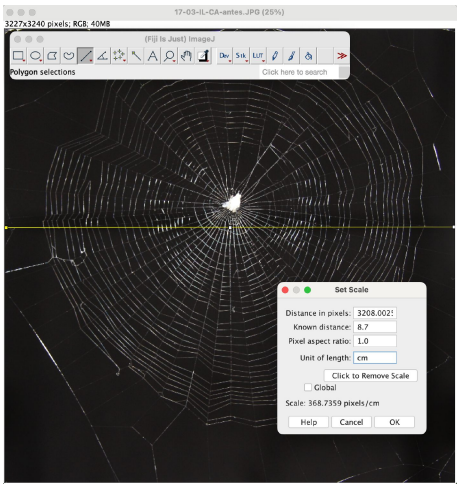


Figura 4.2: Interfaz software (fiji) imageJ donde se establece la distancia conocida para realizar mediciones en centímetros

4.5. Evaluación de la base de datos segmentada

Para la evaluación de las imágenes segmentadas en cada método se utilizó la tabla de evaluación descrita en (Apéndice B) donde las imágenes finales fueron proporcionadas al experto y evaluó numéricamente del del 1 al 5, donde 1 representa la calificación más baja y 5 la más alta según el análisis visual en cada imagen. La valoración total puede verse en (4.1) donde se realiza la suma de calificaciones de cada aspecto en la imagen segmentada, las preguntas $Q1$ a $Q5$ son valoraciones positivas, mientras que $Q6$ tiene una valoración negativa.

$$\text{Valoración total} = (Q_1 + Q_2 + Q_3 + Q_4 + Q_5 - Q_6) \tag{4.1}$$

4.6. Segmentación basada en detección de bordes

En el procesamiento digital de imágenes existe un problema recurrente al aplicar detección de bordes o transformaciones como el método de Canny o un filtro paso-bajo. Mientras estas operaciones ayudan a resaltar ciertas características presentes en las imágenes, el procesamiento se ve obstaculizado debido al ruido, resolución y calidad de la imagen.

Un procedimiento clave para identificar las regiones de interés en una imagen y la extracción de características consisten en obtener la información de los bordes en la imagen. Con la ayuda de los bordes detectados se pueden utilizar técnicas para realizar una segmentación parcial o completa, el siguiente método consiste en el uso de búsqueda de kernels para realizar la detección de bordes.

Los algoritmos para la detección de bordes cuentan con tres pasos (Ramesh Jain, 1995):

- **Filtrado:** El filtrado es comúnmente usado para mejorar el rendimiento en la detección de bordes con respecto al ruido, aunque existe una parte en contra en la que

mientras más filtros son aplicados para reducción de ruido, esto resulta en pérdida de las intensidades de los bordes.

- **Mejora:** Enfatiza los cambios en las intensidades locales de los valores de los píxeles, y esto se realiza generalmente mediante el cómputo de la magnitud del gradiente.
- **Detección:** Consiste en la detección de bordes fuertes, frecuentemente se utiliza un umbral.

Un paso extra es utilizado en algunos algoritmos

- **Localización:** Consiste en la localización exacta del borde, la resolución de subpixeles puede ser usada para esta tarea.

En este trabajo se propone un algoritmo diseñado para procesar imágenes llamado ELoGED (Evolutive Laplacian of Gaussian Edge Detection) para la detección de bordes, como imágenes de entrada para la aplicación del algoritmo consiste en una base de datos de telarañas, el área de interés que se desea resaltar es la red construida por la araña.

Esta propuesta consiste en utilizar la representación multi-escala el cual reduce la imagen original con un factor predeterminado creando una pila de imágenes las cuales pueden ser procesadas individualmente para la extracción de características.

Una aproximación al método de Laplaciano de la Gaussiana es usado para generar una población inicial de kernels y mediante cómputo evolutivo usando la técnica de programación evolutiva es usada para mejorar y seleccionar las mejores soluciones. Finalmente se aplica convolución con los kernels resultantes para la detección de bordes y con ayuda de operaciones morfológicas se segmenta la imagen.

Mediante la exploración de distintos valores del parámetro sigma en el filtro Laplaciano de Gauss (LoG), la aplicación de parámetros de cómputo evolutivo y el ajuste del tamaño del kernel, se espera maximizar la detección de posibles bordes en las imágenes, incrementando las áreas de interés (ROI), en contraste con los métodos clásicos de la literatura que se centran en el uso de umbrales.

4.6.1. Aproximación a LoG (Laplaciano de la Gaussiana)

Retomando las definiciones de (3.2.2) la segunda derivada del filtro Gaussiano lo podemos representar en la ecuación (4.2).

$$\mathrm{LoG}(\sigma, x, y) = c \cdot \frac{(x^2 + y^2 - 2\sigma^2)}{\sigma^4} \cdot e^{-\frac{x^2+y^2}{2\sigma^2}} \tag{4.2}$$

Donde c es un factor de normalización que verifica que la suma de los elementos del filtro sea cero, en este trabajo se normaliza el filtro resultante mediante la siguiente fórmula:

$$\text{filtro\_normalizado}(i, j) = \mathrm{LoG}(i, j) - \frac{\sum_{i=1}^{m} \sum_{j=1}^{n} \mathrm{LoG}(i, j)}{m \times n} \tag{4.3}$$

Donde i, j representa el valor de la posición en la fila y columna y m, n representan el tamaño del kernel (filas y columnas). Esto debido a que el Laplaciano de la Gaussiana

no garantiza que la suma de los elementos del filtro sea cero, por lo tanto para asegurarse de que la suma sea cero se resta a cada elemento del filtro el promedio de todos los elementos. En este trabajo se usa el término aproximación ya que el uso de la técnica de programación evolutiva usando mutación (4.6.11), los valores del filtro no vuelven a normalizarse asegurando esta propiedad. Tomando en cuenta que el algoritmo propuesto no cumple con la restricción que la suma de los valores de la matriz sea 0 para ser considerado un filtro LoG. La función matemática es usada como base para la generación del kernel para una aproximación, más no debe ser considerado como un filtro LoG.

La figura (4.3) muestra una representación gráfica de la función LoG donde c = 1 y σ = 1

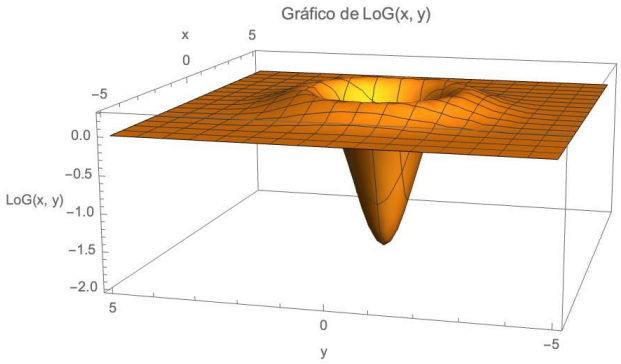


Figura 4.3: Gráfico 3D de LoG (Laplace of Gaussian)

4.6.2. Generación del kernel

El proceso para el conteo de pixeles es mediante el uso de convolución y un filtro o kernel de tamaños impares ($3 \times 3, 5 \times 5, 7 \times 7, etc.$) estos kernels son generados por la formula (4.4). Donde σ (sigma) representa valores entre $1 < \sigma \leq 9$ y $k(i, j)$ es la posición de un elemento del kernel (Nixon y Aguado, 2006).

$$\mathrm{k}(i, j) = LoG(\sigma, i - \frac{n-1}{2}, j - \frac{m-1}{2}) \tag{4.4}$$

Usando esta fórmula la figura (4.4) y (4.5) muestra algunos ejemplos de filtros LoG escalados a valores enteros.

$$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

Figura 4.4: Ejemplo de filtro de 3×3.

Por practicidad se utilizaron valores reales en lugar de escalar a valores enteros, la motivación es que las poblaciones generadas al azar con valores enteros puede resultar poco práctico, ya que la mayor parte de los filtros resultantes contienen valores en cero en todos

$$\begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

Figura 4.5: Ejemplo de filtro de 5×5.

sus elementos y que resulta en un filtro que no es aplicable para la detección de bordes, el usar los resultados con representación real se obtiene un espacio de soluciones más grande. Como ventaja la selección del umbral para segmentar las imágenes se encuentra en un rango más pequeño, la figura (4.6) muestra un filtro generado con la funcion (4.4) con valor de $\sigma = 3.09$.

$$\begin{bmatrix} 0.057\,86 & 0.018\,91 & 0.003\,62 & 0.018\,91 & 0.057\,86 \\ 0.018\,91 & -0.030\,96 & -0.050\,45 & -0.030\,96 & 0.018\,91 \\ 0.003\,62 & -0.050\,45 & -0.071\,55 & -0.050\,45 & 0.003\,62 \\ 0.018\,91 & -0.030\,96 & -0.050\,45 & -0.030\,96 & 0.018\,91 \\ 0.057\,86 & 0.018\,91 & 0.003\,62 & 0.018\,91 & 0.057\,86 \end{bmatrix}$$

Figura 4.6: Ejemplo de filtro de tamaño 3×3.

4.6.3. Convolución

La convolución dentro del algoritmo se realiza mediante la función $imfilter$ de MATLAB, esta función toma como entrada un vector en 2D que representa la imagen I en escala de grises, un vector en 2D que se representa como el kernel k, la opción de filtrado especificado como "conv" que realiza el filtrado multidimensional utilizando convolución y como opción de relleno "replicate" donde los valores del arreglo de entrada que se encuentran fuera de los límites del arreglo se asumen iguales que el valor de borde más cercano del arreglo.

4.6.4. Detector cruce por cero

Una vez procesada la imagen con el kernel obtenido, la operación cruce por cero (3.2.3) itera sobre cada pixel de la imagen buscando píxeles adyacentes donde los valores cambian de signo, estos puntos donde los valores cambian rápidamente pueden considerarse como bordes, con ayuda de un umbral (3.1.5) se puede establecer un criterio de selección o tolerancia para poder filtrar ruido o falsos bordes en la imagen. Los valores de umbral utilizados para el cruce por cero fueron un rango de ($T \in [10, 50]$).

4.6.5. Segmentación con operaciones morfológicas

Para la segmentación de las imágenes se decidió utilizar el elemento estructurante con forma de cuadrado, con un tamaño de 2 píxeles. Esto con base a los comentarios del investigador donde se buscan segmentos de la telaraña sean visibles pero tomando en cuenta el área real de la telaraña, siendo este SE el mejor de todos para esta tarea.

El resultado es un vector con el tamaño especificado que puede verse en la figura (4.7).

$$\begin{bmatrix} 1 & 1 \\ 1 & 1 \end{bmatrix}$$

Figura 4.7: Elemento estructurante en forma de cuadrado de 2×2.

4.6.6. Operación de dilatación

El uso de la función *imdilate* de MATLAB permite segmentar la imagen con un elemento estructurante SE, una vez detectados los bordes con LoG y el cruce por cero. Donde I representa la imagen en escala de grises y "se" el elemento estructurante.

4.6.7. Conteo de pixeles

Dado que la función de aptitud para el proceso de cómputo evolutivo (3.7) consiste en maximizar el conteo de píxeles donde un pixel con valor 1 en la imagen representa el área de interés es decir un borde en la imagen para su posterior segmentación. Es necesario realizar el conteo de los píxeles para seleccionar al individuo más apto, tomamos la suma de los valores distintos a 0 en la matriz resultante después de aplicar la detección de bordes usando la función *nnz* de MATLAB. El número de píxeles se almacena en una variable para posteriormente aplicar la selección de sobrevivientes en la siguiente generación al aplicar cómputo evolutivo, la figura (4.8) muestra un ejemplo donde se realiza el conteo de píxeles.

$$A = \begin{bmatrix} 1 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{bmatrix}$$

Figura 4.8: Tomando como ejemplo el arreglo A, aplicando la función nnz de MATLAB obtenemos como resultado el valor de 5

4.6.8. Cómputo evolutivo

En este trabajo se decidió utilizar el paradigma de algoritmos evolutivos llamado programación evolutiva (PE), ya que por las características que deben tener los individuos

resulta difícil aplicar cruza y este paradigma no implementa el operador de recombinación, lo cual resulta práctico para la exploración y generación de posibles soluciones.

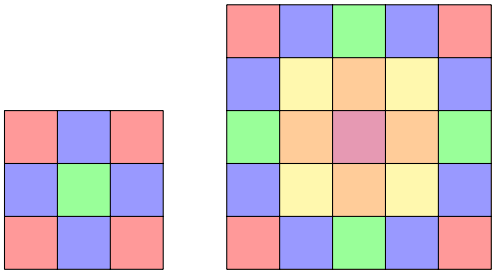
4.6.9. Codificación

Para la codificación de los individuos se utilizó una representación matricial de tamaño $n \times n$ con números reales en sus elementos, la figura (4.9) muestra esta representación en forma de matriz.

$$\begin{bmatrix} 0.425\,394\,977 & -0.117\,869\,939 & 0.425\,394\,977 \\ -0.117\,869\,939 & -1.340\,720\,156 & -0.117\,869\,939 \\ 0.425\,394\,977 & -0.117\,869\,939 & 0.425\,394\,977 \end{bmatrix}$$

Figura 4.9: Individuo usando una representación matricial de 3×3 con valores reales en sus elementos.

Como una condición o restricción, el individuo debe tener un patrón de valores que se repiten en el kernel, la figura (4.10) muestra un patrón de colores que representan valores reales de tamaños 3×3 y 5×5, de la misma manera esto sucede en tamaños de 7×7, 9×9, etc.



(a) 3x3 (b) 5x5

Figura 4.10: Patrón de colores que representan los valores reales que se repiten dentro del kernel.

4.6.10. Función de aptitud

La función de aptitud consiste en la maximización de detección de bordes en la imagen y se define como la suma total de los píxeles detectados como bordes, donde I es una imagen binaria de dimensiones $M \times N$, y cada pixel $I(i, j)$ toma el valor:

$$I(i,j) = \begin{cases} 1 & \text{si } X(i,j) = 1, \\ 0 & \text{si } X(i,j) \neq 1. \end{cases}$$

$$f(I) = \sum_{i=1}^{M} \sum_{j=1}^{N} \mathbb{I}_{\{X(i,j)=1\}}$$

Cada kernel, tanto generado como mutado, se evalúa obteniendo un conteo de bordes que se registra para comparar con las generaciones sucesivas durante el proceso de cómputo evolutivo. Dichos bordes se detectan en función del umbral especificado y del tamaño del kernel. La modificación del kernel mediante mutación tiene como objetivo detectar nuevos bordes.

4.6.11. Mutación

En el proceso de búsqueda mediante cómputo evolutivo para el kernel óptimo, consiste de los siguientes pasos:

- Una población inicial de tamaño p con tamaño de kernel k y una condición de paro s.
- Se selecciona un valor aleatorio de los kernels generados, este valor representa un patrón que se repite en el kernel, los cuales son utilizados para aplicar mutación, la figura (4.11) muestra esta la selección de valores a mutar.

$$\begin{bmatrix} 0.002\,21 & 0.000\,70 & 0.000\,12 & 0.000\,70 & 0.002\,21 \\ 0.000\,70 & \boxed{-0.00117} & -0.001\,89 & \boxed{-0.00117} & 0.000\,70 \\ 0.000\,12 & -0.001\,89 & -0.002\,66 & -0.001\,89 & 0.000\,12 \\ 0.000\,70 & \boxed{-0.00117} & -0.001\,89 & \boxed{-0.00117} & 0.000\,70 \\ 0.002\,21 & 0.000\,70 & 0.000\,12 & 0.000\,70 & 0.002\,21 \end{bmatrix}$$

Figura 4.11: Valores seleccionados de la matriz de tamaño 5×5 para realizar mutación.

Tomando como ejemplo el valor -0.00117, estos valores son los candidatos para aplicar mutación en el kernel, el proceso consiste en generar un nuevo kernel utilizando un valor de σ generado aleatoreamente con la función *rand*, esta función genera un valor aleatorio extraído de la distribución uniforme en el intervalo $(0, 1)$. La función (4.5) genera estos valores aleatorios para sigma, siendo a el límite inferior y b el límite superior y N el número de muestras.

$$\sigma = a + (b - a) \cdot \text{rand}(N, 1) \tag{4.5}$$

Usando este nuevo valor de σ obtenemos como resultado un nuevo kernel con valores diferentes, este kernel se muestra en la figura (4.12).

Reemplazamos los valores del kernel original en la figura (4.11) con los nuevos valores de la figura (4.12) en las posiciones indicadas, dando como resultado el kernel de la figura (4.13).

Este proceso se repite dentro del algoritmo hasta que se cumple la condición de paro.

$$\begin{bmatrix} 0.006\,79 & 0.002\,62 & 0.000\,78 & 0.002\,62 & 0.006\,79 \\ 0.002\,62 & \boxed{-0.00378} & -0.006\,58 & \boxed{-0.00378} & 0.002\,62 \\ 0.000\,78 & -0.006\,58 & -0.009\,77 & -0.006\,58 & 0.000\,78 \\ 0.002\,62 & \boxed{-0.00378} & -0.006\,58 & \boxed{-0.00378} & 0.002\,62 \\ 0.006\,79 & 0.002\,62 & 0.000\,78 & 0.002\,62 & 0.006\,79 \end{bmatrix}$$

Figura 4.12: Generación de kernel 5×5 con valor de sigma aleatorio.

$$\begin{bmatrix} 0.002\,21 & 0.000\,70 & 0.000\,12 & 0.000\,70 & 0.002\,21 \\ 0.000\,70 & -0.003\,78 & -0.001\,89 & -0.003\,78 & 0.000\,70 \\ 0.000\,12 & -0.001\,89 & -0.002\,66 & -0.001\,89 & 0.000\,12 \\ 0.000\,70 & -0.003\,78 & -0.001\,89 & -0.003\,78 & 0.000\,70 \\ 0.002\,21 & 0.000\,70 & 0.000\,12 & 0.000\,70 & 0.002\,21 \end{bmatrix}$$

Figura 4.13: Kernel de 5×5 con mutación aplicada.

4.6.12. Selección de sobrevivientes

Los sobrevivientes se seleccionan en base del modelo de programación evolutiva (3.7.7) debido a que no existe cruza por la restricción de la creación del kernel se involucran poblaciones de padres e hijos donde cada solución es evaluada con q soluciones obtenidas de forma aleatoria. Los padres e hijos son mutados como se detalló en (4.6.11), después de aplicar la mutación se crea una nueva colección donde unimos a padres e hijos en una lista de tamaño R, se reliza un torneo (3.7.5) donde se seleccionan las mejores soluciones asignando un contador de victorias para cada solución.

Esto consiste en elegir elementos de la colección al azar (comúnmente 10) y los comparamos mediante el resultado de conteo de bordes en cada solución, por ejemplo si el conteo de bordes de una solución en la lista de padres e hijos es mayor que la comparación de la colección elegida al azar, entonces el conteo de victorias es incrementado en $+1$ para esa solución (kernel) en la colección de pares e hijos en caso contrario el conteo de victorias no cambia la figura (4.14) muestra un diagrama del proceso.

4.6.13. Aplicación del algoritmo en niveles multi-escala

Al aplicar la generación de niveles de las imágenes de la base de datos, 7 niveles en total son generados para cada imagen, las resoluciones varían debido a los recortes generados para las áreas de interés como se comentó en (4.2). La idea inicial de generar estos niveles consiste en extraer las características individuales de cada nivel, esto significa hacer que el algoritmo encuentre nuevos segmentos y obtener la segmentación correspondiente, pero se encontró un problema al realizar la segmentación.

En el primer nivel no surgen inconvenientes dado que la resolución de la imagen es la máxima, pero en los niveles superiores existe un problema a la hora de aplicar el algoritmo en la pirámide de resoluciones (ver 3.1.8) para segmentar, debido a que se debe realizar

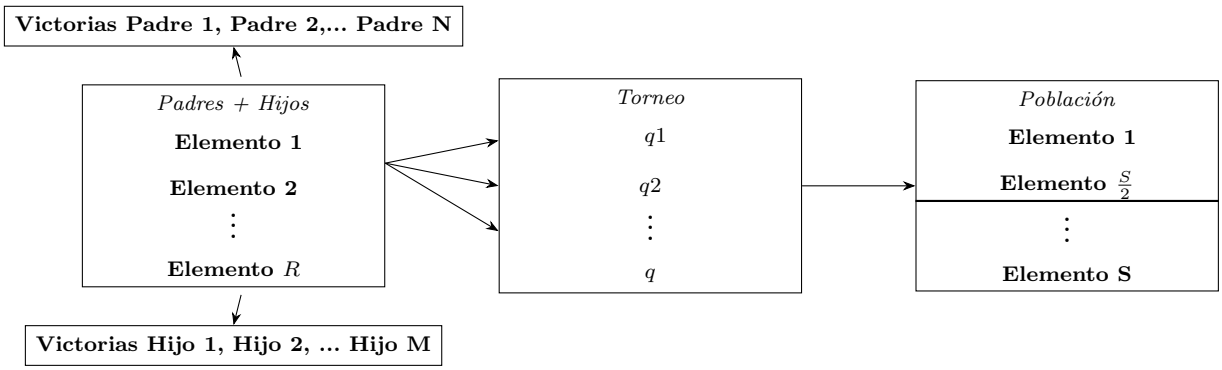
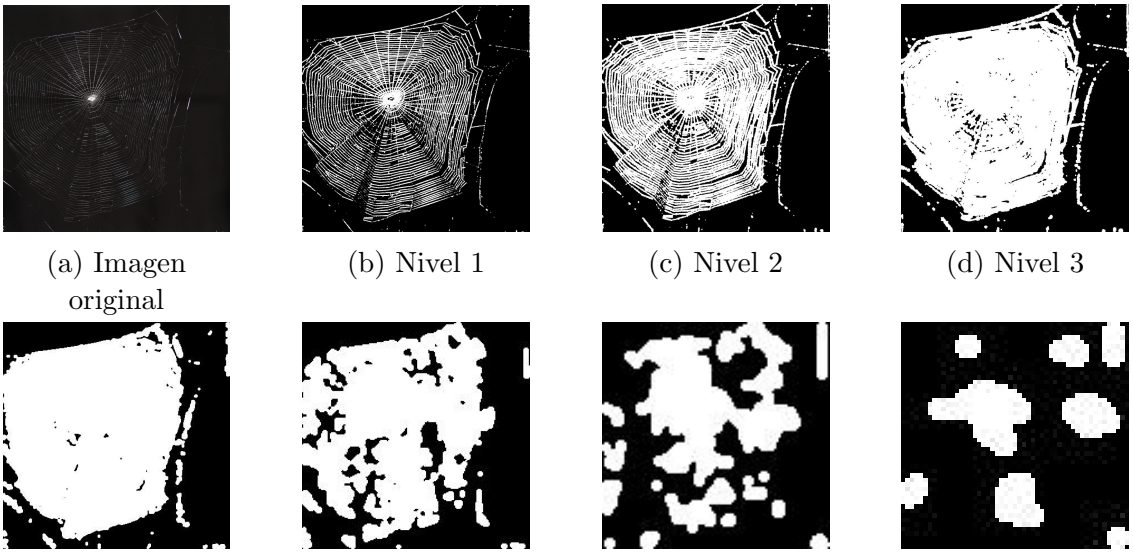


Figura 4.14: Se itera cada solución realizando un torneo con una colección elegida al azar de tamaño q, incrementando el número de victorias según el caso. Como paso final ordenamos la lista en orden descendente basado en el conteo de victorias de los individuos y seleccionamos los primeros $\frac{S}{2}$ elementos de la lista (tamaño original de población).

un ajuste con la operación de dilatación, el cambio brusco de resolución ejemplo: del nivel 1 (3232×3196) al nivel 4 (408×400) píxeles afecta significativamente el grosor de la segmentación y resulta una máscara binaria poco clara, sin mucha precisión en los segmentos de la telaraña.



(a) Imagen original (b) Nivel 1 (c) Nivel 2 (d) Nivel 3

(e) Nivel 4 (f) Nivel 5 (g) Nivel 6 (h) Nivel 7

Figura 4.15: Segmentación aplicada a distintos niveles en el algoritmo propuesto.

Debido a esto, se descartó la posibilidad de utilizar los distintos niveles de la pirámide de resoluciones construida y la obtención de las características en cada nivel para realizar una reconstrucción total de la pirámide. Como parte del trabajo futuro se propone utilizar una modificación del algoritmo propuesto para reducir el tamaño del elemento estructurante en cada nivel.

Figura 4.16: Imagen reconstruida de la segmentación en cada nivel de la pirámide.

Como se puede observar en la (figura 4.16) la reconstrucción final de la pirámide resulta poco legible y sin segmentos de interpretación para las áreas de interés. Como se comentó anteriormente, cambiar el tamaño del elemento estructurante o inclusive la forma utilizada podría mejorar la segmentación en niveles más altos de la pirámide y aumentar las características presentes en la misma.

4.6.14. Diagrama general del algoritmo

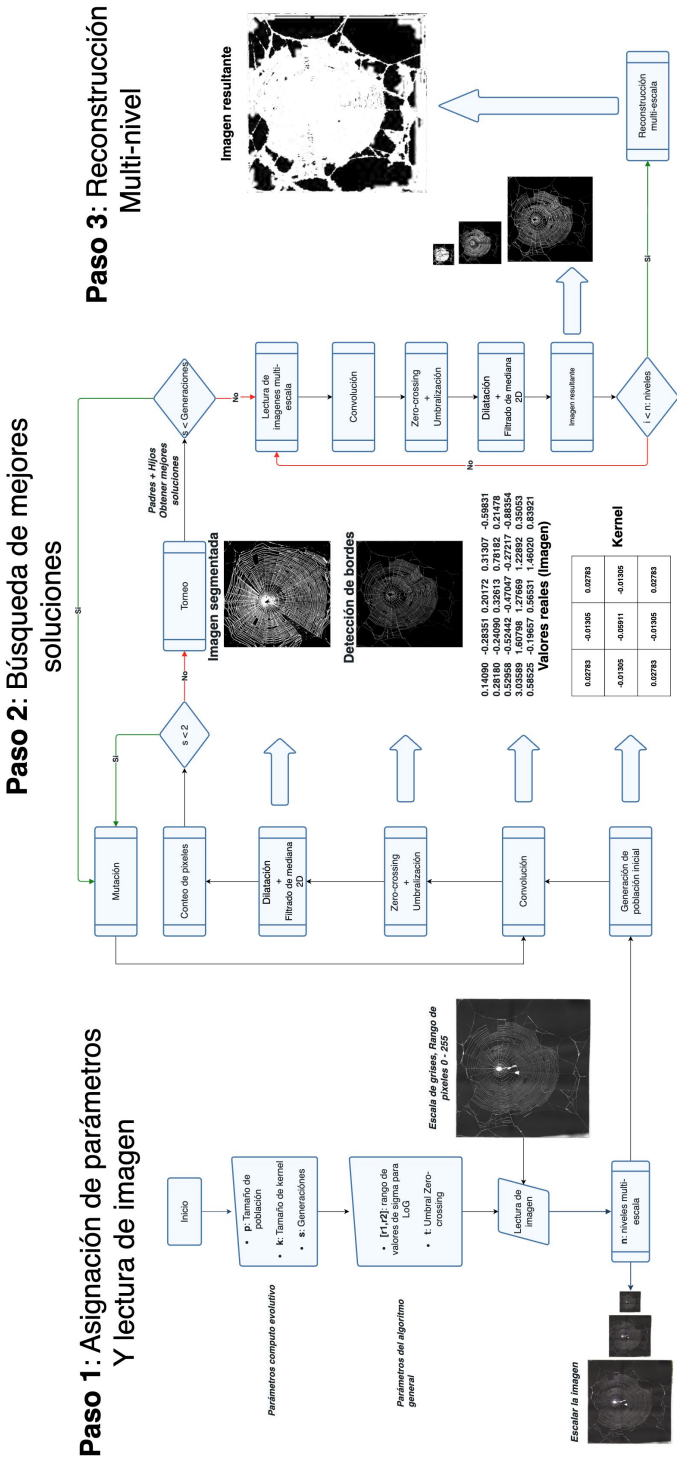


Figura 4.17: Algoritmo general

Esta es una descripción del funcionamiento del algoritmo propuesto que consta de los

siguientes pasos:

1. El algoritmo inicia con la imagen de entrada, esta imagen puede ser de cualquier tipo pero en este trabajo el enfoque se encuentra en detección de la estructura de la red de una telaraña (ver 1.1), esta imagen es convertida a escala de grises.

2. Como paso siguiente, la representación multi-nivel es generada (3.1.8) con base en la imagen de entrada, esto consiste en una reducción de la resolución de la imagen mediante un factor generando así n niveles (imágenes) con distintas dimensiones.

3. Después de procesar la representación multi-nivel se asignan los parámetros para el cómputo evolutivo, estos parámetros se pueden ver en la tabla (4.4).

Parámetro	**Valor**
Tamaño de población	p
Tamaño de kernel	k
Condición de paro	s

Tabla 4.4: Parámetros de computo evolutivo

4. Se asignan los parámetros del algoritmo general, estos parámetros se pueden ver en la tabla (4.5).

Parámetro	**Valor**
Rango de valores sigma	$[r1, r2]$
Umbral	t

Tabla 4.5: Parámetros del algoritmo general

5. Ahora se genera la población inicial de kernels de tamaño p (4.6.2).

6. Se realiza la convolución de la imagen de entrada con cada individuo de la población generada (3.4).

7. Se aplica la detección de cruce por cero con un umbral especificado después de aplicar convolución para detectar los bordes en la imagen (3.2.3).

8. Se realiza el conteo de pixeles (4.6.7).

9. Se aplica la operación morfológica de dilatación para segmentar la imagen (3.3.1).

10. Se aplica un filtro de mediana para eliminar ruido. (3.5).

11. Ahora se tiene una población de kernels con sus respectivos conteos de pixeles (bordes) estos serían los "padres" en el cómputo evolutivo, como paso siguiente aplicamos mutación a la población de soluciones (4.6.11).

12. Obtenemos el conteo de pixeles de los kernels mutados "hijos" usando los pasos 6, 7 y 8.

13. Aplicamos el torneo y elegimos las mejores soluciones (4.6.12).

14. Repetimos los pasos 11, 12 y 13 hasta cumplir la condición de paro.

15. Una vez que se cumple la condición de paro, se obtiene el kernel óptimo de la población final y se repite el algoritmo en cada nivel de la imagen generada en el paso 2.

4.7. Segmentación basada en umbralización local

En muchos casos un único umbral no es suficiente para separar los objetos del fondo, al seleccionar un umbral de forma manual puede no ser suficiente para resaltar los objetos o áreas de interés sin incluir ruido en la imagen, una alternativa es variar el umbral en la imagen. Los métodos locales de umbralización computan un umbral para cada pixel basado en los píxeles vecinos, estos métodos generalmente funcionan mejor para imágenes de baja calidad. Los algoritmos que se presentan a continuación han sido ampliamente utilizados para el reconocimiento óptico de caracteres (OCR, Optical Character Recognition) enfocados a la digitalización de textos (Khurshid et al., 2009).

El funcionamiento de estos algoritmos consisten en el uso de una ventana de tamaño variable que recorre la imagen y que computa de forma local el valor de umbral T óptimo para cada pixel. A continuación se presentan dos alternativas para segmentar la base de datos utilizando este método de umbralización local.

4.7.1. Algoritmo de Niblack

Presentado originalmente en el libro publicado por su autor (Niblack, 1986)

La idea consiste en variar este umbral basado en la media local de los píxeles, el umbral variable se puede computar de la siguiente manera:

$$t = k \cdot \sigma(i,j) + \overline{v(i,j)} \tag{4.6}$$

Para $k = -0.18$, $\sigma(i,j)$ y $\overline{v(i,j)}$ siendo la desviación estándar y la media respectivamente en la vecindad de (i,j).

Los parámetros usados para computar el umbral local son:

- **Imagen**: imagen en escala de grises.
- **Tamaño de ventana**: valor entero impar (3x3, 5x5, 7x7, etc).
- **k**: parámetro para computar el umbral (fijado a -0.18 por el autor).

Parametrización del algoritmo de Niblack

La configuración del algoritmo puede representarse como:

$$\text{Configuración} = ([M\ N], K, \text{Offset}, \text{Padding})$$

- **[M N]**: Consiste en el tamaño de ventana usado por el algoritmo.
- **K**: Es un valor que se multiplica con la desviación estándar. El valor asignado por los autores es -0.18.
- **Offset**: Corresponde al valor de desplazamiento de la ventana.
- **Padding**: Corresponde al valor para lidiar con los bordes de la imagen.

4.7.2. Algoritmo de Feng

Al igual que el algoritmo de Niblack, el algoritmo de Feng (Feng y Tan, 2004) es útil para la extracción de texto en documentos de baja calidad, su propuesta consiste en una umbralización local para realizar una binarización en imágenes en escalas de grises.

El método propuesto consiste en el cálculo de la media local m el mínimo M y la desviación estándar s de los valores en escala de grises en una primera ventana local. Para compensar el efecto negativo de la iluminación irregular se aplica un rango dinámico de la desviación estándar de la escala de grises R_{S} en una ventana secundaria en lugar de aplicarlo en toda la imagen.

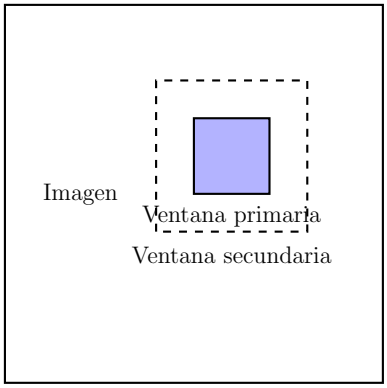


Figura 4.18: Ventana primaria y ventana secundaria por el método propuesto por Feng.

El artículo comenta que el tamaño de la ventana local secundaria puede ajustarse adecuadamente para tolerar diferentes grados de desigualdad en la iluminación. La propuesta consiste en el cálculo de umbralización con la siguiente función:

$$T = (1 - \alpha_1) \times m + \alpha_2 \times \left(\frac{s}{R_s}\right) \times (m - M) + \alpha_3 \times M \tag{4.7}$$

Donde:

$$\alpha_2 = k_1 \left(\frac{s}{R_S}\right)^{\gamma}, \quad \alpha_3 = k_2 \left(\frac{s}{R_S}\right)^{\gamma}$$

Y los valores:

$$\alpha_1, \gamma, k_1, k_2$$

Son constantes positivas.

La ecuación (4.7) consiste de tres componentes y tres coeficientes (α_1, α_2 y α_3) que se definen para permitir una ponderación más flexible y adaptativa de estos tres componentes. Los coeficientes (α_2, α_3) se establecen de manera adaptativa en función de la desviación estándar local normalizada $\left(\frac{s}{R_s}\right)$

4.7.3. Parametrización del algoritmo de Feng

La configuración del algoritmo puede representarse como:

$$\text{Configuración} = ([M\,N], [P\,Q], \alpha, \gamma, k_1, k_2, \text{Padding})$$

- **[M N]**: Consiste en la ventana primaria o la ventana interior.
- **[P Q]**: Consiste en la ventana secundaria y debe ser mayor a la ventana primaria.
- α: Desplaza el umbral.
- γ: Especifica el valor de un exponente.
- k_1: Es una constante multiplicativa.
- k_2: Es una constante multiplicativa.
- **Padding**: Corresponde al valor para lidiar con los bordes de la imagen.

4.8. Segmentación basada en un modelo de red neuronal convolucional

Las redes neuronales convoluciones (CNN) son consideradas una evolución de MLP (Multi-Layer perceptrons), las redes neuronales contienen múltiples capas que permiten que las capas individuales encuentren características dentro de la imagen. Las primeras capas de convolución aprenden algunas características básicas (bordes y líneas), las siguientes capas prenden características más complejas (círculos, cuadrados, etc.).

Las siguientes capas encuentran características más complicadas como caras, la llanta de un coche, etc, las arquitecturas CNN siguen el mismo patrón que las redes neuronales, se apilan capas ocultas sobre otras, los pesos se asignan aleatoriamente durante el entrenamiento y se aplican funciones de activación, se calcula el error $(y - \hat{y})$ y se aplica backpropagation para actualizar los pesos.

La arquitectura de las CNN consisten en lo siguiente:

- Capa de entrada
- Capas convolucionales para la extracción de características
- Capas totalmente conectadas para la clasificación
- Salida o predicción

La idea básica de las redes neuronales es que las neuronas aprendan características de la entrada, en CNN un mapa de características es la salida de un filtro aplicado a la capa anterior. Se llama mapa de características porque es una representación de dónde se encuentra un cierto tipo de característica en la imagen.

4.8.1. U-Net

A continuación se presenta una breve descripción del funcionamiento de U-Net (Ronneberger et al., 2015) y se procedió a analizar las características que la red debe tener para obtener buenos resultados en la segmentación.

Se decidió probar con la arquitectura de U-Net debido a que funciona bien con pocas imágenes de entrenamiento, además de que es ampliamente usado en imágenes médicas donde se requiere alta precisión en la detección de características.

La figura muestra (4.19) la arquitectura de la red U-Net:

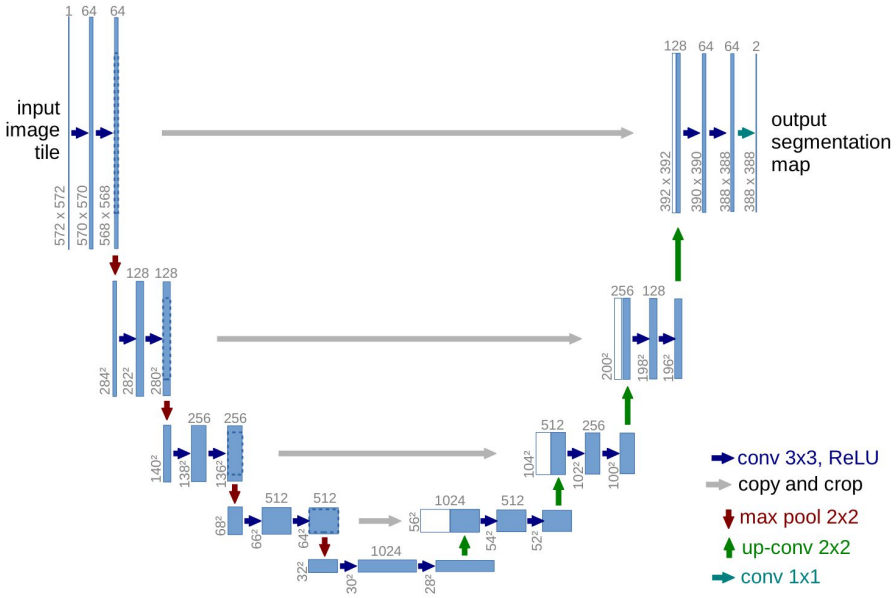


Figura 4.19: Arquitectura U-net (Ronneberger et al., 2015, ver p. 2)

En esencia podemos dividir la arquitectura en dos partes, la parte izquierda llamada el camino de contracción y la parte derecha llamado el camino de expansión. El camino de contracción consisten en la aplicación de convolución de tamaño 3x3 seguido de una activación ReLU y una operación max-pooling de 2x2 con stride de 2 para aplicar downsampling (reducción de resolucion), en cada downsampling se multiplica el número de canales de características.

En el camino de expansión se aplica un upsampling (aumento de muestreo) mediante una convolución 2x2 (up-convolution) el cual reduce el número de canales de características. Se realiza una concatenación del mapa de características recortado del camino de contracción y se aplican dos convoluciones de 3x3 seguidas por la activación ReLU.

Al final se aplica una capa de convolución de 1x1 para mapear cada vector de características de 64 componentes al número deseado de clases, en total la red contiene 23 capas de convolución.

4.8.2. Entrenamiento

Con ayuda de las máscaras binarias generadas con los métodos de umbralización local (4.7.1 y 4.7.2) se procedió a entrenar dos modelos basados en U-Net y realizar pruebas de segmentación con los modelos generados y observar si los modelos aprenden características en las imágenes que ayuden a segmentar mejor las imágenes de las telarañas. Usando un tamaño de imagen de entrada de 1024×1024 píxeles, debido a que el consumo de memoria durante el entrenamiento depende de la resolución de la imagen y las capacidades del equipo de computo, se consideró que este tamaño de resolución fue aceptable y contiene suficiente resolución para observar detalles de la estructura de la telaraña.

Para llevar a cabo el entrenamiento se utilizó el entorno de programación de Google

Codelab con las siguientes características:

- GPU: NVIDIA A100.
- Memoria RAM de GPU: 40 GB.
- Memoria RAM del sistema: 83.5 GB.

A continuación se presentan los parámetros utilizados para cada modelo.

4.8.3. Hiperparámetros

Los hiperparámetros controlan varios aspectos del modelo, estos incluyen costo de tiempo y memoria al correr el algoritmo. No existen números mágicos al momento de ajustar estos valores y estos dependen en gran medida del tipo de problema. Uno de los hiperparámetros más importantes es la tasa de aprendizaje, ya que un valor muy bajo requiere de muchas épocas para que el modelo converja, por otro lado un valor muy alto el modelo toma pasos más grandes en la curva de aprendizaje y existe una gran probabilidad que el algoritmo oscile y se aleje del valor mínimo (Elgendy, 2020), la figura (4.20) ilustra el comportamiento de este hiperparámetro.

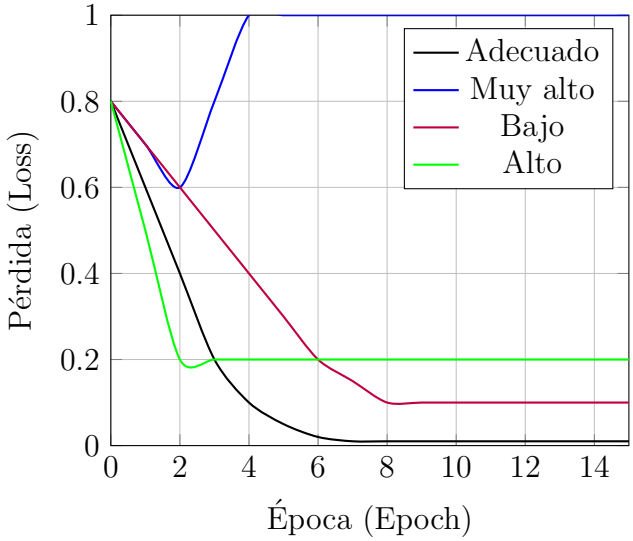


Figura 4.20: Diferencias entre el ajuste del hiperparámetro de tasa de aprendizaje.

Basado en la figura anterior podemos observar el siguiente comportamiento:

- **Aprendizaje bajo:** La tasa de pérdida decrece, pero necesita más tiempo para converger.
- **Aprendizaje alto:** La tasa de pérdida obtiene buenos resultados al inicio, pero está muy lejos de alcanzar el valor óptimo.

- **Aprendizaje muy alto:** La tasa de pérdida decrece inicialmente, pero empieza a incrementar y se aleja demasiado del valor óptimo.
- **Aprendizaje adecuado:** La tasa de pérdida decrece consistentemente hasta alcanzar el valor mínimo posible.

Se utilizaron los siguientes hiperparámetros de entrenamiento tomando como imágenes de entrada las segmentaciones basadas en el algoritmo de umbralizacion local de Niblack y Feng.

Parámetro	**Descripción**	**Algoritmo de Niblack**	**Algoritmo de Feng**
Épocas	También conocido como iteración de entrenamiento, es cuando un modelo completa un ciclo y ve todo el conjunto de entrenamiento.	200	200
Tamaño de lote	Especifica el número de muestras durante el procesamiento.	4	5
Paciencia	Número de épocas sin mejora después de las cuales el entrenamiento se detendrá.	25	25
Tasa de aprendizaje	La tasa de aprendizaje controla qué tan rápidamente se adapta el modelo al problema.	0.0006	0.00135
División de validación	La muestra de datos utilizada para proporcionar una evaluación imparcial del ajuste del modelo.	0.3	0.3

Tabla 4.6: Hiperparámetros de entrenamiento para las máscaras binarias obtenidas mediante el algoritmo de umbralización local con Niblack y Feng en dos configuraciones diferentes.

Para el entrenamiento se excluyeron ciertas imágenes debido a su baja calidad (imagen fuera de foco), la tabla (4.7) enlista las imágenes de telarañas que fueron excluidas.

Nombre de la Imagen
13-03-AL-CA-antes
13-03-S-SA-antes

Tabla 4.7: Imágenes excluidas del entrenamiento de los modelos.

La tabla (4.8) enlista las imágenes que fueron seleccionadas y excluidas del entrenamiento para realizar las pruebas finales y evaluar los modelos, basado en la calificación final (Anexos E, F) de las imágenes segmentadas con los algoritmos de Niblack y Feng.

Nombre de la Imagen
18-02-AL-CA-antes
18-02-AL-CA-despues
18-02-IL-CA-antes
118-02-IL-CA-despues

Tabla 4.8: Imágenes utilizadas para evaluar al modelo

La tabla (4.9) muestra que, al evaluar el modelo con las imágenes de prueba, se obtuvo un rendimiento de F-Score del 90 % utilizando las imágenes segmentadas con el algoritmo de Niblack. Por otro lado, la tabla (4.10) indica un rendimiento de F-Score del 80 % al emplear las imágenes segmentadas con el algoritmo de Feng. Esta diferencia podría deberse a que los elementos segmentados en las imágenes son considerablemente más finos.

Precisión	Sensibilidad	F-score	VP	FP	FN	VN
87.03568837	93.74390768	90.26533651	7 593 594	1 131 096	506 766	33 539 561

Tabla 4.9: Resultados de la evaluación del rendimiento del modelo entrenado con imágenes segmentadas mediante el algoritmo de Niblack.

Precisión	Sensibilidad	F-score	VP	FP	FN	VN
80.57606739	81.01561064	80.79524122	2 714 872	654 456	636 176	38 765 513

Tabla 4.10: Resultados de la evaluación del rendimiento del modelo entrenado con imágenes segmentadas mediante el algoritmo de Feng.

A continuación se presentan se presenta la gráfica de evaluación del modelo y de rendimiento, así como las curvas de aprendizaje.

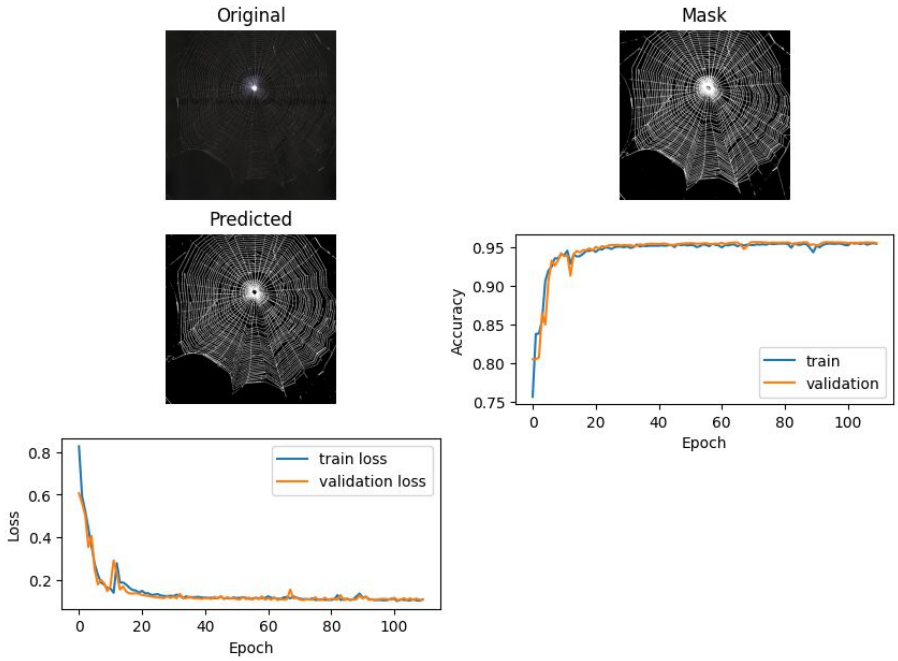


Figura 4.21: Gráficas del modelo de entrenamiento con imágenes segmentadas usando el algoritmo de Niblack.

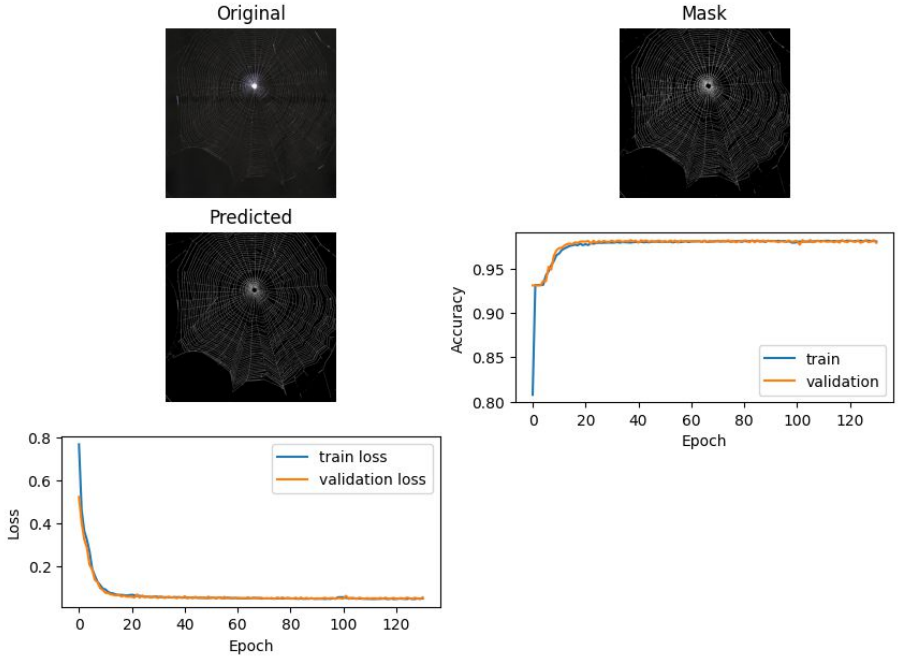


Figura 4.22: Gráficas del modelo de entrenamiento con imágenes segmentadas usando el algoritmo de Feng.

4.9. Registro de imágenes y cálculo del daño estructural

Como se comentó en (3.6.1) el registro de imágenes consistió en aplicar la métrica en MATLAB de información mutua utilizando la función imregister, para ello se creó un optimizador que emplea la modalidad "multimodal" en lugar de la modalidad "monomodal", dado que la modalidad multimodal está diseñada para trabajar con los mismos niveles de brillo y contraste en ambas imágenes, lo cual no ocurre con las imágenes segmentadas en este caso.

Este tipo de registro se utiliza ampliamente en imágenes de resonancia magnética (IRM) o de tomografía computarizada (CT) donde se especifica que los sensores o escáneres operan con el mismo brillo y contraste, o se emplean imágenes obtenidas de diferentes dispositivos. En contraste, en el caso multimodal las imágenes pueden provenir del mismo dispositivo pero con diferentes configuraciones de exposición.

El optimizador cuenta con cuatro parámetros:

- GrowFactor: El optimizador lo utiliza para controlar la tasa a la que se expande el radio de búsqueda en el espacio de parámetros, valor asignado (1.3).
- Epsilon: Controla la precisión de la convergencia al ajustar el tamaño mínimo de radio de búsqueda, valor asignado (0.000005).
- InitialRadius: Tamaño inicial del radio de búsqueda, especificado como un escalar positivo, valor asignado (0.0005).
- MaximumIterations: Número máximo de iteraciones del optimizador, el registro podría converger antes de que el optimizador alcance el número máximo de iteraciones, valor asignado (500).

En el anexo (G) se presenta un ejemplo de registro de imágenes mediante el optimizador. Este proceso consiste en alinear la primera imagen con respecto a la segunda. Una vez finalizado el registro, se almacena la imagen ajustada, lo que permite calcular el daño a partir de la diferencia de píxeles de las imágenes binarizadas, este ejemplo se puede observar en el anexo (H), al final se realiza el conteo de las diferencias en píxeles al comparar la imagen registrada con la imagen de referencia. A partir de estos datos, se determinó el porcentaje de daño estructural.

Capítulo 5

Resultados

5.0.1. Comparación entre la propuesta y algoritmos clásicos de la literatura

Para evaluar el rendimiento del algoritmo propuesto, se compararon los resultados obtenidos con los métodos Canny, Prewitt, Roberts y Sobel. Todos se configuraron con un umbral de 0.1 con excepción Canny que requiere un umbral superior establecido en 0.2, junto con un valor de ventana (sigma) de 1. En el caso del algoritmo propuesto, el umbral se definió en el rango de 0 a 255 que corresponde a la escala de grises de la imagen. Se implementó una escala de conversión que permite utilizar rangos equivalentes a los métodos mencionados, empleando la fórmula (5.1):

$$\text{Umbral} = \frac{\text{valor} - min_2}{max_2 - min_2}\,(max_1 - min_1) + min_1 \quad \text{donde} \quad \begin{cases} min_2 = 0 \\ max_2 = 1 \\ min_1 = 0 \\ max_1 = 255 \\ 0 \leq valor \leq 1 \end{cases} \tag{5.1}$$

Los parámetros del algoritmo se pueden observar en la tabla 5.1:

Parámetro	**Valor**
Tamaño de población	100
Tamaños de kernel	3×3, 5×5, 7×7
Condición de paro	100
Umbral	25.5

Tabla 5.1: Parámetros para evaluar el rendimiento del algoritmo propuesto en comparación con los métodos presentes en la literatura.

La comparación visual de los métodos se puede observar en la figura (5.1), la cual muestra la detección de bordes junto con la anotación realizada por humanos.

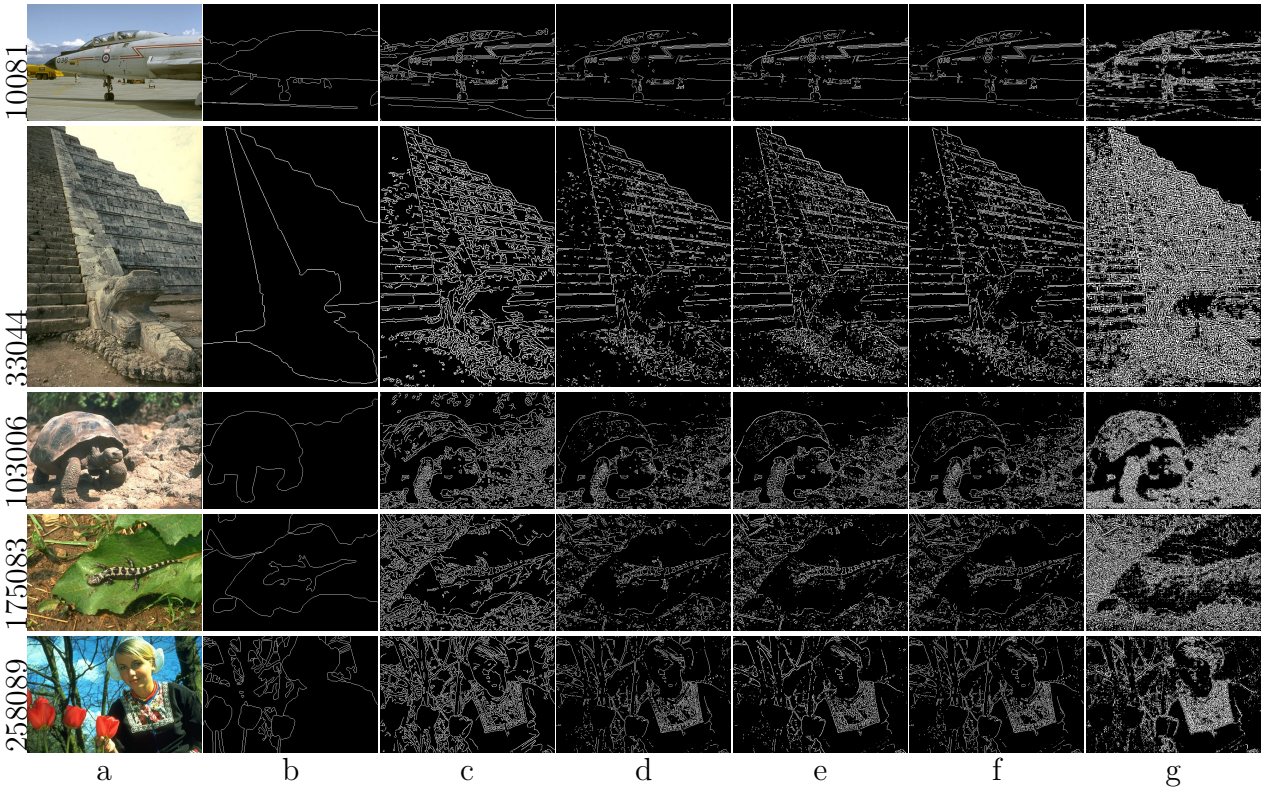


Figura 5.1: Resultado de la detección de bordes, (a) imagen original, (b) anotación realizada por humanos, (c) bordes producidos por Canny, (d) bordes producidos por Prewitt, (e) bordes producidos por Roberts, (f) bordes producidos por Sobel, (g) bordes producidos por el algoritmo propuesto con umbral de 25.5 y tamaño de kernel 3×3.

Se presentan las gráficas que muestran los resultados del cálculo del F-score para cada imagen utilizando diferentes tamaños de kernel en las figuras (5.2, 5.3, 5.4).

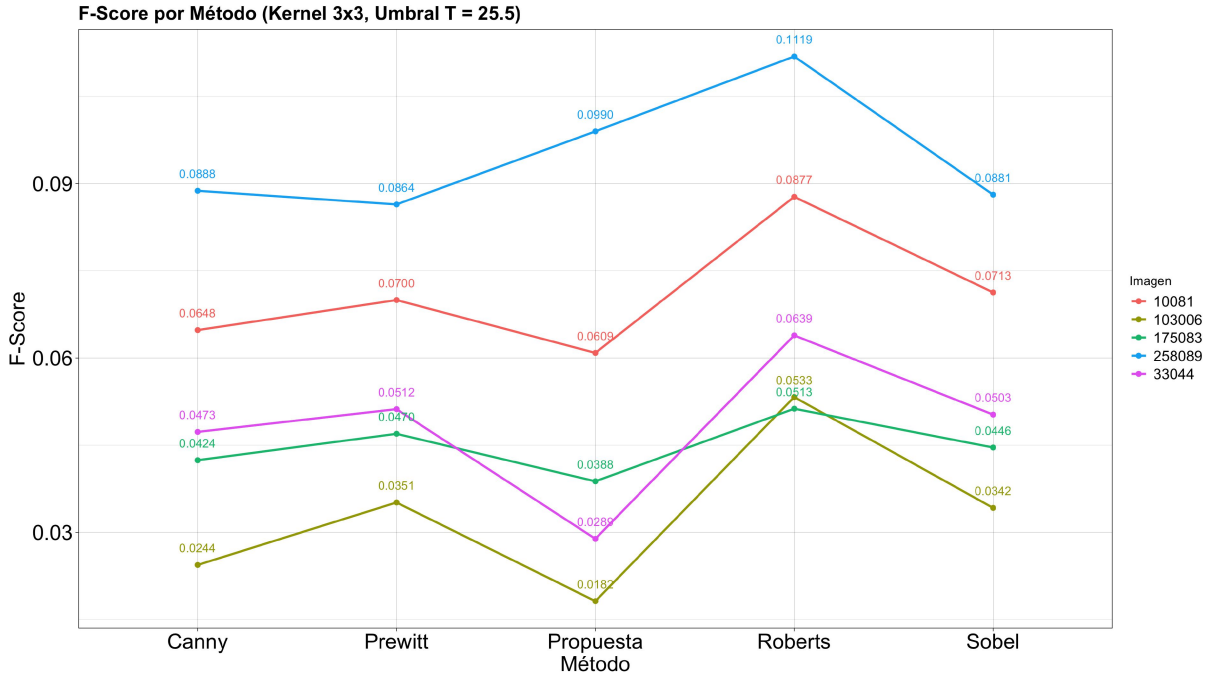


Figura 5.2: Gráfica comparativa del cálculo del F-Score en cada imagen. Con umbral de 25.5 y tamaño de kernel de 3×3.

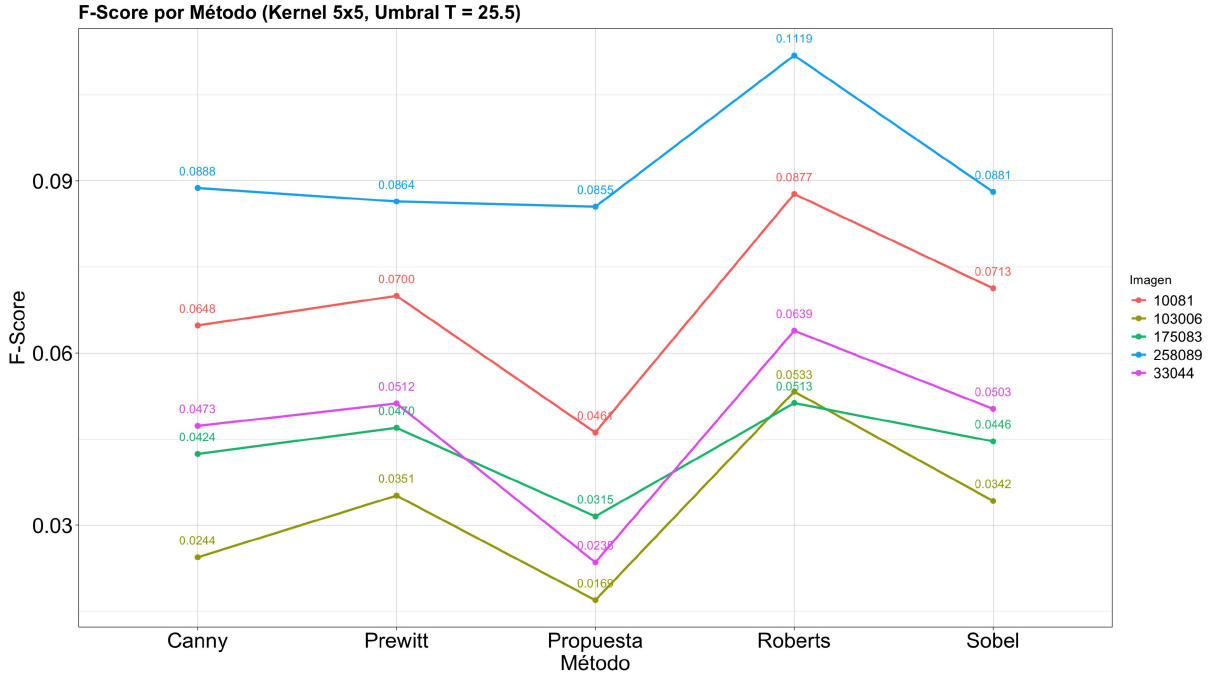


Figura 5.3: Gráfica comparativa del cálculo del F-Score en cada imagen. Con umbral de 25.5 y tamaño de kernel de 5×5.

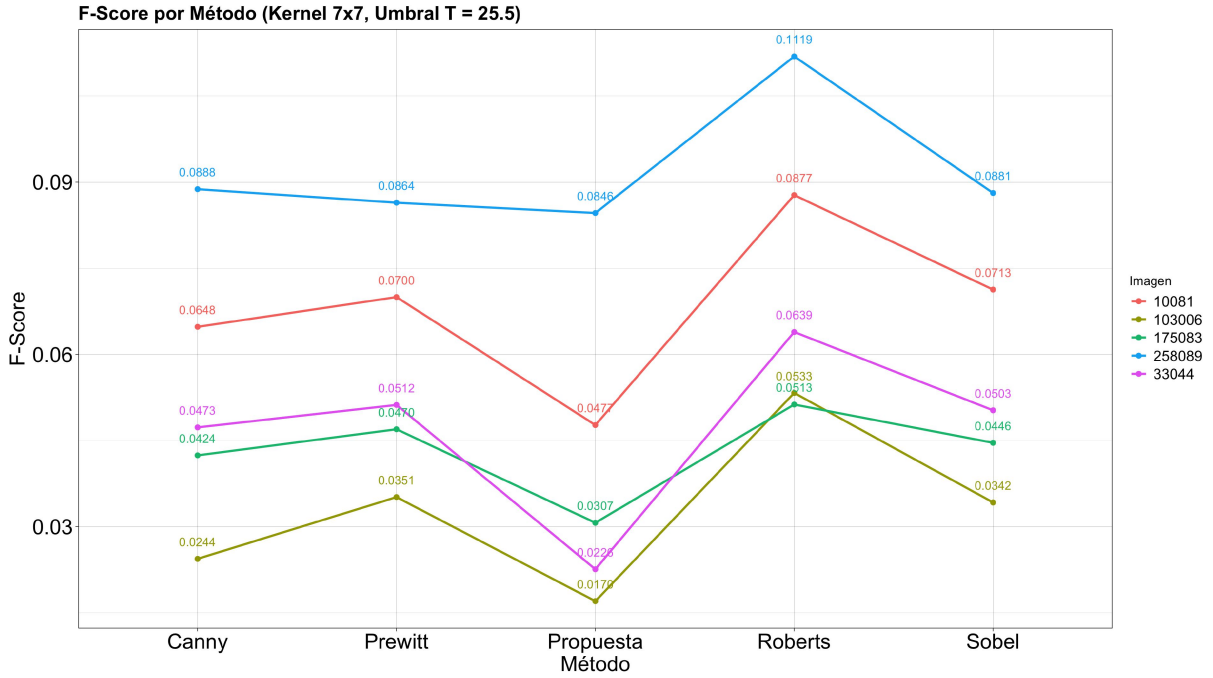


Figura 5.4: Gráfica comparativa del cálculo del F-Score en cada imagen. Con umbral de 25.5 y tamaño de kernel de 7×7.

Se aplicó el mismo procedimiento a la lista de imágenes añadiendo ruido de sal y pimienta con una densidad del 5 %. Los resultados de la detección de bordes se muestran en la figura (5.5).

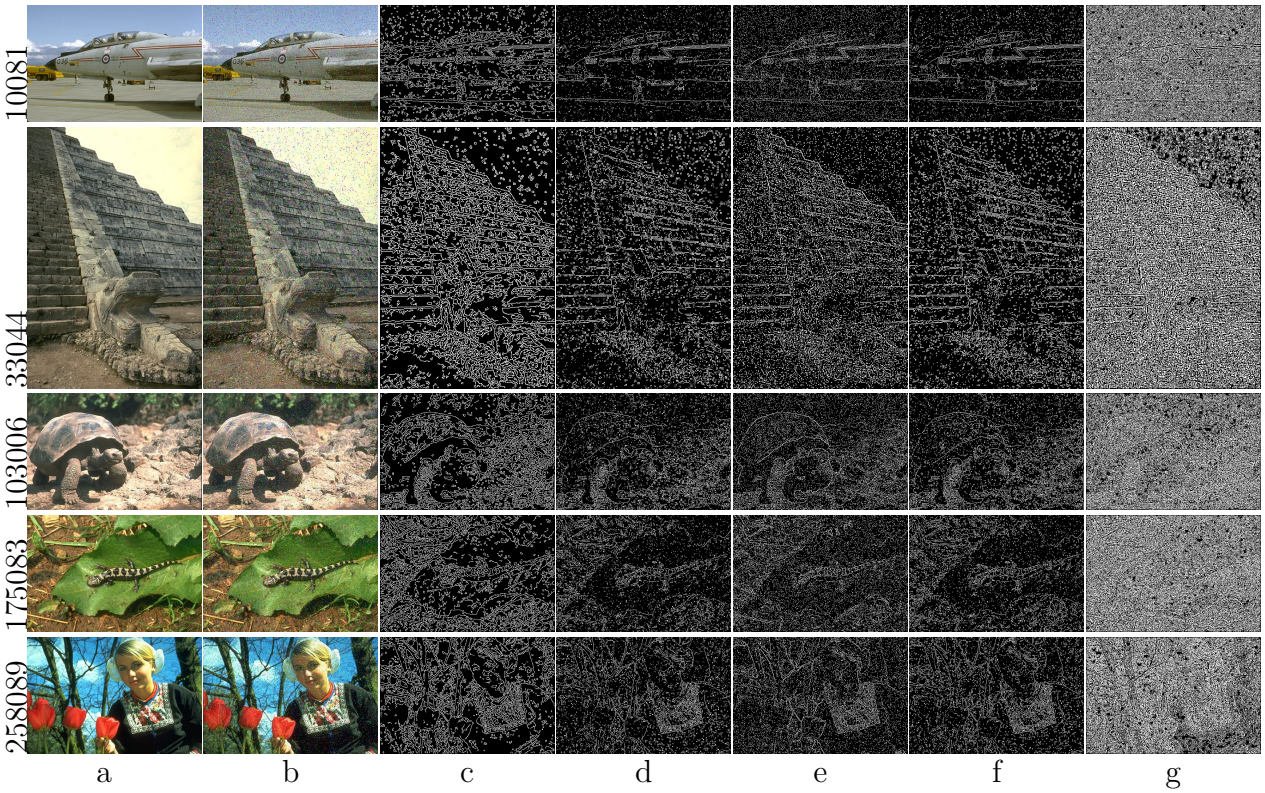


Figura 5.5: Resultado de la detección de bordes, (a) imagen original, (b) imagen con ruido de sal y pimienta con densidad de 5 %, (c) bordes producidos por Canny, (d) bordes producidos por Prewitt, (e) bordes producidos por Roberts, (f) bordes producidos por Sobel, (g) bordes producidos por el algoritmo propuesto con umbral de 25.5 y tamaño de kernel 3×3.

Se presentan las gráficas que muestran los resultados del cálculo del F-score para cada imagen con ruido de sal y pimienta con densidad de 5 % utilizando diferentes tamaños de kernel en las figuras (5.2, 5.3, 5.4).

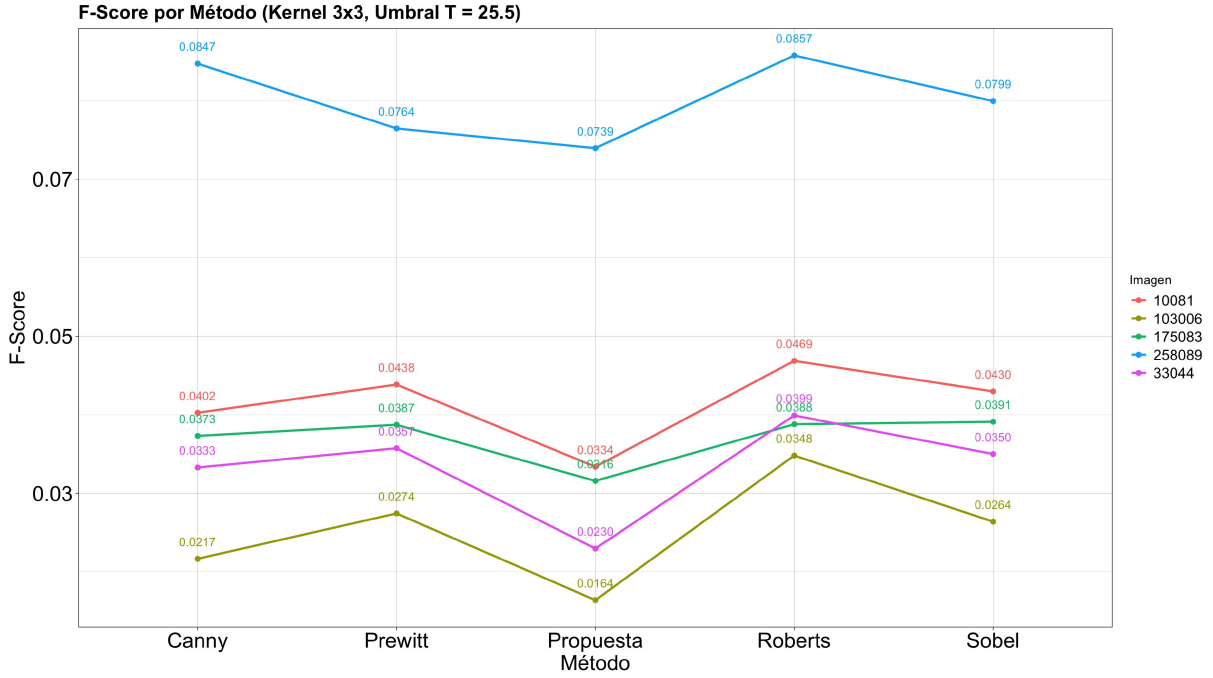


Figura 5.6: Gráfica comparativa del cálculo del F-Score en cada imagen con ruido de sal y pimienta con densidad de 5 %, con umbral de 25.5 y tamaño de kernel de 3×3.

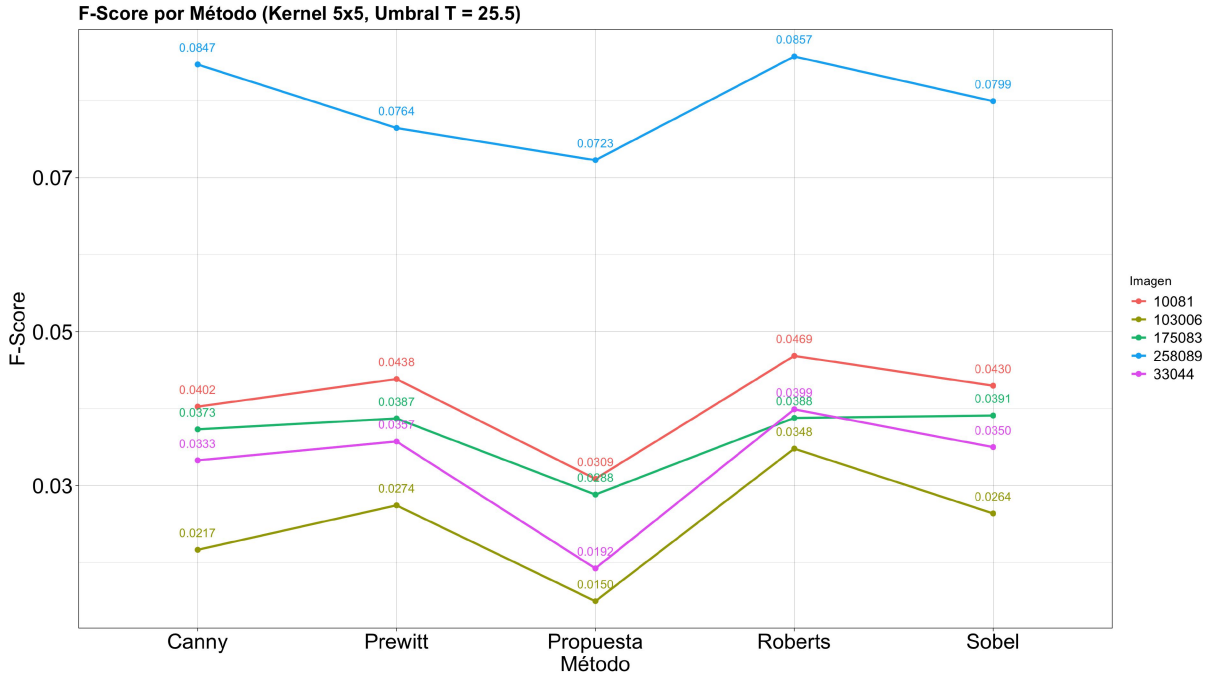


Figura 5.7: Gráfica comparativa del cálculo del F-Score en cada imagen con ruido de sal y pimienta con densidad de 5 %, umbral de 25.5 y tamaño de kernel de 5×5.

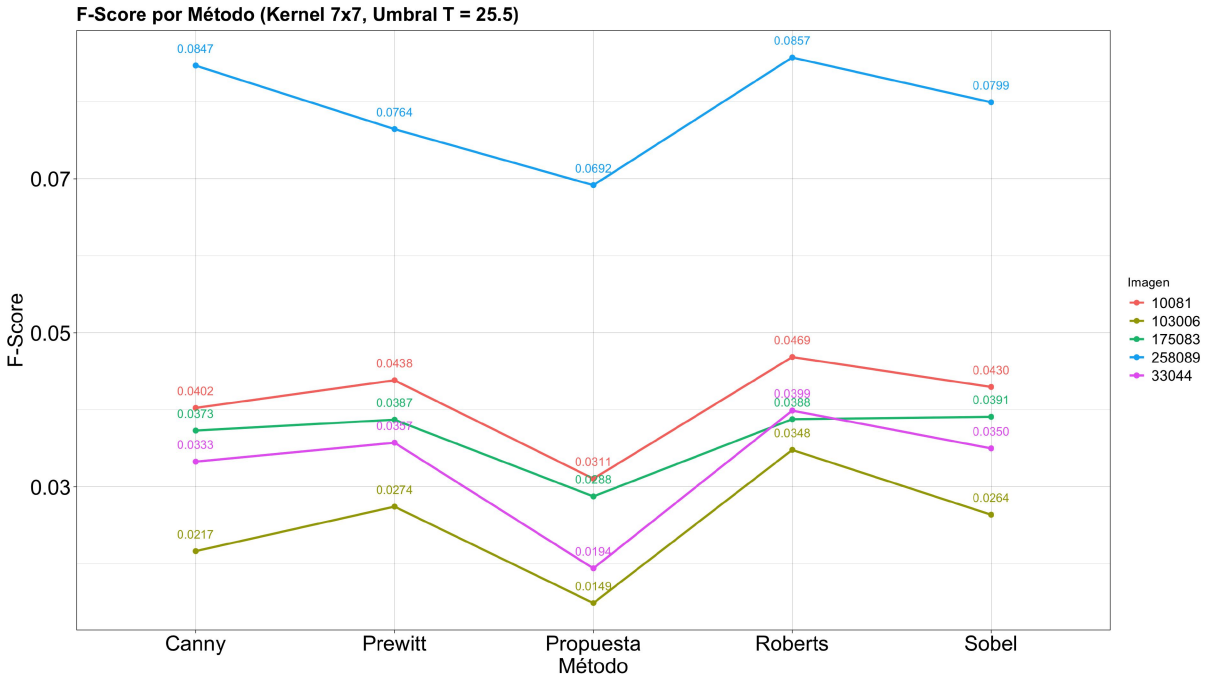
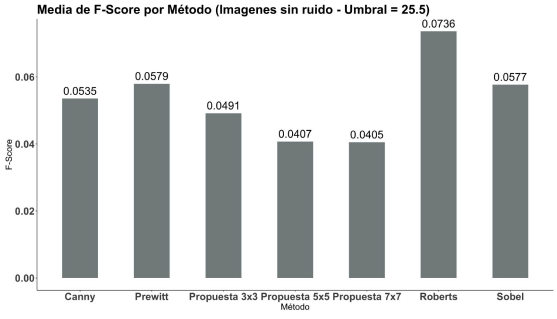
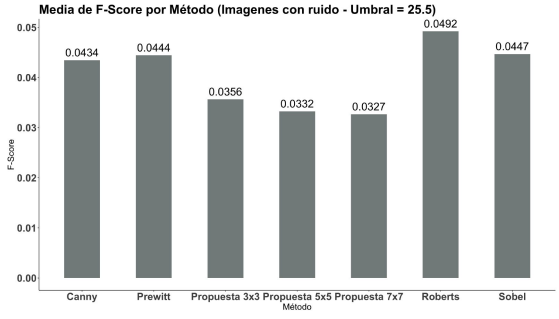


Figura 5.8: Gráfica comparativa del cálculo del F-Score en cada imagen con ruido de sal y pimienta con densidad de 5 %, con umbral de 25.5 y tamaño de kernel de 7×7.

Se calculó la media del F-score para imágenes sin ruido y para imágenes con ruido aplicado y se presentan en la gráfica (5.9), las gráficas indican que la propuesta no supera a los métodos clásicos de la literatura. Además, se observa que al incrementar el tamaño del kernel se reduce su rendimiento en las imágenes seleccionadas de la base de datos con anotaciones humanas.



(a) Medias de F-Score por método en imágenes sin ruido



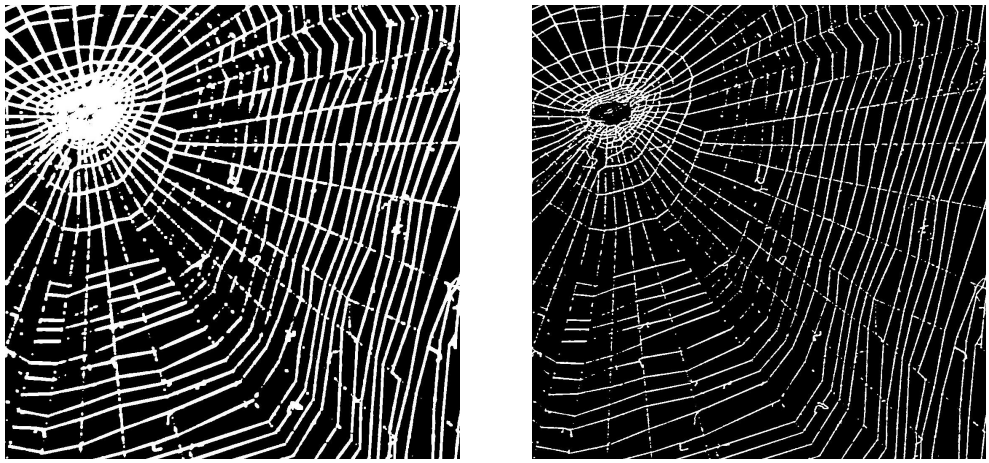
(b) Medias de F-Score por método en imágenes con ruido

Figura 5.9: Las medias del cálculo del F-Score en las imágenes sin ruido (a) y con ruido (b) indican que la propuesta en sus diferentes versiones según el tamaño del kernel, no supera a los métodos clásicos de la literatura.

5.0.2. Comparación entre la propuesta y métodos de umbralización local.

Las observaciones del experto indicaron que el método de segmentación local con el algoritmo de Niblack se obtiene con más claridad debido a la anchura de la segmentación pero se le indicó al experto que si bien la segmentación es más clara no corresponde a los valores reales de la telaraña. La parametrización del algoritmo, mediante métodos clásicos de la literatura o utilizando los propuestos en este trabajo puede determinar la variación del área de segmentación, por ejemplo, el tamaño del kernel puede aumentar o disminuir el área segmentada.

Estas variaciones pueden implicar una mayor o menor aproximación al área real de la telaraña, sin embargo, dicha evaluación debe basarse en un análisis visual realizado por el investigador. La figura (5.10) ilustra la diferencia en la anchura de la segmentación al comparar ambos algoritmos, utilizando los datos presentados en (Anexos C y D).



(a) Zoom de 300 % en imagen segmentada con el algoritmo de Niblack

(b) Zoom de 300 % en imagen segmentada con el algoritmo de Feng

Figura 5.10: El resultado final de ambas segmentaciones evidencia una diferencia en la claridad, observándose una mayor anchura en la segmentación generada por el algoritmo de Niblack con un total de 349,796 píxeles (blancos), en comparación con la obtenida mediante el algoritmo de Feng que presenta 155,756 píxeles (blancos). Ambas mediciones corresponden a la zona aumentada al 300 %.

Mientras que el algoritmo de Feng es el más aproximado a una correcta segmentación de la telaraña por ello se tomaron como referencia los resultados de estos métodos para realizar las métricas.

Las imágenes de los métodos analizados fueron reducidas a una resolución de 1024 por 1024 píxeles debido a que en el modelo entrenado en U-Net se utilizó esa resolución por las limitaciones de memoria en el equipo utilizado.

Para la evaluación de los resultados se obtuvieron las calificaciones del experto de las tablas de evaluación (Anexos E y F) y se sumaron las preguntas Q1 a Q5 y se restó el valor

de la pregunta Q6 debido a que esta pregunta corresponde a una calificación negativa en la imagen segmentada.

Se presentan los valores de promedio, mediana, desviación estándar y moda para ambos métodos locales y se seleccionaron aquellas imágenes que tuvieron un resultado de calificación total mayor o igual al promedio de las tablas de evaluación mencionadas anteriormente.

Promedio de evaluación de segmentación por el experto	Mediana	Std	Moda
18.4871	18	4.9250	18

Tabla 5.2: Tabla de estadísticas algoritmo de Niblack

Promedio de evaluación de segmentación por el experto	Mediana	Std	Moda
18.3589	18	4.5218	18

Tabla 5.3: Tabla de estadísticas algoritmo de Feng

Tomando el conjunto de las catorce imágenes válidas del algoritmo de Niblack se procedió a calcular las matrices de confusión para cada método con base en la binarización de los píxeles en las imágenes.

La tabla (5.4) presenta los resultados del algoritmo propuesto bajo distintas configuraciones de parámetros. En cada experimento, se estableció como condición de paro 10 generaciones, utilizando una población inicial de 100 individuos. Las variaciones entre pruebas se enfocaron en el tamaño del kernel y el valor del umbral para obtener las imágenes segmentadas, donde S es la condición de paro, P es la población inicial, K es el tamaño del kernel y T es el valor del umbral.

S	**P**	**K**	**T**	**Precisión**	**Sensibilidad**	**F-Score**	**VP**	**FP**	**FN**	**VN**
10	100	3	10	0.6658	0.8806	0.7583	2 246 399	1 127 538	304 579	11 001 548
10	100	3	20	0.8648	0.7112	0.7805	1 814 148	283 670	736 830	11 845 416
10	100	3	30	0.9455	0.5036	0.6572	1 284 688	74 032	1 266 290	12 055 054
10	100	5	10	0.4659	0.8424	0.6000	2 148 941	2 463 651	402 037	9 665 435
10	100	5	20	0.6721	0.8933	0.7671	2 278 670	1 111 475	272 308	11 017 611
10	**100**	**5**	**30**	**0.7462**	**0.8657**	**0.8015**	**2 208 393**	**751 267**	**342 585**	**11 377 819**
10	100	7	10	0.4171	0.7681	0.5406	1 959 386	2 738 206	591 592	9 390 880
10	100	7	20	0.6765	0.9102	0.7762	2 322 018	1 110 217	228 960	11 018 869
10	100	7	30	0.7242	0.8676	0.7895	2 213 175	842 651	337 803	11 286 435

Tabla 5.4: Comparativa del rendimiento del algoritmo propuesto basado en los resultados de la segmentación con el algoritmo de Niblack.

La tabla anterior indica que la mejor configuración corresponde a un tamaño de kernel de 5×5 y un valor de umbral de 30. Aunque este ajuste podría considerarse aceptable por el experto, su desempeño sugiere que las variaciones en otros parámetros podrían ofrecer configuraciones mejores en pruebas futuras.

	Área de telaraña	Fondo
Área de telaraña (Positivo)	2 208 393 (VP)	342 585 (FN)
Fondo (Negativo)	751 267 (FP)	11 377 819 (VN)

Tabla 5.5: Matriz de Confusión Método 1: Propuesta

	Área de telaraña	Fondo
Área de telaraña (Positivo)	1 213 616 (VP)	1 337 362 (FN)
Fondo (Negativo)	18 503 (FP)	12 110 583 (VN)

Tabla 5.6: Matriz de Confusión Método 2: Feng

	Área de telaraña	Fondo
Área de telaraña (Positivo)	2 380 235 (VP)	170 743 (FN)
Fondo (Negativo)	382 319 (FP)	11 746 767 (VN)

Tabla 5.7: Matriz de Confusión Método 3: U-net

Método	Precisión	Sensibilidad	F-score
Propuesto	**74.61** %	**86.57** %	**80.15** %
Feng	**98.49** %	**47.57** %	**64.15** %
Predicción U-net (Niblack)	**86.16** %	**93.30** %	**89.59** %

Tabla 5.8: Comparación de métricas de los metodos analizados

Tomando el conjunto de las doce imágenes válidas del algoritmo de Feng se realizó el mismo procedimiento para el cálculo de las métricas que se muestran en la tabla (5.9).

S	**P**	**K**	**T**	**Precisión**	**Sensibilidad**	**F-Score**	**VP**	**FP**	**FN**	**VN**
10	100	3	10	0.3102	0.9759	0.4708	892 860	1 985 273	22 095	9 682 684
10	100	3	20	0.4695	0.8889	0.6144	813 298	919 009	101 657	10 748 948
10	**100**	**3**	**30**	**0.5481**	**0.8204**	**0.6572**	**750 611**	**618 860**	**164 344**	**11 049 097**
10	100	5	10	0.2221	0.9612	0.3608	879 415	3 079 776	35 540	8 588 181
10	100	5	20	0.3091	0.9602	0.4677	878 495	1 963 575	36 460	9 704 382
10	100	5	30	0.3474	0.9631	0.5106	881 234	1 655 429	33 721	10 012 528
10	100	7	10	0.2229	0.9531	0.3613	872 066	3 039 874	42 889	8 628 083
10	100	7	20	0.3054	0.9625	0.4637	880 616	2 002 947	34 339	9 665 010
10	100	7	30	0.3185	0.9242	0.4737	845 589	1 809 467	69 366	9 858 490

Tabla 5.9: Comparativa del rendimiento del algoritmo propuesto basado en los resultados de la segmentación con el algoritmo de Feng.

La tabla anterior indica que la mejor configuración corresponde a un tamaño de kernel de 3×3 y un valor de umbral de 30. Su desempeño no es muy óptimo, el ajuste de los parámetros S y P podría mejorar en gran medida el desempeño del algoritmo.

	Área de telaraña	Fondo
Área de telaraña (Positivo)	750 611 (VP)	164 344 (FN)
Fondo (Negativo)	618 860 (FP)	11 049 097 (VN)

Tabla 5.10: Matriz de Confusión Método 1: Propuesta

	Área de telaraña	Fondo
Área de telaraña (Positivo)	877 913 (VP)	37 042 (FN)
Fondo (Negativo)	1 653 992 (FP)	10 013 965 (VN)

Tabla 5.11: Matriz de Confusión Método 2: Niblack

	Área de telaraña	Fondo
Área de telaraña (Positivo)	844 970 (VP)	69 985 (FN)
Fondo (Negativo)	98 717 (FP)	11 569 240 (VN)

Tabla 5.12: Matriz de Confusión Método 3: Predicción U-net

Método	Precisión	Sensibilidad	F-score
Propuesto	**54.81** %	**82.03** %	**65.71** %
Niblack	**34.67** %	**95.95** %	**50.93** %
Predicción U-net (Feng)	**89.53** %	**92.35** %	**90.92** %

Tabla 5.13: Comparación de métricas de los métodos analizados

5.0.3. Discusión

Al analizar los datos mediante la obtención del F-Score utilizando imágenes seleccionadas de la base de datos BSDS500 y contrastando los resultados con los métodos clásicos de la literatura, se evidencia que el método propuesto no supera a los métodos tradicionales. Esto podría deberse a que la sensibilidad en la selección de parámetros desempeña un papel crucial en la detección de los bordes reales de la imagen y a que la propuesta resulta demasiado susceptible al ruido tipo 'sal y pimienta', lo cual afecta negativamente su rendimiento. El parámetro que se considera tiene mayor impacto es sin duda la selección del umbral óptimo, el cual depende significativamente del análisis visual realizado por una persona para examinar el resultado como del tipo de imagen analizada.

La selección del umbral óptimo no es una tarea sencilla, ya que los bordes obtenidos pueden resultar poco precisos y no representar adecuadamente las regiones de interés para su estudio. Por ello, se podría sugerir el uso de un método automático que permita identificar los umbrales más adecuados en cada imagen, con el fin de maximizar la detección de los bordes reales.

Respecto a la comparación con los métodos de detección de bordes locales descritos en la literatura tomando el conjunto de imágenes válidas con segmentación utilizando el

algoritmo de Niblack las métricas muestran que el método propuesto puede ser comparable hasta cierto punto debido a que los segmentos detectados en la telaraña son más gruesos a comparación del algoritmo de Feng, los resultados de segmentación con este algoritmo son más finos y con contornos más definidos.

Si tomamos las métricas del conjunto de imágenes segmentadas con el algoritmo de Feng claramente el método propuesto es muy inferior, la segmentación usando Feng es más fina y cercana a los límites reales de la telaraña, el método propuesto tendría que ser modificado en su elemento estructurante para obtener detalles más finos.

Sé puede concluir que la integración de técnicas de cómputo evolutivo y visión por computadora en la propuesta se considera poco práctica, debido a la elevada cantidad de parámetros que deben optimizarse y a la complejidad computacional que implica. Además, los análisis basados en las métricas empleadas refutan la hipótesis inicial.

La exploración de otros métodos clásicos, combinada con la implementación de modificaciones y optimizaciones, podría ser una buena estrategia para mejorar los resultados obtenidos. Esto permitiría evaluar el desempeño de diferentes técnicas, optimizando los parámetros y adaptando los algoritmos a las imágenes analizadas.

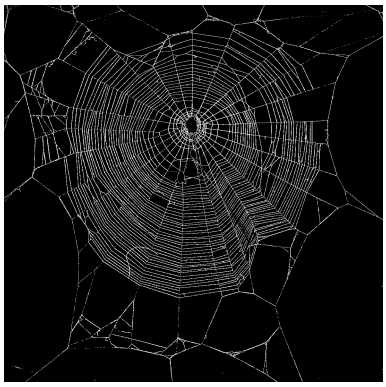
5.0.4. Medición del daño estructural

Para el cálculo del daño estructural se seleccionó un par de imágenes previamente segmentadas con el algoritmo de Feng. En particular, se utilizó la imagen “18-02-AL-CA“, que representa la telaraña con daño estructural, junto con la imagen sin daño. Se llevó a cabo un registro de ambas imágenes y posteriormente se procedió a cuantificar el daño mediante el conteo de las diferencias en los píxeles entre la imagen dañada y la imagen no dañada.

$$\text{Porcentaje de Daño} = \left(\frac{\text{Número de Píxeles Diferentes}}{\text{Píxeles Totales de la Imagen sin Daño}} \right) \times 100 \tag{5.2}$$

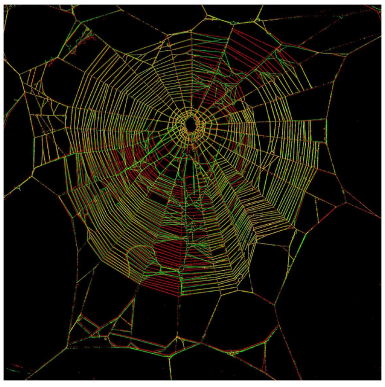
Descripción	**Valor**
Pixeles totales (imagen sin daño)	10,840,484
El número de píxeles diferentes	1,384,503
El porcentaje de daño estructural	12.77 %

Tabla 5.14: Datos de daño estructural en la imagen



(a) 18-02-AL-CA-antes

(b) 18-02-AL-CA-despues



(c) Superposición de imagen (antes y después) el color rojo indica la imagen con el daño estructural después de hacer el registro de imágenes.

Figura 5.11: Segmentación de la imagen “18-02-AL-CA“ junto con el registro de imágenes para el cálculo del daño estructural. El color rojo representa la imagen sin daño, mientras que el color verde representa la imagen con daño y el color amarillo representa las regiones en las que ambas imágenes coinciden espacialmente.

5.0.5. Aplicación de usuario basada en ImageJ

ImageJ es un software de dominio público para el procesamiento y análisis científico de imágenes, este software está basado en el trabajo de Wayne Rasband durante su trabajo en el National Institutes of Health (NHI) (Rasband, s.f.).

ImageJ contiene una documentación y plug-ins y clases reutilizables que permiten desarrollar software especifico según las necesidades del problema a tratar, así como funciones y métodos para el procesamiento de imágenes que pueden ser aplicadas a distintas áreas (medicina, biología, astronomía, etc.), la figura (5.12) muestra la interfaz gráfica de usuario de imageJ.

Figura 5.12: Interfaz imageJ muestra el menú de opciones para aplicar procesamiento de imágenes.

ImageJ está dividido en paquetes de Java y cuenta con una API (Application Programming Interface) para su uso como herramienta de desarrollo. Cuenta también con un repositorio en Github que contiene el código fuente y las últimas actualizaciones hechas por los desarrolladores.

Se desarrolló una aplicación utilizando la API de ImageJ programada en Java, esta aplicación cuenta con una interfaz gráfica de usuario para facilitar el proceso de umbralización y segmentación de las áreas de interés para las imágenes de telarañas, cuenta con las opciones de umbralización local descritas en (4.7). Contiene la opción de selección de imagen para procesarla, selección de parámetros según el tipo de algoritmo y una consola de salida de las acciones que el usuario procesa durante la ejecución del programa. El motivo del desarrollo de esta aplicación consiste en que el experto tenga una herramienta visual que le permita mediante prueba y error generar imágenes segmentadas, así como la información de los parámetros seleccionados y que puedan servir de ejemplos para poder entrenar un modelo de red neuronal (4.8), la figura (5.13) muestra el menú principal para el usuario.

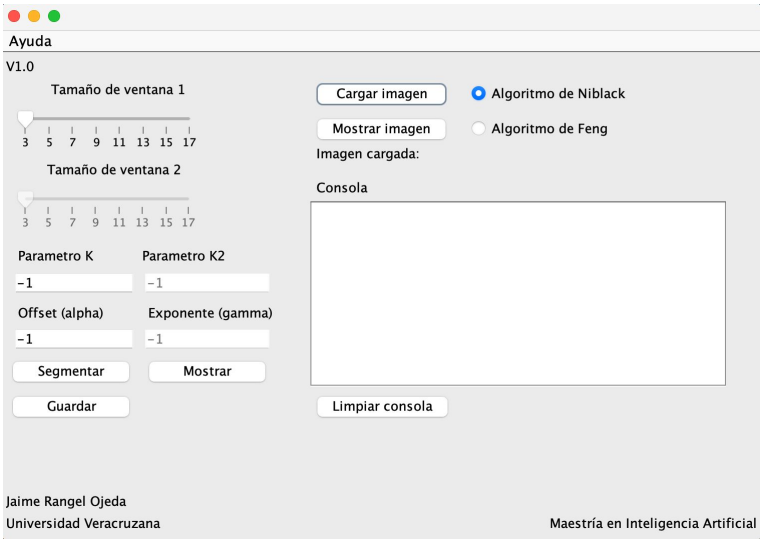


Figura 5.13: Interfaz de usuario desarrollada para el experto, con esta aplicación se plantea segmentar imágenes de telarañas de forma manual y observacional bajo su criterio.

Lectura de imágenes

Al seleccionar el botón "Cargar imagen" permite seleccionar una imagen en formato JPG, PNG, etc. La imagen se mostrara al usuario en una segunda ventana y un texto en consola con el nombre de la imagen que se agregó al programa. También incluye el botón de "Mostrar imagen" en caso de que el usuario cierre la ventana de la imagen cargada y dese visualizarla de nuevo. La figura (5.14) muestra el menú de usuario y una ventana donde se visualiza la imagen seleccionada.

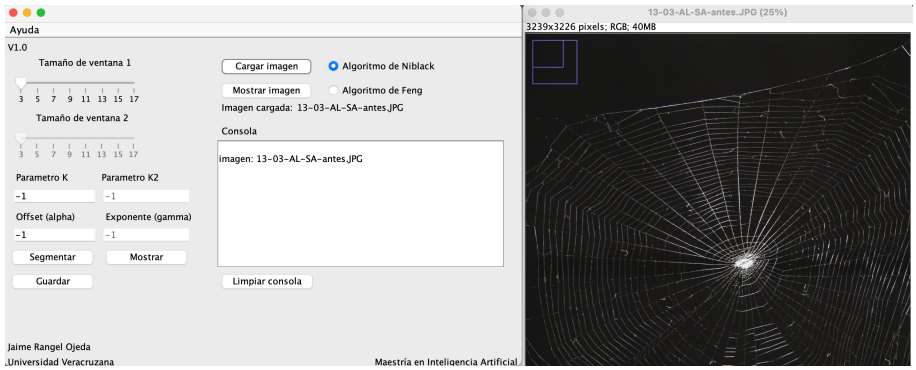


Figura 5.14: Menú de usuario y la imagen cargada visualizada.

Segmentación mediante el algoritmo de Niblack

El algoritmo de Niblack necesita establecer tres parámetros (Tamaño de ventana 1, Parámetro K y valor de Offset) una vez seleccionados los parámetros y con la imagen cargada al programa se puede iniciar la segmentación seleccionando el botón "Segmentar",

en el programa la consola muestra la información de la segmentación. La figura (5.15) muestra el menú de usuario y una ventana con la imagen segmentada.

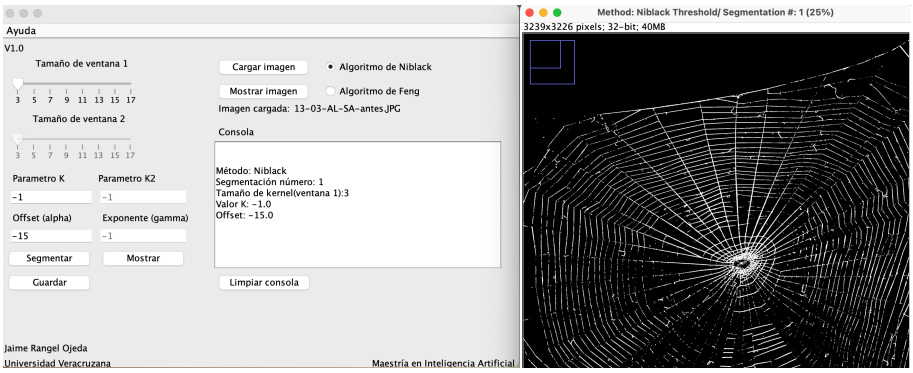


Figura 5.15: Menú de usuario y la imagen segmentada utilizando el método de Niblack.

Segmentación mediante el algoritmo de Feng

El algoritmo de Feng necesita establecer seis parámetros (Tamaño de ventana 1, tamaño de ventana 2, parámetro K_1, parametro K_2, valor de Offset, y exponente) una vez seleccionados los parámetros y con la imagen cargada al programa se puede iniciar la segmentación seleccionando el botón "Segmentar", en el programa la consola muestra la información de la segmentación. La figura (5.16) muestra el menú de usuario y una ventana con la imagen segmentada.

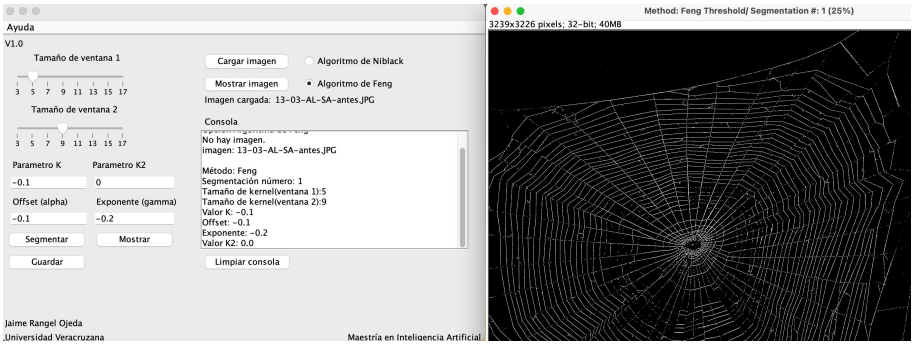


Figura 5.16: Menú de usuario y la imagen segmentada utilizando el método de Feng.

Salida de datos en consola

Se le proporciona al usuario la información de los parámetros seleccionados como referencia en cada segmentación, este es un historial de parámetros que pueden ser utilizados para mejorar la imagen segmentada por parte del experto y calibrar mejor estos parámetros, el cuadro (5.17) muestra un ejemplo de salida de consola en el programa después de segmentar la imagen.

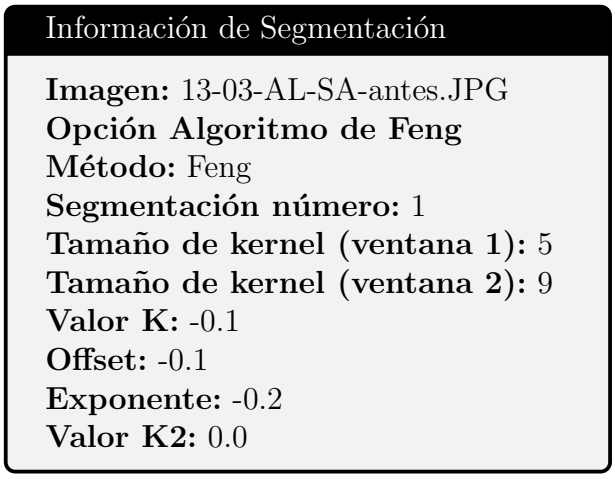


Figura 5.17: Información de segmentación de la imagen 13-03-AL-SA-antes.JPG, esta información se puede utilizar como referencia para calibrar los parámetros en cada segmentación

Almacenamiento de imágenes segmentadas

Basado en el criterio del experto sobre el resultado de la segmentación con los parámetros calibrados tiene la opción de almacenar la imagen resultante presionando el botón "Guardar", el programa mostrará una ventana para la selección del directorio, al finalizar la selección del directorio se almacenará la imagen segmentada en un archivo de tipo $.JPG$ y un archivo de texto en formato *.txt* que contiene los parámetros que fueron utilizados para dicha segmentación.

Capítulo 6

Trabajo futuro

Los resultados muestran que si bien el método propuesto no obtiene un buen porcentaje en las métricas basadas en lo que el experto evaluó en cada método local, pueden obtenerse mejores resultados al configurar el algoritmo para obtener características de cada nivel de resolución que se plantea al momento de realizar la segmentación (4.6.13) y ajustando el tamaño del elemento estructurante, así como el tamaño del kernel, el tamaño de población, generaciones y umbral.

Aunque las desventajas de usar el método propuesto incluyen el costo computacional y tiempo de ejecución debido a las operaciones de convolución y generaciones utilizando computo evolutivo a comparación de los métodos locales analizados y la simplicidad de su uso al momento de seleccionar los parámetros para realizar la segmentación.

Otro punto a tener en consideración es que los resultados deben ser validados en cada prueba y observar las características de la imagen es decir si la telaraña es segmentada de forma correcta sin ruido y delineando los contornos reales de la misma por ello puede explorarse esta forma de segmentar al realizar estas modificaciones al algoritmo propuesto.

En cuanto a la alternativa de usar métodos locales de la literatura (Niblack, 1986, Feng y Tan, 2004) el trabajo consiste en selección manual de parámetros muy observacional con ayuda de un experto y llevando un registro de parámetros.

Por esta razón, se desarrolló un programa con una interfaz gráfica (5.0.5) que permite al investigador llevar a cabo este proceso y obtener las imágenes segmentadas según su criterio, mediante el ajuste de los parámetros correspondientes.

Sería útil realizar un análisis de datos para investigar la correlación entre los parámetros y el tipo de imagen a segmentar. Esto permitiría diseñar un proceso automatizado que elimine la necesidad de un cálculo manual al seleccionar los parámetros óptimos para la segmentación.

Finalmente sí se realiza esta modificación para selección de parámetros observando la correlación de la calidad de segmentación en las imágenes utilizando los métodos locales para obtener ejemplos y entrenar un modelo basado en U-Net. Un proceso adicional para mejorar el entrenamiento del modelo el cual no se consideró en este trabajo es el aumento de datos que puede mejorar significativamente los porcentajes de las métricas analizadas.

Parte I

Anexos

Apéndice A

Base de datos

Antes	**Después**
13-03-AL-CA-antes	13-03-AL-CA-despues
13-03-S-SA-antes	13-03-S-SA-despues
17-03-S-SA-antes	17-03-S-SA-despues
18-02-S-SA-antes	18-02-S-SA-despues
13-03-AL-SA-antes	13-03-AL-SA-despues
17-03-AL-CA-antes	17-03-AL-CA-despues
18-02-AL-CA-antes	18-02-AL-CA-despues
20-02-AL-SA-antes	20-02-AL-SA-despues
13-03-IL-CA-antes	13-03-IL-CA-despues
17-03-AL-SA-antes	17-03-AL-SA-despues
18-02-AL-SA-antes	18-02-AL-SA-despues
20-02-IL-CA-antes	20-02-IL-CA-despues
13-03-IL-SA-antes	13-03-IL-SA-despues
17-03-IL-CA-antes	17-03-IL-CA-despues
18-02-IL-CA-antes	18-02-IL-CA-despues
20-02-IL-SA-antes	20-02-IL-SA-despues
13-03-S-CA-antes	13-03-S-CA-despues
17-03-S-CA-antes	17-03-S-CA-despues
18-02-IL-SA-antes	18-02-IL-SA-despues
20-02-S-SA-antes	20-02-S-SA-despues

Tabla A.1: Lista de imágenes antes y después

Apéndice B

Tabla de evaluación de imágenes segmentadas

Cuestionario de Evaluación de Segmentación de Imágenes de Telarañas
Instrucciones: Por favor, valore del 1 al 5 (donde 1 es la calificación más baja y 5 es la calificación más alta) los siguientes aspectos relacionados con los resultados de la segmentación en las imágenes de telarañas.

Aspecto	**Calificación (1-5)**
Q1-Completitud de la Segmentación: ¿Se ha logrado segmentar completamente la telaraña presente en la imagen?	
Q2-Claridad de la Telaraña Segmentada: ¿Qué tan claramente se puede distinguir la telaraña en la imagen segmentada?	
Q3-Definición de Contornos: ¿Los contornos de la telaraña están bien definidos y coinciden con los límites reales?	
Q4-Consistencia de la Segmentación: ¿La segmentación es consistente en la imagen?	
Q5-Facilidad de Interpretación: ¿Es fácil de interpretar y comprender la imagen segmentada?	
Q6-Ruido: Juzgar el ruido o error en la segmentación de la telaraña	

Apéndice C

Tablas parámetros Niblack

Nombre de la imagen	**Parámetros Niblack**
13-03-AL-CA-antes	$[5,5], -1, -15$, "*replicate*"
13-03-AL-SA-antes	$[5,5], -4, -25$, "*replicate*"
13-03-IL-CA-antes	$[7,7], -2, -10$, "*replicate*"
13-03-IL-SA-antes	$[7,7], -1, -12$, "*replicate*"
13-03-S-CA-antes	$[9,9], -1, -10$, "*replicate*"
13-03-S-SA-antes	$[3,3], -1, -9$, "*replicate*"
17-03-AL-CA-antes	$[5,5], -1, -10$, "*replicate*"
17-03-AL-SA-antes	$[7,7], -1, -10$, "*replicate*"
17-03-IL-CA-antes	$[7,7], -1, -10$, "*replicate*"
17-03-S-CA-antes	$[7,7], -1, -10$, "*replicate*"
17-03-S-SA-antes	$[7,7], -2, -10$, "*replicate*"
18-02-AL-CA-antes	$[7,7], -1, -10$, "*replicate*"
18-02-AL-SA-antes	$[7,7], -3, -14$, "*replicate*"
18-02-IL-CA-antes	$[5,5], -2, -12$, "*replicate*"
18-02-IL-SA-antes	$[5,5], -1, -10$, "*replicate*"
18-02-S-SA-antes	$[5,5], -1, -12$, "*replicate*"
20-02-AL-SA-antes	$[5,5], -2, -7$, "*replicate*"
20-02-IL-CA-antes	$[5,5], -1, -8$, "*replicate*"
20-02-IL-SA-antes	$[5,5], -1, -8$, "*replicate*"
20-02-S-SA-antes	$[5,5], -1, -8$, "*replicate*"

Tabla C.1: Parámetros Niblack para cada imagen (Antes)

Nombre de Imagen	**Parámetros Niblack**
13-03-AL-CA-despues	$[5,5], -1, -7$, "*replicate*"
13-03-AL-SA-despues	$[9,9], -5, -16$, "*replicate*"
13-03-IL-CA-despues	$[9,9], -5, -20$, "*replicate*"
13-03-IL-SA-despues	$[7,7], -1, -6$, "*replicate*"
13-03-S-CA-despues	$[9,9], -1, -15$, "*replicate*"
13-03-S-SA-despues	$[5,5], -2, -10$, "*replicate*"
17-03-AL-CA-despues	$[5,5], -3, -15$, "*replicate*"
17-03-AL-SA-despues	$[7,7], -1, -11$, "*replicate*"
17-03-IL-CA-despues	$[9,9], -2, -7$, "*replicate*"
17-03-S-CA-despues	$[5,5], -3, -15$, "*replicate*"
17-03-S-SA-despues	$[5,5], -2, -9$, "*replicate*"
18-02-AL-CA-despues	$[7,7], -1, -8$, "*replicate*"
18-02-AL-SA-despues	$[7,7], -1, -6$, "*replicate*"
18-02-IL-CA-despues	$[7,7], -1, -9$, "*replicate*"
18-02-IL-SA-despues	$[7,7], -1, -10$, "*replicate*"
18-02-S-SA-despues	$[7,7], -2, -8$, "*replicate*"
20-02-AL-SA-despues	$[7,7], -1, -12$, "*replicate*"
20-02-IL-CA-despues	$[5,5], -1, -10$, "*replicate*"
20-02-IL-SA-despues	$[5,5], -1, -10$, "*replicate*"
20-02-S-SA-despues	$[3,3], -1, -5$, "*replicate*"

Tabla C.2: Parámetros Niblack para cada imagen (Después)

Apéndice D

Tablas parámetros Feng

Nombre de la imagen	**Parámetros Feng**
13-03-AL-CA-antes	[5, 5], [11, 11], -0.1, -0.1, -0.1, 0, “replicate“
13-03-AL-SA-antes	[9, 9], [11, 11], -0.5, -0.5, -1.5, 0, “replicate“
13-03-IL-CA-antes	[9, 9], [11, 11], -0.3, -0.5, -1, 0, “replicate“
13-03-IL-SA-antes	[7, 7], [15, 15], -0.3, -0.5, -1, 0, “replicate“
13-03-S-CA-antes	[5, 5], [11, 11], -0.1, -0.2, -0.1, 0, “replicate“
13-03-S-SA-antes	[5, 5], [11, 11], -0.2, -0.3, -0.5, 0, “replicate“
17-03-AL-CA-antes	[7, 7], [11, 11], -0.4, -0.5, -1, 0, “replicate“
17-03-AL-SA-antes	[7, 7], [11, 11], -0.4, -0.5, -1, 0, “replicate“
17-03-IL-CA-antes	[7, 7], [11, 11], -0.3, -0.8, -1, 0, “replicate“
17-03-S-CA-antes	[7, 7], [11, 11], -0.5, -0.5, -1, 0, “replicate“
17-03-S-SA-antes	[5, 5], [13, 13], -0.3, -0.4, -1, 0, “replicate“
18-02-AL-CA-antes	[5, 5], [13, 13], -0.3, -0.4, -1, 0, “replicate“
18-02-AL-SA-antes	[7, 7], [13, 13], -0.3, -0.5, -1, 0, “replicate“
18-02-IL-CA-antes	[7, 7], [13, 13], -0.3, -0.5, -1, 0, “replicate“
18-02-IL-SA-antes	[7, 7], [13, 13], -0.5, -0.4, -1, 0, “replicate“
18-02-S-SA-antes	[7, 7], [13, 13], -0.5, -0.4, -1, 0, “replicate“
20-02-AL-SA-antes	[5, 5], [13, 13], -0.4, -0.5, -1.2, 0, “replicate“
20-02-IL-CA-antes	[7, 7], [15, 15], -0.4, -0.3, -1, 0, “replicate“
20-02-IL-SA-antes	[7, 7], [11, 11], -0.4, -0.5, -1.2, 0, “replicate“
20-02-S-SA-antes	[7, 7], [11, 11], -0.3, -0.5, -1.2, 0, “replicate“

Tabla D.1: Parámetros Feng para cada imagen (Antes)

Nombre de la imagen	**Parámetros Feng**
13-03-AL-CA-despues	[5, 5], [9, 9], -0.1, -0.2, -0.1, 0, “replicate“
13-03-AL-SA-despues	[11, 11], [13, 13], -0.2, -0.2, -0.6, 0, “replicate“
13-03-IL-CA-despues	[9, 9], [13, 13], -0.2, -0.3, -0.5, 0, “replicate“
13-03-IL-SA-despues	[7, 7], [11, 11], -0.2, -0.3, -0.4, 0, “replicate“
13-03-S-CA-despues	[5, 5], [11, 11], -0.2, -0.1, -0.3, 0, “replicate“
13-03-S-SA-despues	[5, 5], [7, 7], -0.1, -0.2, -0.1, 0, “replicate“
17-03-AL-CA-despues	[9, 9], [11, 11], -0.3, -0.3, -0.3, 0, “replicate“
17-03-AL-SA-despues	[9, 9], [11, 11], -0.2, -0.3, -0.5, 0, “replicate“
17-03-IL-CA-despues	[7, 7], [13, 13], -0.3, -0.6, -0.5, 0, “replicate“
17-03-S-CA-despues	[9, 9], [13, 13], -0.2, -0.1, -0.2, 0, “replicate“
17-03-S-SA-despues	[9, 9], [13, 13], -0.2, -0.1, -0.4, 0, “replicate“
18-02-AL-CA-despues	[9, 9], [13, 13], -0.2, -0.2, -0.3, 0, “replicate“
18-02-AL-SA-despues	[9, 9], [13, 13], -0.2, -0.1, -0.4, 0, “replicate“
18-02-IL-CA-despues	[9, 9], [13, 13], -0.2, -0.1, -0.5, 0, “replicate“
18-02-IL-SA-despues	[7, 7], [11, 11], -0.3, -0.1, -0.5, 0, “replicate“
18-02-S-SA-despues	[7, 7], [11, 11], -0.3, -0.1, -0.5, 0, “replicate“
20-02-AL-SA-despues	[7, 7], [11, 11], -0.3, -0.1, -0.7, 0, “replicate“
20-02-IL-CA-despues	[7, 7], [11, 11], -0.2, -0.1, -0.5, 0, “replicate“
20-02-IL-SA-despues	[7, 7], [11, 11], -0.2, -0.2, -0.5, 0, “replicate“
20-02-S-SA-despues	[7, 7], [11, 11], -0.2, -0.1, -0.5, 0, “replicate“

Tabla D.2: Parámetros Feng para cada imagen (Después)

Apéndice E

Tabla de calificaciones algoritmo de Niblack

Nombre	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Suma**	**Total**	**Valida**
17-03-AL-CA-antes	5	5	5	5	5	1	25	24	SI
17-03-S-SA-antes	5	5	5	5	5	1	25	24	SI
18-02-AL-CA-antes	5	5	5	5	5	1	25	24	SI
18-02-AL-CA-despues	5	5	5	5	5	1	25	24	SI
18-02-IL-CA-antes	5	5	5	5	5	1	25	24	SI
18-02-IL-CA-despues	5	5	5	5	5	1	25	24	SI
20-02-S-SA-antes	5	5	5	5	5	1	25	24	SI
20-02-S-SA-despues	5	5	5	5	5	1	25	24	SI
18-02-IL-SA-antes	5	5	5	5	5	2	25	23	SI
18-02-S-SA-antes	5	5	5	5	5	2	25	23	SI
20-02-AL-SA-antes	5	5	5	5	5	2	25	23	SI
20-02-IL-SA-antes	5	5	5	5	5	2	25	23	SI
20-02-IL-SA-despues	5	5	5	5	5	2	25	23	SI
20-02-AL-SA-despues	5	5	5	5	5	3	25	22	SI
13-03-IL-CA-antes	4	4	4	4	4	2	20	18	NO
13-03-IL-CA-despues	4	4	4	4	4	2	20	18	NO
13-03-IL-SA-despues	4	4	4	4	4	2	20	18	NO
17-03-AL-SA-antes	4	4	4	4	4	2	20	18	NO
17-03-S-CA-despues	4	4	4	4	4	2	20	18	NO
17-03-S-SA-despues	4	4	4	4	4	2	20	18	NO
18-02-AL-SA-antes	4	4	4	4	4	2	20	18	NO
18-02-AL-SA-despues	4	4	4	4	4	2	20	18	NO
18-02-S-CA-antes	4	4	4	4	4	2	20	18	NO
18-02-S-CA-despues	4	4	4	4	4	2	20	18	NO
18-02-S-SA-despues	4	4	4	4	4	2	20	18	NO
20-02-IL-CA-despues	4	4	4	4	4	2	20	18	NO
13-03-IL-SA-antes	4	4	4	4	4	3	20	17	NO
17-03-AL-CA-despues	4	4	4	4	4	3	20	17	NO
17-03-AL-SA-despues	4	4	4	4	4	3	20	17	NO
17-03-IL-CA-antes	4	4	4	4	4	3	20	17	NO
17-03-IL-CA-despues	4	4	4	4	4	3	20	17	NO
17-03-S-CA-antes	4	4	4	4	4	3	20	17	NO
20-02-IL-CA-antes	4	4	4	4	4	3	20	17	NO
13-03-AL-SA-despues	3	3	3	3	3	3	15	12	NO
13-03-S-CA-antes	3	3	3	3	3	3	15	12	NO
13-03-S-CA-despues	3	3	3	3	3	3	15	12	NO
13-03-AL-SA-antes	3	3	3	3	3	4	15	11	NO
13-03-AL-CA-antes	2	2	2	2	2	5	10	5	NO
13-03-S-SA-antes	2	2	2	2	2	5	10	5	NO

Tabla E.1: Tabla de evaluación de segmentación de imágenes para Niblack

Apéndice F

Tabla de calificaciones algoritmo de Feng

Nombre	**Q1**	**Q2**	**Q3**	**Q4**	**Q5**	**Q6**	**Suma**	**Total**	**Valida**
18-02-AL-CA-antes	5	5	5	5	5	0	25	25	SI
18-02-IL-CA-antes	5	5	5	5	5	0	25	25	SI
18-02-IL-CA-despues	5	5	5	5	5	0	25	25	SI
17-03-AL-CA-antes	5	5	5	5	5	1	25	24	SI
17-03-S-CA-antes	5	5	5	5	5	1	25	24	SI
17-03-S-CA-despues	5	5	5	5	5	1	25	24	SI
17-03-S-SA-antes	5	5	5	5	5	1	25	24	SI
18-02-AL-CA-despues	5	5	5	5	5	1	25	24	SI
18-02-IL-SA-antes	5	5	5	5	5	1	25	24	SI
20-02-IL-SA-despues	5	5	5	5	5	1	25	24	SI
13-03-IL-CA-despues	5	5	4	4	5	1	23	22	SI
17-03-AL-SA-despues	4	4	4	4	4	1	20	19	SI
13-03-AL-SA-antes	4	4	4	4	4	2	20	18	NO
13-03-S-CA-despues	4	4	4	4	4	2	20	18	NO
17-03-AL-CA-despues	4	4	4	4	4	2	20	18	NO
17-03-S-SA-despues	4	4	4	4	4	2	20	18	NO
18-02-AL-SA-despues	4	4	4	4	4	2	20	18	NO
18-02-S-CA-antes	4	4	4	4	4	2	20	18	NO
20-02-AL-SA-antes	4	4	4	4	4	2	20	18	NO
20-02-AL-SA-despues	4	4	4	4	4	2	20	18	NO
20-02-IL-CA-antes	4	4	4	4	4	2	20	18	NO
20-02-IL-CA-despues	4	4	4	4	4	2	20	18	NO
20-02-IL-SA-antes	4	4	4	4	4	2	20	18	NO
20-02-S-SA-antes	4	4	4	4	4	2	20	18	NO
20-02-S-SA-despues	4	4	4	4	4	2	20	18	NO
13-03-IL-CA-antes	4	4	4	3	4	2	19	17	NO
13-03-IL-SA-antes	4	4	4	4	4	3	20	17	NO
17-03-AL-SA-antes	4	4	4	4	4	3	20	17	NO
18-02-AL-SA-antes	4	4	4	4	4	3	20	17	NO
18-02-S-CA-despues	4	4	4	4	4	3	20	17	NO
18-02-S-SA-antes	4	4	4	4	4	3	20	17	NO
13-03-AL-SA-despues	3	4	4	3	4	3	18	15	NO
13-03-IL-SA-despues	3	4	4	3	4	3	18	15	NO
13-03-AL-CA-antes	2	5	4	3	4	4	18	14	NO
13-03-S-CA-antes	3	3	3	3	3	3	15	12	NO
17-03-IL-CA-antes	3	3	3	3	3	3	15	12	NO
18-02-S-SA-despues	3	3	3	3	3	3	15	12	NO
17-03-IL-CA-despues	3	3	3	3	3	4	15	11	NO
13-03-S-SA-antes	2	2	2	2	2	5	10	5	NO

Tabla F.1: Tabla de evaluación de segmentación de imágenes para Feng

Apéndice G

Código de registro de imágenes

```
clear all;
close all;

%Cargamos las imagenes
%%Estas son las imágenes antes y después (segmentadas)
image_1 = "18-02-AL-CA-antes";
image_2 = "18-02-AL-CA-despues";

fixed = imread(image_1 + '.png');
moving = imread(image_2 + '.png');

%Verificar el tamaño de imagenes
if size(fixed) ~= size(moving)
   fixed = imresize(fixed, size(moving));
end

%Usamos el tipo de dato uint8 para las imagenes
fixed = im2uint8(fixed);
moving = im2uint8(moving);

%Seleccionamos el tipo multimodal
[optimizer,metric] = imregconfig("multimodal");

%Especificamos los valores del optimizador
optimizer.InitialRadius = 0.0005;
optimizer.Epsilon = 0.000005;
optimizer.GrowthFactor = 1.3;
optimizer.MaximumIterations = 500;

%Hacemos el registro de imagenes
movingRegistered = imregister(moving,fixed,...
```

```
32 "affine",optimizer,metric);
33
34 %Mostramos las diferencias entre la imagen inicial y la imagen final
35 imshowpair(fixed,movingRegistered,"Scaling", ...
36     "joint",'ColorChannels', [1 2 0])
37
38 %Almacenamos la imagen registrada
39 imwrite(movingRegistered, image_2 + "_Registrada" + ".png");
```

Apéndice H

Código para cálculo de daño estructural

```
clear all;
close all;

%Cargamos las imagenes
image_1 = imread('18-02-AL-CA-antes.png');
image_2 = imread('18-02-AL-CA-despues_Registrada.png');

%Validamos el tamaño de las imagenes
if size(image_1) ~= size(image_2)
   image_2 = imresize(image_2, size(image_1));
end

% Calcular las diferencias entre las dos imágenes
differences = image_1 ~= image_2;

% Calcular el área afectada (número de píxeles diferentes)
area_affected = sum(differences(:));

% Calcular el porcentaje de daño estructural
total_area = numel(image_1);
damage_percentage = (area_affected / total_area) * 100;

% Mostrar los resultados del porcentaje del daño
fprintf('Pixeles totales imagen sin daño: %d\n',...
    total_area);
fprintf('El número de píxeles diferentes: %d\n',...
    area_affected);
fprintf('El porcentaje de daño estructural es: %.2f%%\n',...
    damage_percentage);
```

```
30
31 % Mostramos los resultados del daño
32 imshowpair(image_1,image_2,'falsecolor', 'ColorChannels', [1 2 0]);
```

Bibliografía

Abdulghafour, M. (2003). Image Segmentation using Fuzzy Logic and Genetic Algorithms. https://api.semanticscholar.org/CorpusID:268120834

Adelson, E., Anderson, C., Bergen, J., Burt, P., & Ogden, J. (1984). Pyramid methods in image processing. *RCA engineer*, *29*(6), 33-41.

Alam, M. S., & Jenkins, C. H. (2005). Damage Tolerance in Naturally Compliant Structures. *International Journal of Damage Mechanics*, *14*(4), 365-384. https://doi.org/10.1177/1056789505054313

Bagul, D. (s.f.). *Edge Detection by Zero-crossing*. https://notebook.community/darshanbagul/ComputerVision / EdgeDetection - ZeroCrossings / EdgeDetectionByZeroCrossings (accessed: 06.15.2023).

Batista-Plaza, D. M. K., Travieso. (2017). Biometric analysis for the recognition of spider species according to their webs, 1-6. https://doi.org/10.1109/IC3.2017.8284286

Berkeley. (s.f.). *Berkeley Segmentation Dataset 500*. https://www.kaggle.com/datasets/balraj98/berkeley-segmentation-dataset-500-bsds500?resource=download

Blackledge, T. A., Kuntner, M., & Agnarsson, I. (2011). The Form and Function of Spider Orb Webs. En *Advances in Insect Physiology* (pp. 175-262, Vol. 41). Elsevier. https://doi.org/10.1016/B978-0-12-415919-8.00004-5

Blackledge, T. A., & Zevenbergen, J. M. (2006). Mesh Width Influences Prey Retention in Spider Orb Webs. *Ethology*, *112*(12), 1194-1201. https://doi.org/10.1111/j.1439-0310.2006.01277.x

Bovik, A. C. (2009). *The essential guide to image processing* [OCLC: ocn262718637]. Academic Press.

Burger, W., & Burge, M. J. (2016). *Digital Image Processing: An Algorithmic Introduction Using Java*. Springer London. https://doi.org/10.1007/978-1-4471-6684-9

Canon. (s.f.). *EOS 7D Cámaras EOS*. https://www.cla.canon.com/cla/es/support/consumer/cameras/eos_cameras/eos_7d#productOverviewTab

Chacon, P., & Eberhard, W. G. (1980). Factors affecting numbers and kinds of prey caught in artificial spider webs, with considerations of how orb webs trap prey. *Bulletin of The British Arachnological Society*.

Challenge, G. (s.f.). *DRIVE: Digital Retinal Images for Vessel Extraction*. https://drive.grand-challenge.org/

Chen, M., Wang, J., Zhu, L., Deng, H., Wu, H., & Zhang, Z. (2023). Research on contour extraction and edge detection of sugarcane image based on MATLAB. *2023 IEEE*

3rd International Conference on Power, Electronics and Computer Applications (ICPECA), 1784-1787. https://doi.org/10.1109/ICPECA56706.2023.10076152

Chen, S., & Yang, X. (2021). An Enhanced Adaptive Sobel Edge Detector Based on Improved Genetic Algorithm and Non-Maximum Suppression. *2021 China Automation Congress (CAC)*, 8029-8034. https://doi.org/10.1109/CAC53003.2021.9727626

Chmiel, K., Herberstein, M. E., & Elgar, M. A. (2000). Web damage and feeding experience influence web site tenacity in the orb-web spider Argiope keyserlingi Karsch. *Animal Behaviour*, *60*(6), 821-826. https://doi.org/10.1006/anbe.2000.1541

Coello, C. A. C. (2019). *Computación Evolutiva*. Academia Mexicana de Computación, A.C.

Craig, C. L. (1988). Insect Perception of Spider Orb Webs in Three Light Habitats. *Functional Ecology*, *2*(3), 277. https://doi.org/10.2307/2389398

Craig, C. L. (1990). Effects of background pattern on insect perception of webs spun by orb-weaving spiders. *Animal Behaviour*, *39*(1), 135-144. https://doi.org/10.1016/S0003-3472(05)80733-X

Cui, Y., An, Y., Sun, W., Hu, H., & Song, X. (2021). Multiscale Adaptive Edge Detector for Images Based on a Novel Standard Deviation Map. *IEEE Transactions on Instrumentation and Measurement*, *70*, 1-13. https://doi.org/10.1109/TIM.2021.3083888

D. Dumitrescu, B. L. (2000). *Evolutionary Computation*. CRC Press.

Dale, J. W., & Schantz, M. v. (2007). *From genes to genomes: concepts and applications of DNA technology* (Second ed). John Wiley & sons.

Eberhard, W. G. (1976). Physical properties of sticky spirals and their connections: sliding connections in orb webs. *Journal of Natural History*, *10*(5), 481-488. https://doi.org/10.1080/00222937600770391

Eberhard, W. G. (1980). Effects of orb web orientation and spider size on prey retention. *Bulletin of The British Arachnological Society*.

Eberhard, W. G. (1988). Behavioral Flexibility in Orb Web Construction: Effects of Supplies in Different Silk Glands and Spider Size and Weight. *Journal of Arachnology*, *16*(3), 295-302. http://www.jstor.org/stable/3705916

Eberhard, W. G. (1990). Function and Phylogeny of Spider Webs. *Annual Review of Ecology and Systematics*, *2*, 341-372. http://www.jstor.org/stable/2097029

Eddins, S. (s.f.-a). *Multiresolution image pyramids and impyramid – part 1*. https://blogs.mathworks.com/steve/2019/04/02/multiresolution-image-pyramids-and-impyramid-part-1/ (accessed: 12.08.2023).

Eddins, S. (s.f.-b). *Multiresolution image pyramids and impyramid – part 2*. https://blogs.mathworks.com/steve/2019/04/09/multiresolution-image-pyramids-and-impyramid-part-2/ (accessed: 12.08.2023).

Eddins, S. (s.f.-c). *Multiresolution pyramids part 3: Laplacian pyramids*. https://blogs.mathworks.com/steve/2019/04/16/multiresolution-pyramids-part-3-laplacian-pyramids (accessed: 12.08.2023).

Eiben, A., & Smith, J. (2015). *Introduction to Evolutionary Computing*. Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-44874-8

Elgendy, M. (2020). *Deep learning for vision systems*. Manning Publications Co.

Feng, M.-L., & Tan, Y.-P. (2004). Contrast adaptive binarization of low quality document images. *IEICE Electronics Express*, *1*(16), 501-506. https://doi.org/10.1587/elex.1.501

Fogel, D. (1993). On the Philosophical Differences between Evolutionary Algorithms and Genetic Algorithms. *Proceedings of the Second Annual Conference on Evolutionary Programming*, 23-29.

Fogel, D. B., & Stayton, L. C. (1994). On the effectiveness of crossover in simulated evolutionary optimization. *Biosystems*, *32*(3), 171-182. https://doi.org/10.1016/0303-2647(94)90040-X

Fogel, D. (1994). Evolutionary programming: an introduction and some current directions. *Statistics and Computing*, *4*(2). https://doi.org/10.1007/BF00175356

Fogel, L. J., Owens, A. J., & Walsh, M. J. (1966). Artificial Intelligence through Simulated Evolution. https://api.semanticscholar.org/CorpusID:262309796

Galun, M., Basri, R., & Brandt, A. (2007). Multiscale Edge Detection and Fiber Enhancement Using Differences of Oriented Means. *2007 IEEE 11th International Conference on Computer Vision*, 1-8. https://doi.org/10.1109/ICCV.2007.4408920

Hernandez, S. G., & Saavedra, J. F. S. (s.f.). Umbralización adaptativa de imágenes basada en histogramas espacio color.

Hutchison, D., Kanade, T., Kittler, J., Kleinberg, J. M., Mattern, F., Mitchell, J. C., Naor, M., Nierstrasz, O., Pandu Rangan, C., Steffen, B., Sudan, M., Terzopoulos, D., Tygar, D., Vardi, M. Y., Weikum, G., Alpert, S., Galun, M., Nadler, B., & Basri, R. (2010). Detecting Faint Curved Edges in Noisy Images [Series Title: Lecture Notes in Computer Science]. En K. Daniilidis, P. Maragos & N. Paragios (Eds.), *Computer Vision – ECCV 2010* (pp. 750-763, Vol. 6314). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-15561-1_54

ImageJ. (s.f.). *ImageJ wiki*. https://imagej.net/software/imagej/

INBIOTECA. (s.f.). *Historia*. https://www.uv.mx/inbioteca/quienessomos/historia/

Jähne, B. (2005). *Digital Image Processing*. Springer.

Jain, N., Kumar, S., & Kumar, A. (2016). Analysis of edge detection techniques using soft computing approaches, 1-4. https://doi.org/10.1109/SCEECS.2016.7509310

Kelly, S. P., Sensenig, A., Lorentz, K. A., & Blackledge, T. A. (2011). Damping capacity is evolutionarily conserved in the radial silk of orb-weaving spiders. *Zoology*, *114*(4), 233-238. https://doi.org/10.1016/j.zool.2011.02.001

Khurshid, K., Siddiqi, I., Faure, C., & Vincent, N. (2009, enero). Comparison of Niblack inspired binarization methods for ancient documents. En K. Berkner & L. Likforman-Sulem (Eds.). https://doi.org/10.1117/12.805827

Li, Z.-N., Drew, M. S., & Liu, J. (2014). *Fundamentals of Multimedia*. Springer International Publishing. https://doi.org/10.1007/978-3-319-05290-8

Lim, J. S. (1990). *Two-dimensional signal and image processing*. Prentice Hall PTR.

Lubin, Y., Ellner, S., & Kotzman, M. (1993). Web Relocation and Habitat Selection in Desert Widow Spider. *Ecology*, *74*(7), 1915-1928. https://doi.org/10.2307/1940835

Marr, D., & Hildreth, E. C. (1980). Theory of edge detection. *Proceedings of the Royal Society of London. Series B. Biological Sciences*, *207*, 187-217. https://doi.org/10.1098/rspb.1980.0020

MathWorks. (s.f.-a). *confusionmat*. https://la.mathworks.com/help/stats/confusionmat.html#d126e312355

MathWorks. (s.f.-b). *edge*. https://la.mathworks.com/help/images/ref/edge.html#d126e36927

MathWorks. (s.f.-c). *Elemento estructurante morfológico*. https://la.mathworks.com/help/images/ref/strel.html

MathWorks. (s.f.-d). *imregconfig*. https://la.mathworks.com/help/images/ref/imregconfig.html

MathWorks. (s.f.-e). *imregister*. https://la.mathworks.com/help/images/ref/imregister.html

MathWorks. (s.f.-f). *Registrar imágenes con distorsión de proyección utilizando puntos de control*. https://la.mathworks.com/help/images/registering-an-aerial-photo-to-an-orthophoto.html

MathWorks. (s.f.-g). *Thresholding algorithms: Niblack (local)*. https://craftofcoding.wordpress.com/2021/09/30/thresholding-algorithms-niblack-local/ (accessed: 14.05.2024).

Mattes, D., Haynor, D. R., Vesselle, H., Lewellyn, T. K., & Eubank, W. (2001). Nonrigid multimodality image registration (M. Sonka & K. M. Hanson, Eds.), 1609-1620. https://doi.org/10.1117/12.431046

Mcnett, B. J., & Rypstra, A. L. (2000). Habitat selection in a large orb-weaving spider: vegetational complexity determines site selection and distribution. *Ecological Entomology*, *25*(4), 423-432. https://doi.org/10.1046/j.1365-2311.2000.00279.x

Mezura-Montes, E. (s.f.). *Clase 7 Computacion Evolutiva*. https://eminus.uv.mx/eminus4/page/resource/viewOnline?file=%5Ccur_46568%5CContenido%5Celem_152213%5C3a82a748-1f54-4021-b587-d3ceb9e77362.pdf (accessed: 12.10.2023).

Milan Sonka, R. B., Vaclav Hlavac. (2014). *Image Processing, Analysis, and Machine Vision*. Cengage Learning.

Motl, J. (s.f.-a). *Feng local image thresholding*. https://la.mathworks.com/matlabcentral/fileexchange/41771-feng-local-image-thresholding

Motl, J. (s.f.-b). *Niblack local thresholding*. https://la.mathworks.com/matlabcentral/fileexchange/40849-niblack-local-thresholding

Nentwig, W. (1982). Why do only certain insects escape from a spider's web? *Oecologia*, *53*(3), 412-417. https://doi.org/10.1007/BF00389023

Niblack, W. (1986). *An introduction to digital image processing*. Englewood Cliffs, N.J. : Prentice-Hall International.

Nixon, M. S., & Aguado, A. S. (2006). *Feature extraction and image processing* (1. ed., reprinted). Newnes.

of Biomedical Imaging, N. I., & (NIBIB), B. (s.f.). *Tomografía Computarizada (TC)*. https://www.nibib.nih.gov/espanol/temas-cientificos/tomograf%C3%ADa-computarizada-tc

of Edinburgh, T. U. (s.f.-a). *Laplacian/Laplacian of Gaussian*. https://homepages.inf.ed.ac.uk/rbf/HIPR2/log.htm (accessed: 10.12.2023).

of Edinburgh, T. U. (s.f.-b). *ZeroCrossing*. https://homepages.inf.ed.ac.uk/rbf/HIPR2/flatjavasrc/ZeroCrossing.java (accessed: 1.20.2024).

Ofir, N., Galun, M., Alpert, S., Brandt, A., Nadler, B., & Basri, R. (2020). On Detection of Faint Edges in Noisy Images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *42*(4), 894-908. https://doi.org/10.1109/TPAMI.2019.2892134

Ofir, N., & Keller, Y. (2021). Multi-scale Processing of Noisy Images using Edge Preservation Losses. *2020 25th International Conference on Pattern Recognition (ICPR)*, 1-8. https://doi.org/10.1109/ICPR48806.2021.9413325

Peters, J. F. (2017). *Foundations of Computer Vision* (Vol. 124). Springer International Publishing. https://doi.org/10.1007/978-3-319-52483-2

Radha. R, J. S. (2014). An Effective Algorithm for Edges and Veins Detection in Leaf Images, 128-131. https://doi.org/10.1109/WCCCT.2014.1

Rafael C. Gonzales, R. E. (2002). *Digital Image Processing*. Prentice Hall.

Raghunathan, S., Stredney, D., Schmalbrock, P., & Clymer, B. D. (s.f.). Image Registration Using Rigid Registration and Maximization of Mutual Information.

Ramesh Jain, B. G. S., Rangachar Kasturi. (1995). *Machine Vision*. McGraw-Hill,

Rao, D. (s.f.). *Rao Spider Lab*. https://raospiderlab.org/

Rasband, W. (s.f.). *Wayne Rasband*. https://imagej.net/people/rasband

Rittschof, C. C., & Ruggles, K. V. (2010). The complexity of site quality: multiple factors affect web tenure in an orb-web spider. *Animal Behaviour*, *79*(5), 1147-1155. https://doi.org/10.1016/j.anbehav.2010.02.014

Ronneberger, O., Fischer, P., & Brox, T. (2015, mayo). *U-Net: Convolutional Networks for Biomedical Image Segmentation* [arXiv:1505.04597 [cs]]. arXiv. Consultado el 12 de marzo de 2024, desde http://arxiv.org/abs/1505.04597
Comment: conditionally accepted at MICCAI 2015.

Russ, J. C. (1995). *The image processing handbook* (2nd ed). CRC Press.

Rypstra, A. L. (1982). Building a better insect trap; An experimental investigation of prey capture in a variety of spider webs. *Oecologia*, *52*(1), 31-36. https://doi.org/10.1007/BF00349008

Shabankareh, S. G., & Shabankareh, S. G. (2019). Improvement of Edge-Tracking Methods using Genetic Algorithm and Neural Network, 1-7. https://doi.org/10.1109/ICSPIS48872.2019.9066026

Sucar., L. E. (2011). *Visión computacional*. Instituto Nacional de Astrofísica, Óptica y Electrónica. México.

Szeliski, R. (2021). *Computer Vision: Algorithms and Applications*. Springer.

University, C. M. (s.f.). *Q1.2: What's Evolutionary Programming (EP)?* https://www.cs.cmu.edu/Groups/AI/html/faqs/ai/genetic/part2/faq-doc-3.html (accessed: 12.29.2023).

Whelan, P. F., & Molloy, D. (2001). *Machine Vision Algorithms in Java*. Springer London. https://doi.org/10.1007/978-1-4471-0251-9

Zhang, J., Marszalek, M., Lazebnik, S., & Schmid, C. (2006). Local Features and Kernels for Classification of Texture and Object Categories: A Comprehensive Study. *2006 Conference on Computer Vision and Pattern Recognition Workshop (CVPRW'06)*, 13-13. https://doi.org/10.1109/CVPRW.2006.121

Zhang, M., Li, N., Wang, F., & Fu, T. (2022). Analysis of Image Edge Extraction Methods. *2022 IEEE 5th International Conference on Automation, Electronics and Electrical*

Engineering (AUTEEE), 830-837. https://doi.org/10.1109/AUTEEE56487.2022.9994429

Zhang, Y., Han, X., Zhang, H., & Zhao, L. (2017). Edge detection algorithm of image fusion based on improved Sobel operator, 457-461. https://doi.org/10.1109/ITOEC.2017.8122336