Universidad Autónoma de Baja California

Facultad de Ingeniería, Arquitectura y Diseño

Maestría y Doctorado en Ciencias e Ingeniería

Algoritmos genéticos para el reconocimiento de un patrón triangular en procesamiento de imágenes para conteo de larvas de peces

TESIS

que para cubrir parcialmente los requisitos necesarios para obtener el grado de

MAESTRO EN INGENIERÍA

Presenta

MIGUEL RICARDO GONZÁLEZ MÁRQUEZ

Ensenada, Baja California, junio de 2018.

Universidad Autónoma de Baja California

Facultad de Ingeniería, Arquitectura y Diseño

Algoritmos genéticos para el reconocimiento de un patrón triangular en procesamiento de imágenes para conteo de larvas de peces

TESIS

que para cubrir parcialmente los requisitos necesarios para obtener el grado de

MAESTRO EN INGENIERÍA

Presenta

Miguel Ricardo González Márquez

Aprobada por:

Dr. Miguel Enrique Martínez Rosas
Director de Tesis

Dra. María de los Ángeles Cosío León
Miembro del Comité

Dr. Carlos Alberto Brizuela Rodríguez
Miembro del Comité

Dr. Humberto Cervantes de Ávila
Miembro del Comité

Ensenada, Baja California, junio del 2018.

Resumen de la tesis de **Miguel Ricardo González Márquez**, presentada como requisito parcial para la obtención del grado de MAESTRO EN INGENIERÍA del programa de Maestría y Doctorado en Ciencias e Ingeniería (MYDCI) de la UABC. Ensenada Baja California, México, junio de 2018.

Algoritmos genéticos para el reconocimiento de un patrón triangular en procesamiento de imágenes para conteo de larvas de peces

Resumen Aprobado por:

Dr. Miguel Enrique Martínez Rosas
Director de Tesis

En este trabajo se aborda la problemática de clasificación y conteo de larvas de peces a través del procesamiento digital de imágenes (un método no invasivo), y se hace uso de una metaheurística (Algoritmo Genético y una parte del modelo de islas) para el reconocimiento de un patrón triangular, de manera que sea capaz de generar dicha clasificación y realizar la contabilización de larvas de peces de manera eficiente y certera. Los tiempos de procesamiento y los porcentajes de certeza por imagen son comparados con una investigación previa a este trabajo y que al servir de referencia es ampliamente citada a lo largo de este documento.
Entre las aportaciones de este trabajo, se debe mencionar particularmente el desarrollo de un Generador Aleatorio-Sintético de Patrones Triangulares (GASPT), el cual fue concebido con la intención de entrenar a la metaheurística utilizada en el proceso integrado; así mismo, se establece una frontera límite, generando un parámetro acerca de la cantidad de larvas por muestra que nuestra propuesta es capaz de procesar con un error no mayor al 2 %, siendo así capaces de no sacrificar la eficiencia del algoritmo.

Palabras Clave: *Clasificación*, *Metaheurística*, *Algoritmos Genéticos*, *Patrón-triangular*, *Larvas de peces*, *Umbral-adecuado*

Dedicatoria

A Dios por tantas bendiciones que han llegado a mi vida, las cuales la han llenado de alegría, y dándome la fuerza para seguir adelante día con día.

A mis padres **Rogelia** y **Ricardo**, por su apoyo incondicional desde que tengo memoria y por enseñárme con su ejemplo que los sacrificios del presente de verán recompensados en el futuro.

A mis hermana **Margarita** por estar presente en cada prueba que Dios me ha puesto y ayudar a superarla.

A mi esposa **Daniela** por ese apoyo incondicional e incansable, por su amor y su paciencia durante este proceso, por estar a mi lado y dejarme compartir ese logro a su lado.

Agradecimientos

Quiero expresar mi agradecimiento:

A mi director de tesis **Dr. Miguel Enrique Martínez Rosas**. Su dirección, su paciencia, pero sobre todo su dedicación durante todo este proceso han sido pilar para poder llevar a cabo esta investigación.

A mis sinodales **Dra. María de los Ángeles Cosío León**, **Dr. Carlos Alberto Brizuela Rodríguez** y **Dr. Humberto Cervantes de Ávila** por sus consejos, criticas y apoyo durante el desarrollo de esta tesis.

A mis compañeros y amigos, por hacer de este proceso una buena experiencia, brindando retroalimentación, a nivel profesional y personal, cuando esta fuese necesaria

A todos los docentes que impartieron las clases, por su guía y enseñanza durante este proceso de aprendizaje.

Al personal de la Universidad Autónoma de Baja California (UABC) de la Facultad de Ingeniería, Arquitectura y Diseño, por sus atenciones, amabilidad y disposición para ayudarme.

Al Consejo Nacional de Ciencia y Tecnología (CONACyT) por el apoyo económico brindado y a la Facultad de Ingeniería, Arquitectura y Diseño de la Universidad Autónoma de Baja California (UABC) por las facilidades otorgadas para la realización de éste trabajo.

Índice general

1. Introducción **1**
- 1.1. Planteamiento del problema . . . 1
- 1.2. Justificación . . . 2
- 1.3. Objetivo General . . . 3
- 1.4. Objetivos específicos . . . 3
- 1.5. Secuencia de la tesis . . . 4

2. Revisión de literatura **5**
- 2.1. Antecedentes . . . 5
  - 2.1.1. Imagen . . . 7
  - 2.1.2. ¿RGB o escala de grises? . . . 7
  - 2.1.3. Segmentación . . . 9
- 2.2. Metaheurísticas en PDI . . . 13
  - 2.2.1. Recocido simulado . . . 15
  - 2.2.2. Búsqueda Tabú . . . 16
  - 2.2.3. Optimización por enjambre de partículas . . . 17
  - 2.2.4. Algoritmos Genéticos . . . 18
  - 2.2.5. Algoritmos Coevolutivos . . . 20
  - 2.2.6. Algoritmo Genético Paralelo - Modelo de islas . . . 23
- 2.3. Conteo de peces . . . 24

3. Desarrollo de Investigación **28**
- 3.1. Algoritmo propuesto por Martínez . . . 29
- 3.2. Selección de representación de pixeles (espacio de color o escala de grises) . . . 32
- 3.3. Cálculo de umbral adecuado . . . 34
- 3.4. Cúmulos de pixeles . . . 35
- 3.5. Clasificación de áreas . . . 37
- 3.6. Algoritmos propuestos . . . 37
  - 3.6.1. Patrón triangular . . . 37
  - 3.6.2. Usando un método exacto . . . 50
  - 3.6.3. Usando una metaheurística . . . 55
  - 3.6.4. Casos especiales . . . 57
- 3.7. Método integrado . . . 60

4. Pruebas y resultados **61**
4.1. Escala de grises 62
4.2. Umbral óptimo 62
4.3. Cúmulos de pixeles 65
4.4. Clasificación de áreas 65
4.5. Patrón triangular (método exacto) 68
4.6. Patrón triangular (metaheurística) 70
4.7. Método integrado 71
4.7.1. Versión 1 (implementación sin tomar en cuenta casos especiales) . . . 72
4.7.2. Versión 1.1 (implementación tomando en cuenta casos especiales) . . 72
4.8. Discusiones 75

5. Conclusiones Generales **76**
5.1. Aportaciones 77
5.2. Trabajo a futuro 77

Índice de figuras

1.1. Ejemplo de imagen de larvas de peces en ambiente controlado [1]. 2

2.1. Imagen RGB (izquierda), imagen en escala de grises (derecha). 7
2.2. Representación escala de grises a partir de información en espacio de color RGB [2] 8
2.3. Curva de sensibilidad espectral. [2] 9
2.4. Ejemplo umbral adecuado implementado, $T = 120$ 9
2.5. Implementación umbral óptimo usando el método de Otsu, $T = 120$ 10
2.6. Implementación procedimiento Sobel 11
2.7. Ejemplo del CED implementado 12
2.8. (a) imagen de entrada (b) Resultados con CED (C) Método alternativo (d) Resultados obtenidos por Slatnia et al. [3]. 12
2.9. Imagen orginal (izquierda), Resultados implementando el procedimiento de Singh et al. [4] (derecha). 13
2.10. Clasificación de algunas Metaheurísticas según la cantidad de soluciones procesadas 14
2.11. Imagen de entrada (izquierda), método ETS (derecha) [5] 16
2.12. Imagen de entrada (izquierda), Segmentación por Otsu (centro), método QP-SO (derecha) [6] 18
2.13. Imagen de entrada y círculos encontrados sobrepuestos (izquierda), mapa de bordes (derecha) [7] 19
2.14. Imagen de entrada (izquierda), entropía de Renyi (centro), entropía de Tsallis (derecha) [8] 20
2.15. Gráfica de convergencia CCGA (Cooperative Coevolutionary Genetic Algorithm) [9] 23
2.16. Descripción gráfica de la comunicación entre islas 24
2.17. Imagen capturada por Toh et al. [10] 24
2.18. Diagrama del montaje de Toh et al. [10] 25
2.19. Imágenes analizadas por Fan et al. [11] 25
2.20. Traslapes caracterizados por Fan et al. [11] 25
2.21. Sistema de Adquisición de Imágenes (SAI V5) [1] 26
2.22. Figura procesada por el método de Martínez [1] 27

3.1. Diagrama de flujo sobre el proceso de Martínez [1] . . . 28
3.2. Imagen seccionada en ventanas (Zona especial). . . . 29
3.3. 40 px de ancho . . . 30
3.4. 100 px de ancho . . . 30
3.5. Triangulación de los centros localizados . . . 31
3.6. Perímetros y cálculo de distancias . . . 31
3.7. Espacio paramétrico por CHT . . . 32
3.8. Imágenes en escala de grises. Usando Ecuación 2.1 (izquierda), esando Ecuación 2.2 (derecha) . . . 33
3.9. Usando Ecuación 2.2 y umbralizado con $T = 128$ . . . 33
3.10. Usando Ecuación 2.1 y umbralizado con $T = 128$ . . . 34
3.11. Resultado intervalo de umbrales $115 \leq T \leq 120$ . . . 35
3.12. Ejemplo imagen binarizada . . . 35
3.13. Descripción gráfica de *findContours()* , a, b, c y d son los puntos extraidos . . . 36
3.14. Implementación de *findContours()* . . . 36
3.15. Triángulo formado por ojos y aparato digestivo . . . 38
3.16. Triángulo formado por ojos y aparato digestivo (descripción gráfica) . . . 38
3.17. Descripción formal del patrón triangular (caso 1) . . . 39
3.18. Obtención de los puntos a y b (patrón triangular) . . . 40
3.19. Descripción formal del patrón triangular (caso 2) . . . 41
3.20. Obtención de los puntos a y b (patrón triangular) . . . 42
3.21. Descripción formal del patrón triangular (caso 3) . . . 43
3.22. Obtención de los puntos a y b (patrón triangular) . . . 44
3.23. Descripción formal del patrón triangular (caso 4) . . . 45
3.24. Obtención de los puntos a y b (patrón triangular) . . . 46
3.25. Descripción formal del patrón triangular (caso 5) . . . 46
3.26. Descripción formal del patrón triangular (caso 6) . . . 47
3.27. Descripción formal del patrón triangular (caso 7) . . . 48
3.28. Descripción formal del patrón triangular (caso 8) . . . 48
3.29. 50 patrones sintéticos creados con el generador propuesto (implementación en C++) . . . 50
3.30. Representacón gráfica de una nube de puntos y un grafo . . . 51
3.31. Triangulación de un conjunto de puntos . . . 52
3.32. La distancia de los puntos L_2 a un punto L_1 . . . 53
3.33. Gráfica de bigotes de medidas reales: (1)$\rightarrow \overline{ab}$, (2)$\rightarrow \overline{ac}$, (3)$\rightarrow \overline{bc}$ . . . 54
3.34. Gráfica de bigotes de medidas reales: (1)$\rightarrow R_1$, (2)$\rightarrow R_1$ . . . 54
3.35. Representación binaria de cada indivuo . . . 55
3.36. Larvas en posiciones no ideales . . . 57
3.37. Larvas traslapadas . . . 58
3.38. Larvas fuera del espacio de captura . . . 58
3.39. Ruido detectado como parte de una larva . . . 59
3.40. Larvas con pigmentación diferente y tamaño de ojos fuera de la clasificación 59

4.1. Imágenes en RGB (experimento realizado por Martínez en Junio de 2015 [1]) 61
4.2. Imágenes en RGB (experimento realizado por Martínez en Junio de 2015 [1]) 61
4.3. Pase a escala de grises usando la Ecuación 2.2: img1 (izquierda), img2 (derecha) . . . 62
4.4. Pase a escala de grises usando la Ecuación 2.2: img3 (izquierda), img4 (derecha) . . . 62
4.5. Gráfica obtenida del Algoritmo 7 para contabililzar la cantidad de pixeles obtenidos en cada nivel de umbral por imagen . . . 63
4.6. Acercamiento en el intervalo $115 \leq T \leq 120$ . . . 64
4.7. Acercamiento en el intervalo $180 \leq T \leq 220$ . . . 64
4.8. Resultado función *findContours()*:img1 (a), img2 (b) . . . 65
4.9. Resultado función *findContours()*:img3 (a), img4 (b) . . . 65
4.10. Gráfica de bigotes de los valores de áreas de ojos ($1 \rightarrow$ *img1*, $2 \rightarrow$ *img2*, $3 \rightarrow$ *img3*, $4 \rightarrow$ *img4*) . . . 66
4.11. Gráfica de bigotes de los valores de áreas de los aparatos digestivos ($1 \rightarrow$ *img1*, $2 \rightarrow$ *img2*, $3 \rightarrow$ *img3*, $4 \rightarrow$ *img4*) . . . 66
4.12. Gráfica de bigotes de los valores de la relación $\frac{base}{altura}$ de ojos ($1 \rightarrow$ *img1*, $2 \rightarrow$ *img2*, $3 \rightarrow$ *img3*, $4 \rightarrow$ *img4*) . . . 67
4.13. Gráfica de bigotes de los valores de la relación $\frac{base}{altura}$ de los aparatos digestivos ($1 \rightarrow$ *img1*, $2 \rightarrow$ *img2*, $3 \rightarrow$ *img3*, $4 \rightarrow$ *img4*) . . . 67
4.14. 20 patrones (a), 40 patrones (b) . . . 68
4.15. 60 patrones (a), 100 patrones (b) . . . 68
4.16. 140 patrones (a), 200 patrones (b) . . . 69
4.17. Porcentaje de certeza según la cantidad de patrones a evaluar (método exacto) 69
4.18. Tiempo de procesamiento según la cantidad de patrones a identificar (método exacto) . . . 69
4.19. 20 patrones (a), 40 patrones (b) . . . 70
4.20. 60 patrones (a), 100 patrones (b) . . . 70
4.21. 140 patrones (a), 200 patrones (b) . . . 70
4.22. Tendencia del porcentaje de certeza según la cantidad de patrones a evaluar usando una metaheurística . . . 71
4.23. Tendencia del tiempo de procesamiento según la cantidad de patrones a identificar usando una metaheurística . . . 71
4.24. Resultados V 1.0: *img1* (a), *img2* (b) . . . 72
4.25. Resultados V 1.0: *img3* (a), *img4* (b) . . . 72
4.26. Resultados V 1.1: *img1* (a), *img2* (b) . . . 72
4.27. Resultados V 1.1: *img3* (a), *img4* (b) . . . 73

Índice de tablas

2.1. Comparación entre espacios de color 8
2.2. Comparación entre metaheurísticas 21
2.3. Características del proceso evolutivo 22

3.1. Parámetros para el algoritmo evolutivo 56

4.1. Conteos manuales, resultados de los conteos usando el algoritmo V 1.1, y tiempos de procesamiento 74

Capítulo 1

Introducción

El presente trabajo de investigación trata sobre el conteo y clasificación de larvas de peces a través de Procesamiento Digital de Imágenes (PDI). El trabajo de investigación que se propone en este documento, hace uso de Algoritmos Genéticos (GA) y una parte del modelo de islas, para el reconocimiento de un patrón encontrado; para el entrenamiento del algoritmo se usa un generador sintético de dicho patrón. Por otro lado, los algoritmos propuestos son implementados en C++, python y se hace uso de la biblioteca de visión por computadora OpenCV. De esta manera se generó un algoritmo eficiente capaz de realizar un conteo y clasificación de larvas de peces a través de PDI con un error menor al 2 %, haciendo todo esto en un tiempo computacional menor a un segundo por imagen, en comparación a los reportados por Martínez [1].

1.1. Planteamiento del problema

En la Facultad de Ciencias Marinas de la Universidad Autónoma de Baja California (UABC) campus Ensenada, hay un laboratorio, en el cual producen larvas de peces (específicamente *Totoaba macdonali*), ya que es una especie en peligro de extinción. Una de las etapas de este proceso de producción, es el conteo de dichos especímenes en etapa larvaria. Este conteo se lleva a cabo de manera manual, es decir, por un método invasivo; debido a esto una parte considerable de la población de dicha especie no sobrevive y por ende no llega a vida silvestre, pero también hacer un conteo de manera manual toma mucho tiempo y con posibilidades altas de cometer errores.

El trabajo realizado por Martínez [1], propone como uno de sus trabajos a futuro, la reducción en los tiempos de procesamiento del algoritmo propuesto y que además incluya la utilización de un lenguaje de programación en código libre, y ahí es donde retomamos su trabajo dando pie a esta investigación. El tiempo promedio reportado en su tesis es de 45 segundos promedio por imagen, con una certeza de entre el 97 % y el 100 %. Se debe tomar en cuenta que la captura de las imágenes de larvas de peces fueron efectuadas en un ambiente controlado, esto da la posibilidad de adquirir imágenes con la calidad suficiente para extraer datos certeros, un ejemplo de las imágenes con las que trabajamos se muestra en la Figura

1.1. Los factores a considerar para obtener una implementación eficiente de los algoritmos propuestos son: la cantidad de pixeles contenidos en las imágenes (impuesta por la resolución de dichas imágenes), los recursos del equipo de cómputo donde se pretende realizar el procesamiento (el cual restringe el tiempo de ejecución de los algoritmos), la posición de los peces y los traslapes que se presenten al momento de la captura de imágenes, entre otros; los factores mencionados se describen a detalle posteriormente, asi como también la propuesta de investigación para reconocer y evaluar las posibles detecciones erroneas de larvas.

Figura 1.1: Ejemplo de imagen de larvas de peces en ambiente controlado [1].

1.2. Justificación

En PDI el reconocimiento de objetos es un problema abierto y la solución a dicho problema tiene múltiples aplicaciones. Aún cuando existen algoritmos para detectar bordes, reconocer áreas de interés y reconocer patrones, la implementación de detección objetos, cuando se trata de seres vivos y en movimiento, presenta aún muchos retos por resolver, por esta razón en nuestra investigación al hacer un conteo de larvas de peces traslúcidas en movimiento, introduce una dificultad aún mayor. El uso de una metaheurístia en PDI (GA específicamente), ha demostrado dar buenos resultados en espacios de búsqueda complejos, según lo reportado en la literatura. Modelar un problema para que pueda ser abordado por una metaheurística, permite hacer una ejecución más eficiente, para poder optimizar el tiempo de ejecución; para tener un punto de comparación, utilizaremos los datos proporcionados por Martínez y los compararemos con nuestro algoritmo. Los algoritmos propuestos son implementados en un lenguaje de programación de código libre, ya que esto permitirá el uso de licencias libres. El tiempo de procesamiento es uno de los factores fundamentales a tomar en cuenta, dentro

de él podemos encontrar diferentes elementos que inciden directamente en la optimización de dicho tiempo, tales como: el tipo de programación a desarrollar (paralelo, secuencial), los recursos del equipo de cómputo donde se pretende implementar el algoritmo (cantidad de memoria, capacidad del procesador), la arquitectura de computadora establecida (x86, x64), el lenguaje a utilizar (C++, python, octave); además se evalua su desempeño bajo diferentes condiciones, de manera que se puedan obtener resultados eficientes, certeros y veraces. Para solucionar la problemática planteada, se proponen los objetivos descritos a continuación.

1.3. Objetivo General

Diseñar y evaluar un algoritmo para el conteo eficiente de larvas de peces a través del procesamiento digital de imágenes.
Con el propósito de alcanzar el objetivo general se plantea los siguientes objetivos específicos:

1.4. Objetivos específicos

- *Establecer parámetros de referencia:*
  - Banco de imágenes a procesar
  - Tipo representación de pixeles a trabajar (RGB, escala de grises)
  - Hacer una revisión de las diferentes metaheurísticas en PDI, especificando el uso de GA en el reconocimiento de un patrón definido
  - Hacer uso de python y C++ como lenguaje de programación para implementar los algoritmos
  - Hacer uso de la paquetería de visión por computadora OpenCV para poder procesar las imágenes
- *Hacer mediciones de tiempo de procesamiento*
  - Identificar cuáles de las variables tienen un mayor impacto sobre la eficiencia de nuestros algoritmos
  - Comparar métodos exactos contra la metaheurística propuesta
  - Comparar los tiempos de procesamiento obtenidos y el porcentaje de certeza, con los reportados por Martínez en su investigación [1]

1.5. Secuencia de la tesis

Este trabajo está dividido en los siguientes capítulos:

Capítulo 2 En este capítulo presenta la revisión de literatura, incluyendo definiciones y la descripción detallada de los conceptos de PDI: detección de bordes, umbralización, reconocimiento de áreas de interés, reconocimiento de patrones. También el cómo una metaheurística sirve como auxiliar en PDI, incluyendo las investigación realizadas en donde ambas áreas convergen.

Capítulo 3 Se describe el diseño e implementación de los algoritmos propuestos, los pasos utilizados y la base matemática de los mismos; se presenta el generador sintético de patrones, la solución del patrón generado tanto por un método exacto como usando un GA (modelo de islas), y el método integrado para generar un conteo y clasificación en imágenes reales.

Capítulo 4 En éste capítulo se presentan los resultados del desempeño de la implementación los algoritmos propouestos en el Capítulo 3, mostrando en gráficas los tiempos y el porcentaje de certeza en dichos resultados.

Capítulo 5 Finalmente, en este capítulo se dan las conclusiones del trabajo, y se hacen las comparaciones con el trabajo de Martínez [1], y se plantea los trabajos a futuro.

Capítulo 2

Revisión de literatura

En este capítulo se formaliza el concepto de imagen digital, se presenta una revisión de la literatura acerca de los siguientes temas: espacio de color RGB y escala de grises, el proceso de segmentación en PDI (detección de bordes, reconocimiento de áreas de interés, umbralización), reconocimiento de patrones; se define el concepto de metaheurísticas y las investigaciones encontradas en la literatura donde se converge con PDI, se compara el uso de las diferentes metaheurísticas en PDI, se fundamenta el uso de GA, se conceptualiza algunos de los diferentes paradigmas (modelo coevolutivo y de islas). Se hace una revisión sobre los métodos utilzados especificamente en el conteo de peces para verificar la situación actual de esta problemática y se resalta la pertinencia de esta investigación.

2.1. Antecedentes

El origen del procesamiento digital de imágenes (PDI) se remonta a los años 20's, cuando en la industria de la prensa, fueron enviadas imágenes a través de un cable submarino entre Londres y Nueva York, introduciendo el "Sistema cableado de transmisión de imágenes Bartlane"; esto redujo el tiempo requerido para transportar imágenes a través del océano Atlántico, desde más de una semana hasta menos de tres horas, sin embargo, unos de los principales problemas era poder aumentar la calidad de las imágenes. Para los 40's se establecen las bases de lo que hoy llamamos una computadora digital moderna cuando John von Newman introdujo dos términos: la memoria, capaz de almacenar datos y programas y la bifurcación condicional. Ambas ideas son los fundamentos de la unidad central de procesamiento (CPU). Para los 70's ya se estaba pensando en el término de paralelismo y mayor capacidad de procesamiento, siendo todo esto la base para las computadoras con la capacidad de procesar imágenes, como lo conocemos ahora [12].

Las aplicaciones del PDI son múltiples, y gracias al avance tecnológico de las últimas décadas en el área de la computación ahora es posible realizar un procesamiento más detallado, con imágenes de mayor resolución y con mayor profundidad de colores.

En la actualidad, los sistemas de visión artificial son mecanismos que utilizan técnicas para el procesamiento y análisis de la información obtenida de una imagen digital, esto ha

cobrado gran importancia para captar y monitorizar procesos de manera rápida haciendo el trabajo más eficiente y preciso, aunque este sigue siendo un problema sin resolver debido a la variedad de aplicaciones. Entre las aplicaciones de estos métodos, se pueden mencionar: el conteo de objetos sin contacto físico, inspección de la calidad de productos donde se puede verificar con métodos tradicionales, automatizar tareas repetitivas, por mencionar algunos. El campo de procesamiento digital de imágenes se refiere a analizar imágenes digitales a través de una computadora, tener en cuenta que una imagen digital es una función bidimensional en el espacio que forma una matriz de valores, y deducimos de esto que cada valor corresponde al color o nivel de tonos de gris de los elementos que están contenidos en una imagen digital y dichos elementos los podemos llamar: elementos de imagen, puntos o pixeles, siendo este último el más utilizado [12]. Algunas de las técnicas tradicionales para el análisis de imágenes digitales son: umbralización, detección de bordes, segmentación, morfología de una imagen, vecindad de pixeles, etc. En los sistemas de visión actuales, existen diferentes métodos novedosos para analizar imágenes, tales como redes neuronales (NN), autómatas finitos (FA), autómatas celulares (CA), algoritmos evolutivos (EA).

En las ultimas décadas el reconocimiento de patrones ha cobrado gran importancia debido a la diversidad de aplicaciones que este proceso tiene, así como también el tiempo computacional que se toma para analizar una imagen; dichas aplicaciones pueden ser: reconocimiento de áreas geográficas a través de imágenes satelitales, reconocimiento de tumores cerebrales analizando resonancias mangnéticas, conteo de frutas procesando imágenes, conteo estimado de gavocitos para la fecundación de peces, conteo de larvas de peces para estimar la población total sin necesidad de manipularlos manualmente.

Cuando se analiza una imagen digital una de las tareas tradicionales es encontrar los bordes, los cuales definen un cambio drástico en la luminosidad o el color de la imagen, para ello podemos usar el “Canny Edge Detector” (CED), el cual consiste en detectar un cambio radical en el patrón de pixeles, dibujando una línea sobre ese cambio. También podemos utilizar el “Shen-Castan Edege Detector”(SCED), el cual ataca el mismo problema de manera diferente [13]; en cuanto a funcionalidad se refiere, ambos algoritmos dan resultados factibles y nos servirán como base para el análisis y el desarrollo del nuevo algoritmo.

Un método eficiente para el conteo de objetos en una imagen digital, es la implementación de algoritmos genéticos, el cual también es utilizado para la reconstrucción de imágenes a través de la combinación de la vecindad de pixeles en cierto pixel identificado, esto nos puede dar como resultado un patrón y dicho patrón puede ser aprovechado por un autómata, ya sea finito o celular, para realizar un análisis o un simple conteo. Un ejemplo de esto es el trabajo propuesto por Skaruz et al. [14], al utilizar un algoritmo genético para detectar patrones de cambio aplicando las reglas de un autómata celular para reconstruir pixeles de una imagen destruida o modificada.

En los últimos años han surgido diferentes métodos de conteo de peces, un ejemplo de ello es lo que hicieron Zheng y Zhang [15], usando un método de conteo a través de redes neuronales artificiales difusas, el cual se destaca por la novedosa y efectiva forma de abordar la problemática, ya que proponen una solución para los traslapes de peces partiendo de las carácteristicas geométricas que los peces proporcionan.

En particular, el enfoque de este proyecto es optimizar las técnicas de procesamiento

digital de imágenes para reducir el tiempo computacional, usando GA para el conteo de larvas de peces. Nuestro objetivo deberá ser logrado a través de reconocer un patrón y definirlo, esto nos permitirá compararlo en toda la imagen y poder almacenar el número de veces que se repite dicho patrón, los objetivos se describirán a detalle más adelante.

Un problema muy importante en cada uno de los algoritmos utilizados es: la optimización, tanto de los recursos utilizados, el tiempo computacional, como del algoritmo en si, por lo tanto abre una puerta a la investigación y a la contribución de nuestra parte. Se hace uso de un GA, ya que es un método de optimización iterativa que permite tener diferentes candidatos de solución explorando varias zonas de búsqueda en un espacio y combinando soluciones de buena calidad, esperando así obtener la mejor [16].

2.1.1. Imagen

Una imagen es una proyección en dos dimensiones de una escena tridimencional capturada por un sensor. Matemáticamente podemos ver a una imagen como una función espacial $f(x, y)$, dónde (x, y) representa una coordenada espacial y f el valor en esa coordenada; también puede tener una representación matricial $A(i, j)$ (filas, columnas) y un valor l que representa el valor de luminosidad que está en el intervalo $\{0, 1, ..., 255\}$ para imágenes de 8 bits que son las más comunes [2].

2.1.2. ¿RGB o escala de grises?

Una imagen en escala de grises representa el valor de luminosidad l en cada punto (2D) de una escena tridimencional; por otro lado una imagen a color representa esa luminosidad y la información de color dentro de dicha escena. Las imágenes a color, RGB específicamente, son representadas por 3 matrices de imagen, una para cada canal de color: rojo, verde y azul [2], un ejemplo de cada uno de ellas lo podemos ver en la Figura 2.1. En visión por computadora se puede trabajar tanto con imágenes en escala de grises o imágenes RGB, cada una de ellas tiene ventajas y desventajas tal como lo vemos en la Tabla 2.1.

Figura 2.1: Imagen RGB (izquierda), imagen en escala de grises (derecha).

Tabla 2.1: Comparación entre espacios de color

Representación de pixeles	Ventajas	Desventajas
Espacio RGB	▪ Permite extraer mayores características ▪ Mayor niveles de segmentación	▪ Tiempo de ejecución mayor para procesamiento ▪ Complejidad computacional mayor
Escala de grises	▪ Complejidad computacional menor ▪ Tiempos de ejecución menor para procesamiento	▪ Menor cantidad de características a extraer

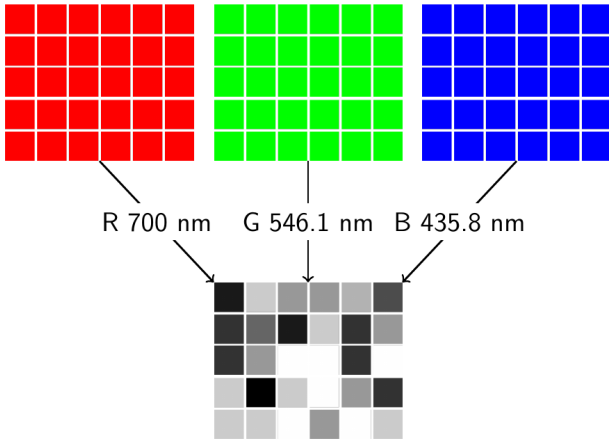


Figura 2.2: Representación escala de grises a partir de información en espacio de color RGB [2]

Una cámara digital captura usualmente imágenes en espacio RGB, pero computacionalmente es más sencillo trabajar en escala de grises, es por ellos que se necesita convertir esa imagen RGB a escala de grises (Figura 2.2). Para realizar esta transformación encontramos en la literatura dos métodos: el método de promediación se muestra en la Ecuación 2.1 y el método estandarizado está descrito por la Ecuación 2.2, la razón de este ajuste se muestra en la Figura 2.3, basado en la sensibilidad para captar los colores del sistema visual humano.

$$A_{gs} = \frac{R + G + B}{3} \tag{2.1}$$

$$A_{gs} = 0.299R + 0.587G + 0.114B \tag{2.2}$$

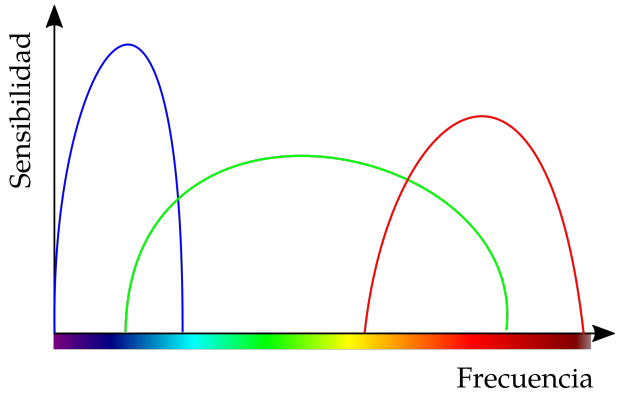


Figura 2.3: Curva de sensibilidad espectral. [2]

2.1.3. Segmentación

Umbralización

Una imagen binaria $B(i,j)$ es creada de una imagen en escala de grises $A(i,j)$ usando como punto de decisión un valor de umbral T, el cual se se describe en la Ecuación 2.3 [2]:

$$B(i,j) = \begin{cases} 1, & \text{Si } A(i,j) \geq T \\ 0, & \text{cualquier otro caso} \end{cases} \tag{2.3}$$

Cuando usamos imágenes codificadas en 8 bits el valor de T puede estar en el intervalo $\{0, 1, ..., 255\}$, por otro lado si usamos imágenes codificadas en 16 bits este intervalo será $\{0, 1, ..., 65535\}$ lo cual nos puede dar una mayor profundidad de tonos de gris, pero a su vez esto aumenta el tiempo de procesamiento.

Utilizar un solo valor de umbral o encontrar cuál es el valor más adecuado entre todo el intervalo de valores disponibles, es una de las principales tareas de PDI. Para encontrar un umbral adecuado se puede utilizar el procedimiento propuesto por Singh et al. [4] descrito en el Algoritmo 1, los resultados obtenidos se muestran en la Figura 2.4.

Figura 2.4: Ejemplo umbral adecuado implementado, $T = 120$

Otra opción para encontrar un umbral óptimo es usar el método propuesto por Otsu en 1979 [17]; Otsu supone que el histograma de imagen es la suma de dos distribuciones normales y propone seleccionar el valor de umbral T que minimice la varianza dentro de la clase (primer plano/fondo) $\sigma_W^2(T)$:

$$\sigma_W^2(T) = w_f(T)\sigma_f^2(T) + w_b(T)\sigma_b^2(T) \tag{2.4}$$

Donde $w_f(T)$ y $w_b(T)$ son las porciones que corresponden al primer plano/fondo, y $\sigma_f^2(T)$ y $\sigma_b^2(T)$ son las varianzas del primer plano/fondo en valores de escala de grises. Los resultados de su implementación usando OpenCV los podemos apreciar en la Figura 2.5.

Figura 2.5: Implementación umbral óptimo usando el método de Otsu, $T = 120$

Detección de bordes

La detección de bordes es ampliamente utilizada en PDI, debido a que ser capaz de separar automáticamente, objetos del fondo no es una tarea sencilla. Un borde puede ser considerado como la frontera entre un objeto y el otro, o entre un objeto y el fondo. Técnicamente, la detección de bordes es detectar los pixeles que correspondan a un borde [13]. La tarea de detectar bordes, se puede modelar como un proceso de filtrado. El filtro Sobel es comúmente utilizado para realizar este proceso, el cual consiste en convolucionar en ventanas los siguientes kernel con una imagen de entrada:

$$s_h = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \quad s_v = \begin{bmatrix} 1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

El filtro Sobel incorpora un suavizado dentro de la derivada parcial de los kernels de convolución [2], y su implementación se muestra en la Figura 2.6.

John Canny, uno de los pioneros en esta área, propone un detector de bordes conocido como “Canny Edge Detector” (CED) [18]. El CED tiene como meta identificar bordes de manera eficiente y describe un modelo óptimo para lograrlo; pero también presenta tres problemas primordiales: el rango de error, la localización y la respuesta, siendo este último el

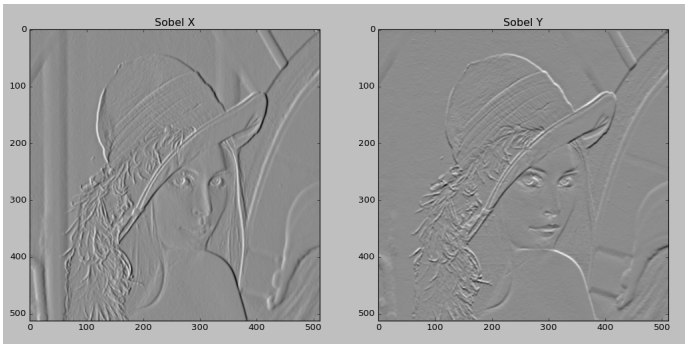


Figura 2.6: Implementación procedimiento Sobel

que describe que no se pueden detectar múltiples bordes en un área pequeña. El algoritmo que utiliza para detección de bordes es el siguiente:

1. Leer imagen de entrada $I(x, y)$
2. Crear una máscara Gaussiana unidimensional $G(\sigma)$ para convolucionar con I. Siendo σ un parámetro de entrada para la máscara.
3. Crear una máscara Gaussiana unidimensional en x e y direcciones; llamar a ellas G_x y G_y, usando el mismo valor de σ del paso 2.
4. Convolucionar I con G a lo largo de las filas para obtener el componente de imagen x llamado I_x, y hacia abajo de las columnas para dar el componente de imagen y llamado I_y.
5. Convolucionar I_x con G_x, para obtener I'_x, y convolucionar I_y con G_y, para obtener I'_y
6. Calcular la magnitud de los bordes usando la Ecuación 2.5:

$$M(x, y) = \sqrt{I'_x(x, y)^2 + I'_y(x, y)^2} \tag{2.5}$$

Un ejemplo de este procedimiento implementado usando OpenCV se muestra en la Figura 2.7.

Slatnia et al. [3] usan un proceso evolutivo para buscar reglas para un autómata celular usando un GA, todo este proceso encuentra bordes de imágenes, los resultados son mostrados en la Figura 2.8

Por otro lado Singh et al. en el 2008 [4] proponen un detector de bordes usando la teoría de entropía de Shannon, el proceso completo se divide en dos partes: primero, ellos obtienen una imagen binaria usando un método de umbralización descrito en el Algoritmo 1, y posteriormente encuentran los bordes usando un kernel de tamaño 3x3, y usando entropía localizan pixeles de borde como se muestra en la Figura 2.9

En la siguiente sección explicaremos los conceptos de metaheurística, sus clasificaciones, algunos ejemplos y cómo se relacionan con el PDI.

Figura 2.7: Ejemplo del CED implementado

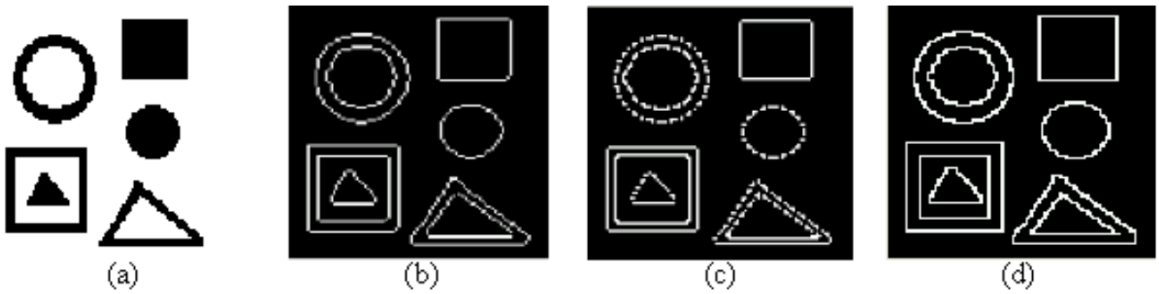


Figura 2.8: (a) imagen de entrada (b) Resultados con CED (C) Método alternativo (d) Resultados obtenidos por Slatnia et al. [3].

Algoritmo 1: Algoritmo para encontrar un umbral adecuado

Entrada: T: Un valor inicial de umbral de entrada
Entrada: $A(i, j)$: imagen de entrada en escala de grises
Salida : T: El valor nuevo de umbral de salida
mientras $T_n \neq T_v$ **hacer**
  $\mu_1 \leftarrow$ el promedio de todos los pixeles $A(i, j) > T_n$
  $\mu_2 \leftarrow$ el promedio de todos los pixeles $A(i, j) \leq T_n$
  $T_v \leftarrow T_n$
  $T_n \leftarrow (\mu_1 + \mu_2)/2$
fin
$T \leftarrow T_n$
regresar T

Figura 2.9: Imagen orginal (izquierda), Resultados implementando el procedimiento de Singh et al. [4] (derecha).

2.2. Metaheurísticas en PDI

Un problema de optimización complejo puede describirse como un problema que no puede garantizar una frontera de solución por ningún método determinístico en un tiempo límite razonable. Estos problemas pueden tener diversas clasificaciones, desde continuos o discretos, restringidos o no restringidos, mono o multiobjetivo; por esta razón se sugiere el uso de algoritmos metaheurísticos, ya que esta clase de algoritmos son diseñados para resolver un problema complejo de optimización, de manera aproximada en un intervalo de tiempo razonable. Una de sus caráctersisticas es la de evaluar una solución o un conjunto de soluciones a través de una función objetivo establecia, hacen uso de la aleatoriedad par construir dichas soluciones y para explorar en el espacio de búsqueda dónde se encuentran éstas soluciones; otra característica que podemos restaltar es que tienen un parámetro de parada cuando se cumple un criterio o cuando la mejor solución durante este proceso ya no se puede mejorar. Su etimilogía viene del prefijo griego “meta” que significa mas allá, y “eureka” que significa, hallar o inventar [19]. La mayoría de las metaheurísticas comparten las siguientes características:

- Son bioinspirados (imitan comportamiento de la naturaleza)
- Usan componentes estocásticos (hacen uso de la aleatoriedad)
- No hacen uso de operadores diferenciales o Matrices Hessianas en la función objetivo
- Tienen varios parámetros que necesitan ser adaptados al problema específico

Dentro de las metaheurísticas encontradas en la literatura podemos observar una clasificación clara: de una sola solución y de población de soluciones, tal como se muestra en la Figura 2.10. En esta sección explicaremos cuatro de las metaheurísticas mencionadas en la Figura 2.10, las cuales se encontró en la literatura su acercamiento a PDI.

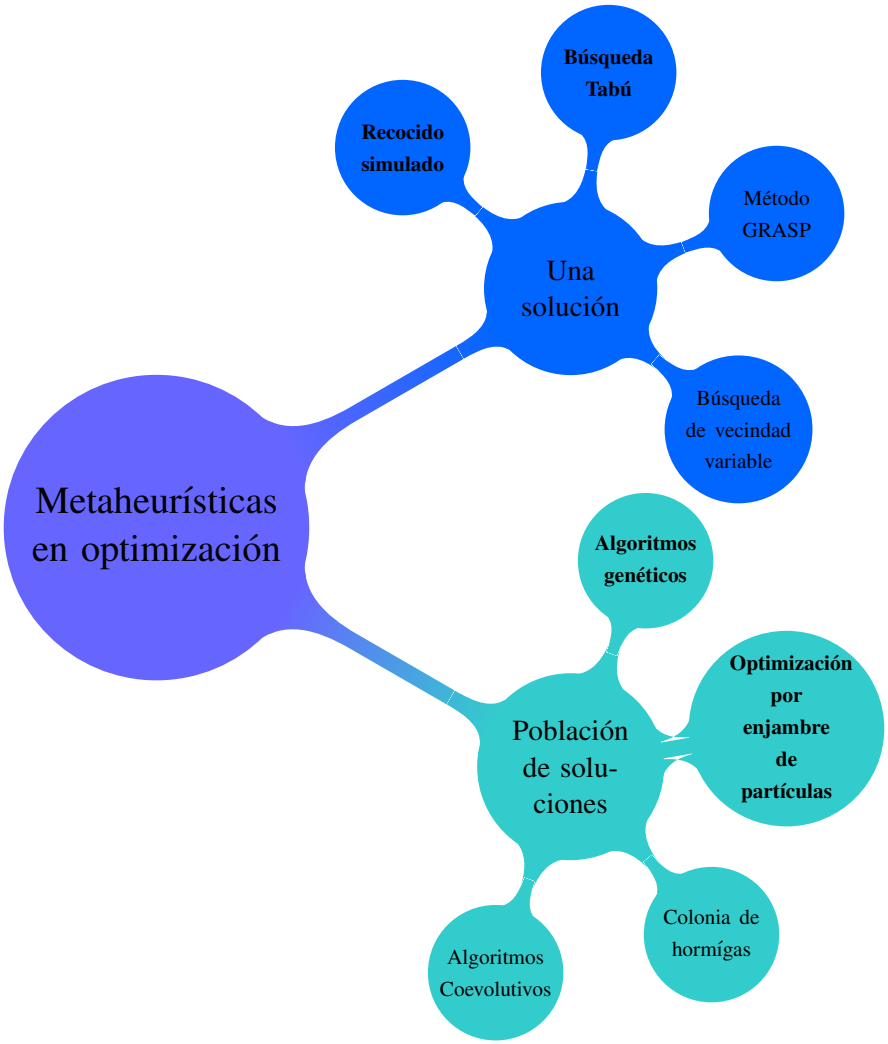


Figura 2.10: Clasificación de algunas Metaheurísticas según la cantidad de soluciones procesadas

2.2.1. Recocido simulado

El recocido simulado (SA, por su sigla en inglés Simulated Annealing) es un método que está basado en la aleación de metales, el cual consiste en llevar un material a altas temperaturas y bajar esa temperatura lentamente. SA propone en imitar este procedimiento de aleación a la solución de un problema de optimización: la función objetivo del problema, de igual manera que la energía de un material, es luego minimizada introduciendo una temperatura T, el cual es un parámetro simple y controlable del algoritmo. Dicho algoritmo inicia generando una solución inicial (ya sea aleatoriamente o usando una heurística) e inicializando el parámetro T; luego en cada iteración es seleccionada de manera aleatoria una solución s' en la vecindad $N(s)$ de la solución actual s. La solución s' es aceptada como la nueva solución dependiendo del valor de T y los valores de la función objetivo $f(s)$ y $f(s')$, respectivamente, si $f(s') \leq f(s)$ entonces s' reemplaza a s. Por otro lado si $f(s') > f(s)$, s' también puede reemplazar a s con una probabilidad $p(T, f(s'), f(s)) = e^{-\frac{f(s')-f(s)}{T}}$. La temperatura T decrece conforme el proceso avanza, de tal manera que la probabilidad de aceptación inicia alta y decrece junto con dicho parámetro [19]. El procedimiento es explicado en el Algoritmo 2.

Algoritmo 2: Algoritmo general del SA

Entrada: $T \rightarrow$ Valor inicial de temperatura
Entrada: $s \rightarrow$ Solución inicial
Salida : $s \rightarrow$ La mejor solución encontrada
mientras *el criterio de parada no se cumple* **hacer**
  repetir
    Aleatoriamente seleccionar $s' \in N(s)$
    si $f(s') \leq f(s)$ **entonces**
      $s \leftarrow s'$
    en otro caso
      $s \leftarrow s'$ con la probabilidad $p(T, f(s'), f(s))$
    fin
  hasta *El "equilibrio de termodinámica"del sistema es alcanzado*
  Decrementar T
fin
regresar s

Para mostrar un ejemplo de la implementación de SA en PDI podemos observar el trabajo de Tan et al. [20], en dónde ellos proponen un modelo de optimización, y transportan la detección de bordes a un modelo de este tipo. Dan la descripción matemática de lo que es un borde y se centran en cuatro características deseables que un borde debería tener: Localización precisa, Continuidad (esto permitirá establecer una frontera), Longitud (3 pixeles de largo). De esta manera establecen una función de costo para evaluar bordes.

2.2.2. Búsqueda Tabú

La Búsqueda Tabú (TS, por su sigla en inglés Tabu Search) fue introducido por Glover en 1986 [21]. Se desarrolló con la intención principal de manejar algoritmos de búsqueda local. Usa explícitamente la historia de la búsqueda, para escapar de mínimos locales e implementar una estrategia de exploración; su idea fundamental es aprender del pasado, tal como lo hace la humanidad, a diferencia del SA, TS usa una memorial local para almacenar los puntos visitados [19]. El Algoritmo 3, explica el proceso general del funcionamiento de TS.

En la búsqueda de literatura solo se encontró una investigación donde se uso de esta metaheurística en PDI. Jiang et al [5] presentan un trabajo de segmentación de células, donde proponen una TS con un método evolutivo (ETS por sus siglas en inglés, Evolutionary Tabu Search). Ellos presentan un modelo de optimización en el cual ajustan puntos a la frontera de una elipse, creando poblaciones de soluciones, empleando una estrategia evolutiva y el uso TS logran encontrar y segmentar células de una imagen. Los resultados se muestran en la Figura 2.11.

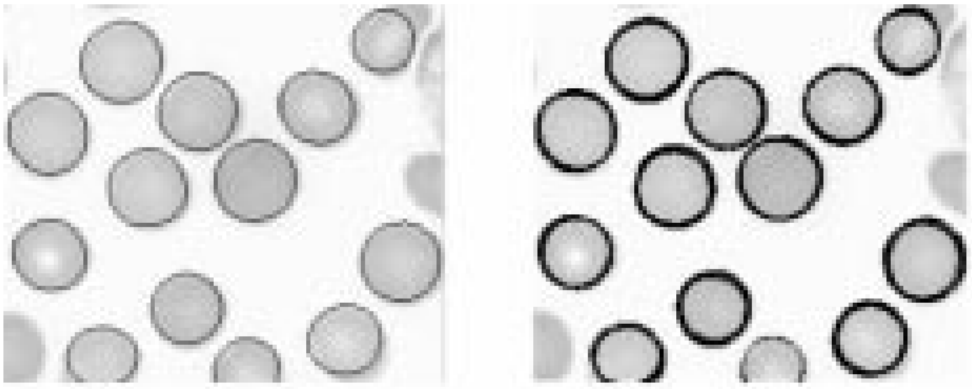


Figura 2.11: Imagen de entrada (izquierda), método ETS (derecha) [5]

Algoritmo 3: Algoritmo general del TS

Entrada: $s \rightarrow$ Solución inicial elegida aleatoriamente
Salida : La mejor solución encontrada
$ListaTabu \leftarrow \emptyset$
mientras *el criterio de parada no se cumple* **hacer**
  Seleccionar la mejor solución $s' \in N(s)|ListaTabu$
  $s \leftarrow s'$
  Actualizar $ListaTabu$
fin
regresar *La mejor solución encontrada*

2.2.3. Optimización por enjambre de partículas

La Optimización por Enjambre de Partículas (PSO, por su sigla inglés Particle Swarm Optimization), fue introducido por Kennedy y Eberhart en 1995 como una técnica global de optimización [22]. Es un algoritmo bioinspirado que trata de imitar el comportamiento de las aves para resolver problemas de optimización. En PSO numerosas entidades llamadas partículas son creadas estocásticamente en un espacio de búsqueda. Cada partícula es una solución candidata del problema, y es representada por una velocidad, una posición y cada una de ellas tiene una memoria propia, la cual almacena su mejor posición previa. El conjunto de partículas i está topológicamente conectado y esto es llamado vencindario. El vecindario podría ser toda la población de partículas o un subconjunto de ellas [19].

En la inicialización del PSO, las posiciones y velocidades de cada partícula son iniciadas comúnmente de manera aleatoria. La velocidad define la dirección y la distancia que la partícula recorre, y se actualiza de acuerdo a la siguiente Ecuación:

$$V_{id}(t+1) = V_{id}(t) + C_1\phi_1(P_{id}(t) - X_{id}(t)) + C_2\phi_2(P_{gd}(t) - X_{id}(t)) \tag{2.6}$$

Dónde $i = 1, 2, ..., N$, N es el tamaño del enjambre; ϕ_1 y ϕ_2 son los dos números aleatorios creados de una distribución uniforme entre $[0, 1]$, C_1 y C_2 son términos múltiplicadores constantes conocidos como *coeficientes de aceleración*. La posición debe también actualizarse y se representa con la siguiente ecuación:

$$X_{id}(t+1) = X_{id}(t) + V_{id}(t+1) \tag{2.7}$$

El proceso general de ésta metaheurística es descrito en el Algoritmo 4.

Algoritmo 4: Algoritmo general del PSO

```
Inicializar la población de partículas con posiciones aleatorias y velocidades en D
  dimensiones en el espacio de búsqueda
mientras el criterio de parada no se cumple hacer
    para cada partícula i hacer
        Adaptar velocidad de partícula usando Ecuación 2.6
        Adaptar la posición de cada partícula usando Ecuación 2.7
        Evaluar fitness f(X_i)
        si f(X_i) < f(P_i) entonces
            P_i ← X_i
        fin
        si f(X_i) < f(P_g) entonces
            P_g ← X_i
        fin
    fin
fin
```

Gao et al. [6] presentan un trabajo donde realizan una segmentación via umbrales utilizando un PSO mejorado que llaman PSO con comportamiento cuántico (QPSO, por sus siglas en inglés Quantum Particle Swarm Optimization). La diferencia radica en que usan una ecuación estocástica con la intención de tener una convergencia mucho más rápida. Utilizan la teoría umbralización por Otsu y comparan los resultados obtenidos contra él para obtener un nivel de umbral. Los resultados son presentados en la Figura 2.12.

Figura 2.12: Imagen de entrada (izquierda), Segmentación por Otsu (centro), método QPSO (derecha) [6]

2.2.4. Algoritmos Genéticos

Los Algoritmos Genéticos (GA, por su sigla en inglés Genetic Algorithms) son algoritmos de optimización numérica inspirados por la selección natural y la genética natural [23]. Fue introducido por primera vez por John Holland en la decada de los 70's en la universidad de Michigan [24]. Un GA maneja básicamente cuatro estrategias de evolución: *representación* de una solución, una estrategia de *selección*, tipo de cruzamiento (*crossover*) y un operador de *mutación* [19]. El proceso general de un GA es el siguiente [23]:

1. Una población de soluciones o suposiciones de la solución del problema
2. Una manera de calcular qué tan bueno o malo es la solución dentro de la población (llamada función de aptitud, por su traducción al inglés *fitness*, dicho término se usará en inglés durante todo este documento)
3. Un método para mezclar fragmentos de las mejores soluciones para formar nuevas
4. Un operador de mutación para evitar pérdidas de diversidad en las soluciones

¿Pero porqué usar un GA en lugar de un método tradicional o uno exacto? La respuesta radica en que han probado ser capaces de resolver problemas complejos en espacios de búsqueda grandes, donde estos métodos tradicionales no son capaces de resolverlos en un tiempo razonable. La otra característica en comparación a otras metaheurísticas es la capacidades de escapar de óptimos locales solo por hacer el proceso explicado anteriormente, y

no necesita de una estrategia para escapar de ellos. Entre los tantos problemas prácticos en las diferentes áreas dónde los GA han dado resultados satisfactorios se pueden mencionar los siguientes [23]: *Procesamiento digital de imágenes*, Tecnológia láser, Física de estado sólido, Reglas de autómata celular evolutivo, Redes de computadoras, Reconocimiento facial, entre muchos otros. El proceso general de un GA está descrito en el Algoritmo 5 [9].

Algoritmo 5: Algoritmo general del GA

Entrada: $P_m \rightarrow$ probabilidad de mutación
Entrada: $P_c \rightarrow$ probabilidad de cruzamiento
Entrada: $P_s \rightarrow$ tamaño de la población
Salida : $B_i \rightarrow$ el mejor individuo
$gen \leftarrow 0$
$Pop(gen) \leftarrow$ inicializar aleatoriamente la población
evaluar el *fitness* para cada individuo en $Pop(ge)$
mientras *el criterio de parada no se cumple* **hacer**
  $gen \leftarrow gen + 1$
  seleccionar $Pop(gen)$ de $Pop(gen - 1)$ basado en el *fitness*
  aplicar operadores genéticos a $Pop(gen)$
  evaluar *fitness* a cada individuo en $Pop(gen)$
fin
$B_i \leftarrow$ el mejor individuo de Pop
regresar B_i

Siendo el PDI el de mayor interés para nosotros, podemos mencionar por ejemplo el caso de Ayala et al. [7] en el que presentan un detector de círculos usando procesamiento de imágenes y el uso de GA para el ajuste de círculos sintéticos a un mapa de bordes obtenido de una imagen en escala de grises usando un filtro sobel. El filtro sobel lo implementan en MATLAB y el resto del proceso es implementado en C++. Sus resultados se muestran en la Figura 2.13.

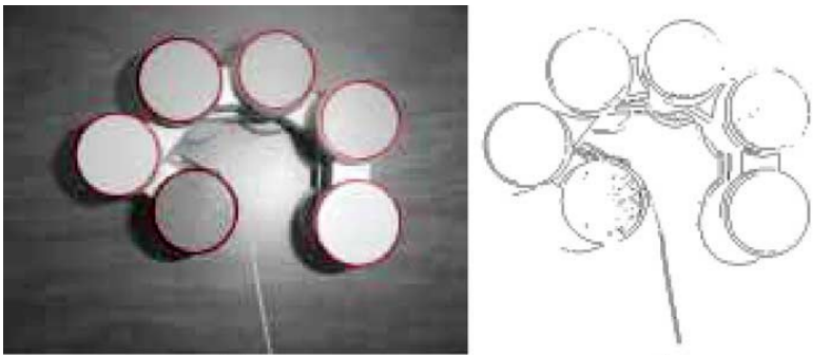


Figura 2.13: Imagen de entrada y círculos encontrados sobrepuestos (izquierda), mapa de bordes (derecha) [7]

Por otro lado Abdel et al. [8] presentan una técnica de segmentación basada en entropía y el uso de un GA; el método propone una novedosa segmentación de imágenes en dos dimensiones, basadas en la Entropía de Tsallis y Renyi [25,26], el uso del GA se hace cuando se trata de maximizar esa entropía con la intención de separar de manera adecuada los objetos en la imagen del fondo. Los resultados reportados se presentan en la Figura 2.14:

Figura 2.14: Imagen de entrada (izquierda), entropía de Renyi (centro), entropía de Tsallis (derecha) [8]

Para resumir esta sección, la Tabla 2.2 presenta una comparación acerca de las ventajas y desventajas de las cuatro metaheurísticas explicadas anteriormente.

2.2.5. Algoritmos Coevolutivos

Este paradigma de cómputo evolutivo se ha dejado fuera de la clasificación anteriormente mencionada con la intención de hacer una explicación más detallada ya que es el paradigma empleado en esta investigación. Los Algoritmos Coevolutivos (CoEA, por sus siglas en inglés, Coevolutionary Algorithms) se introdujeron a principios de los 90's, por Hillis [35], en clasificación de redes. A diferencia de un proceso convencional de cómputo evolutivo, cada individuo aparte de ser evaluado independiente, utiliza una función de *fitness* subjetiva, debido a que depende no solo de su evaluación individual sino también de su interacción con otros individuos [19].

Hay muchas variantes de CoEA, pero todas estas pueden resumirse en dos categorías [19]:

- **Coevolución competitiva:** en esta clasificación las poblaciones imitan el comportamiento de la naturaleza donde el más fuerte prevalece, usando el concepto de presa y depredador. De tal manera que existen dos poblaciones en escencia para imitar estos comportamientos. Con esta idea los menos fuertes son forzados a desaparecer o adaptarse de tal manera que lleguen a un punto donde puedan sobrevivir durante el proceso de evolución.

- **Coevolución cooperativa:** en esta clasificación en específico trata de imitar la relación ecológica de diferentes especies viviendo juntas en un proceso de beneficio mutuo. Potter y De Jong [9] proponen una metodología, la cual consiste en descomponer un problema complejo en un conjunto de subproblemas más sencillos. A cada subproblema se le asigna una población y cada inidividuo es evaluado pero a su vez interactuan entre ellos para obtener un fitness de esa población.

Tabla 2.2: Comparación entre metaheurísticas

Sigla	Ventajas	Desventajas	Referencia
SA	▪ Fácil implementación ▪ Adaptable a otras metaheurísticas ▪ Costo computacional bajo ▪ Se usa en segmentación de imágenes	▪ Estancamiento en óptimos locales ▪ Sensible a la función objetivo ▪ Procesa una sola solución	[19, 20, 27–29]
PSO	▪ Adaptable a otras metaheurísticas ▪ Mayor adaptación a problemas más complejos ▪ Procesa poblaciones de soluciones	▪ Computacionalmente es costoso ▪ Por su robustes, presenta mayores retos al momento de implementarlo	[19, 30–32]
TS	▪ Adaptable a otras metaheurísticas ▪ Su escencia es evitar caer en óptimos locales ▪ Tiene opción de expandir el espacio de búsqueda	▪ Su costo dependerá directamente de la longitud de la lista tabú ▪ Por lo general no funciona solo, necesita un método paralelo para poder llevarlo a cabo. ▪ Sus aplicaciones en PDI son pocas	[5, 19, 21]
GA	▪ Simple implmentación ▪ Su escencia es evitar caer en óptimos locales ▪ Exploración de espacios de búsqueda amplios ▪ Procesa poblaciones de soluciones	▪ Su costo dependerá directamente de la longitud de la población y la cantidad de generaciones ▪ Sensible a a la función de fitness	[7, 8, 19, 33, 34]

La diferencia entre ambas clasificaciones radica en la manera en que descomponen los subproblemas empleados en la segunda categoría. A continuación explicaremos a detalle la metodología empleada por Potter et al. [9].

En este paradigma, podemos asignar *fitness* a cada individuo o una subpoblación reuniendo todos los valores de cada individuo, y la forma en que interactuan entre ellos, de tal manera que la función de *fitness* evalua a cada individuo de una especie en particular, estimando qué tan bien coopera con otras subespecies para producir buenas soluciones [9]. El algoritmo general de un CoEA se muestra en el Algoritmo 6.

Algoritmo 6: Algoritmo general del CoEA

$gen = 0$
para cada *especie* **hacer**
  $Pop_s(gen) =$ inicializar aleatoriamente la población
  evaluar el *fitness* para cada individuo en $Pop_s(ge)$
fin
mientras *el criterio de parada no se cumple* **hacer**
  $gen = gen + 1$
  seleccionar $Pop_s(gen)$ de $Pop_s(gen-1)$ basado en el *fitness*
  aplicar operadores genéticos a $Pop_s(gen)$
  evaluar *fitness* a cada individuo en $Pop_s(gen)$
fin

Potter et al. [9] muestran resultados sobre una función de *fitness* dada y lo comparan con un el proceso de un GA simple. Los parámetros de entrada de ambas implementaciones están dadas en la Tabla 2.3, y la función de *fitness* está dada por Ecuación 2.8. La gráfica de resultados se muestran en la Figura 2.15.

$$f(x) = 3n + \sum_{i=1}^{n} x_i^2 - 3\cos(2\pi x_i) \tag{2.8}$$

Donde $n = 20$ y $-5.12 \leq x_i \leq 5.12$.

Tabla 2.3: Características del proceso evolutivo

Característica	Valor
Representación	binario (16 bits)
Selección	proporcional al *fitness*
Escala de *fitness*	técnica de escalamiento por ventana (ancho de 5)
Estrategia de élites	preservación del mejor inidividuo
Operadores genéticos	2 puntos *crossover*, cambio de bit por mutación
Probabilidad de mutación	1/longitud del cromosona
Probabilidad de cruzamiento	0.6
Tamaño de población	100

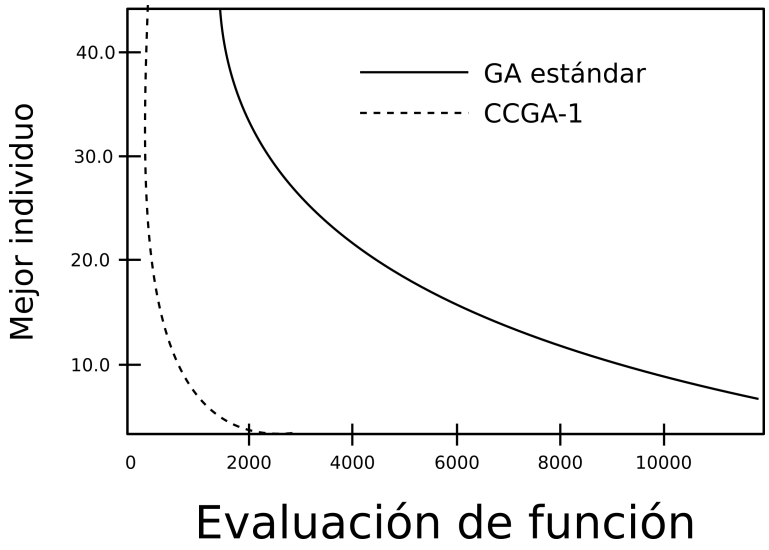


Figura 2.15: Gráfica de convergencia CCGA (Cooperative Coevolutionary Genetic Algorithm) [9]

2.2.6. Algoritmo Genético Paralelo - Modelo de islas

En este paradigma la idea básica es dividir la población total en n subpoblaciones, y a cada subpoblación creada se le aplica el proceso evolutivo básico de un GA, la característica fundamental de este paradigma radica en que cada cierto número de generaciones, se efectua un cambio de información entre subpoblaciones, dicho proceso se le conoce como *emigración*. A este proceso de emigración se le añade un nuevo parámetro el cual es la tasa de migración, el cual se refiere a la cantidad de invidividuos permitidos que pasarán de una isla (subpoblación) a otra. La literatura menciona que este parámetro está vinculado directamente con la convergencia del problema, por lo cual no se debe elegir este parámetro de forma arbitraria. Para la comunicación entre islas hay varias clasificaciones, de las cuales se pueden englobar en los siguientes 3 tipos:

a) **Comunicación estrella:** La cual consiste en nombrar a una subpoblación amo, y el resto esclavas, la regla dice que: existirá migración de cualesquiera de las islas esclavas al amo y viceversa, pero no entre las esclavas.

b) **Comunicación en red:** Este esquema se refiere a que todas tienen el mismo tipo de clasificación y pudiese existir migración entre todas.

c) **Comunicación tipo anillo:** Se asigna un identificador a cada isla y solamente podrá haber migración entre los dos adyacentes a cada subpoblación.

Todo esto lo podemos ver gráficamente en la Figura 2.16. Este modelo de islas fue implementado primeramente por Whitley et al. [36].

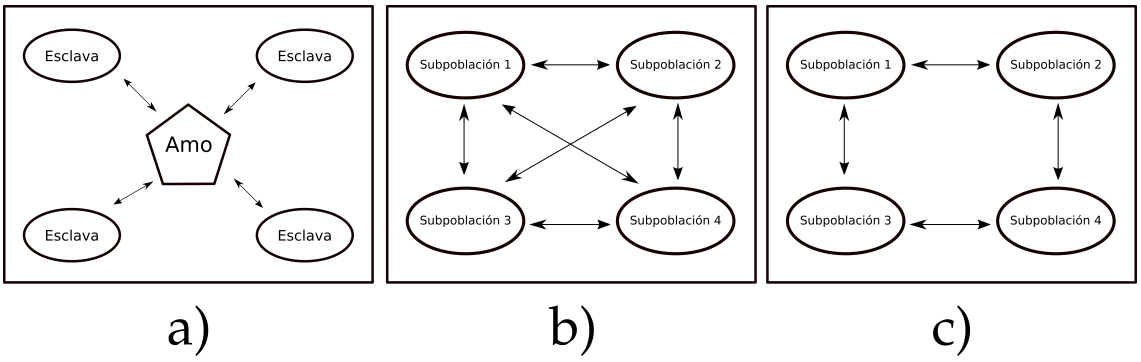


Figura 2.16: Descripción gráfica de la comunicación entre islas

2.3. Conteo de peces

El conteo de peces o larvas de peces usando un sistema de visión por computadora, no es un tema nuevo, sin embargo no es problema resuelto; esto se debe a que las condiciones y la variedad de especies, hacen que no sea un solo problema si no una variedad de problemas que a su vez pueden ser resueltos de diferentes maneras.

Toh et al. [10], presentan un método simple de conteo el cual consta de una primera etapa de preprocesado, donde eliminan ruido y el fondo de tal manera que tratan de quedarse con puras "manchas"tras un proceso de umbralización, dichas manchas que correspondan únicamente a un pez, la finalidad de ellos es poder contar "Feeder fish"de manera certera, para que esto pueda ser usado como una aplicación, tal como se muestra en la Figura 2.17.

Figura 2.17: Imagen capturada por Toh et al. [10]

Cabe mencionar que ellos hacen un montaje para hacer esta captura de videos, de esta manera tienen un ambiente controlado y pueden extraer los cuadros (*frames*) del video y procesarlos de manera independiente. El diagrama del montaje se presenta en la Figura 2.18.

Una vez que obtienen la imagen que ellos llaman la "imagen de manchas"lo que hacen es calcular el área de cada mancha, y calcular la mediana de los valores de las áreas y usar un

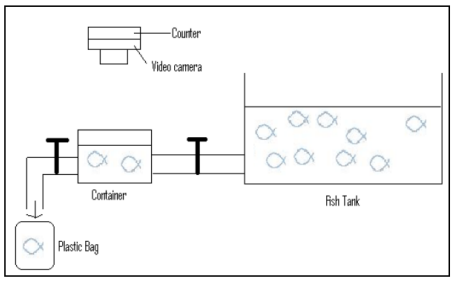


Figura 2.18: Diagrama del montaje de Toh et al. [10]

par de umbrales de áreas y para tomar la decisión a partir de esos valores ($1.4 * mediana < area < 5 * mediana$). Presentan una efectividad de entre 96 % y 100 %.

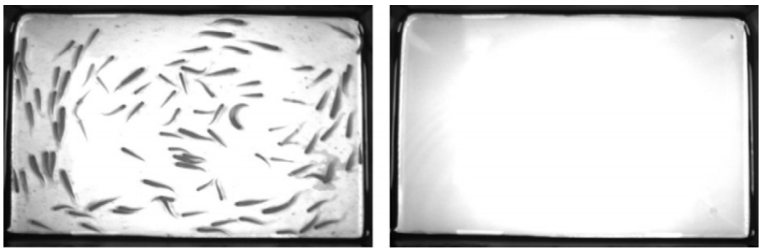


Figura 2.19: Imágenes analizadas por Fan et al. [11]

Un método novedoso es el que presentan Fan et al. [11], usando una Máquina de Soporte Vectorial por Mínimos Cuadrados (LS-SVM, por sus siglas en inglés Least-Square Support Vector Machine) y una Red Neuronal con Retropropagación (BPNN, por sus siglas en inglés Back-Propagation Neural Network) presentan un modelo de clasificación. Analizan 7 características geométricas y con ello obtienen resultados acertados; tratan de solucionar el problema de traslape en los peces, caracterizando los posibles traslapes (Figura 2.20) con 600 subimágenes, entrenando al SVM únicamente con 300. Un ejemplo de las imágenes analizadas por ellos se muestra en la Figura 2.19.

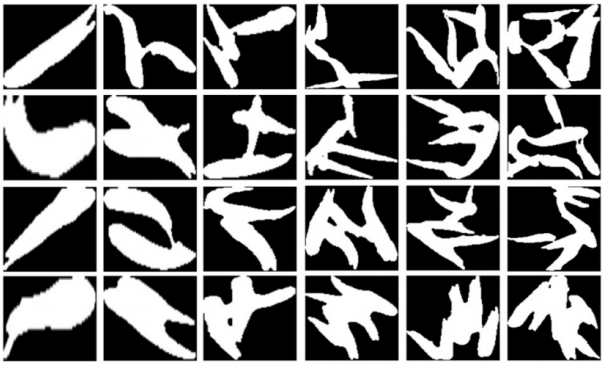


Figura 2.20: Traslapes caracterizados por Fan et al. [11]

Por otro lado Martínez [1], propone un sistema de captura con ambiente controlado, para capturar imágenes de larvas de peces (*Totoaba macdonali*). En su trabajo muestra cinco versiones del dispositivo, siendo la versión 5 la que les proporcionó los mejores resultados tal como se muestra en la Figura 2.21.

En lo que al algoritmo de conteo respecta, básicamente establecen cinco procesos principales:

1. Generación de subimágenes con medidas definidas (ventanas)
2. Encontrar un umbral óptimo en cada sub-imagen, usando el Método de Otsu y generando una imagen binaria por cada sub-imagen
3. Buscar los tres puntos que correspondan a una larva, y medir las distancias entre ellos; generar una sub-imagen rectangular con el área mínima que corresponda a cada tres puntos que cumplan las condiciones correspondientes a una larva.
4. Verificar que cada subimagen del paso anterior corresponda a una larva y extrapolar cada uno de las posiciones donde las larvas fueron encontradas en la sub-imágenes a la imagen original. Los resultados son mostrados en la Figura 2.22.

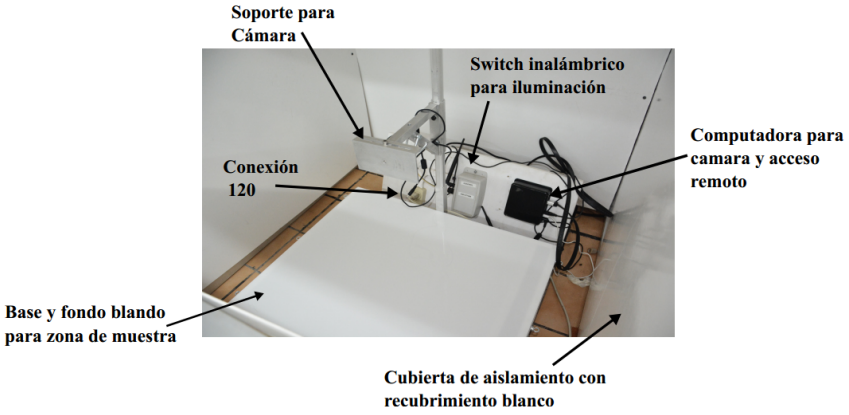


Figura 2.21: Sistema de Adquisición de Imágenes (SAI V5) [1]

Entre sus resultados él presentan tiempos de procesamiento entre los 27 y 120 segundos (45 segundos promedio) por imagen, con una exactitud de entre el 97 % y el 100 %, siendo esto uno de los retos en esta investigación, es decir, ser capaces de reducir ese tiempo y no sacrificar los resultados que Martínez reporta.

En el siguiente capítulo se muestra todo el trabajo realizado durante esta investigación, se da formalidad a cada uno de los algoritmos propuestos, se muestra como la metaheurística utilizada tiene la oportunidad de presentar excelentes resultados y por último se explica a detalle todas los retos que conlleva el procesamiento digital de imágenes.

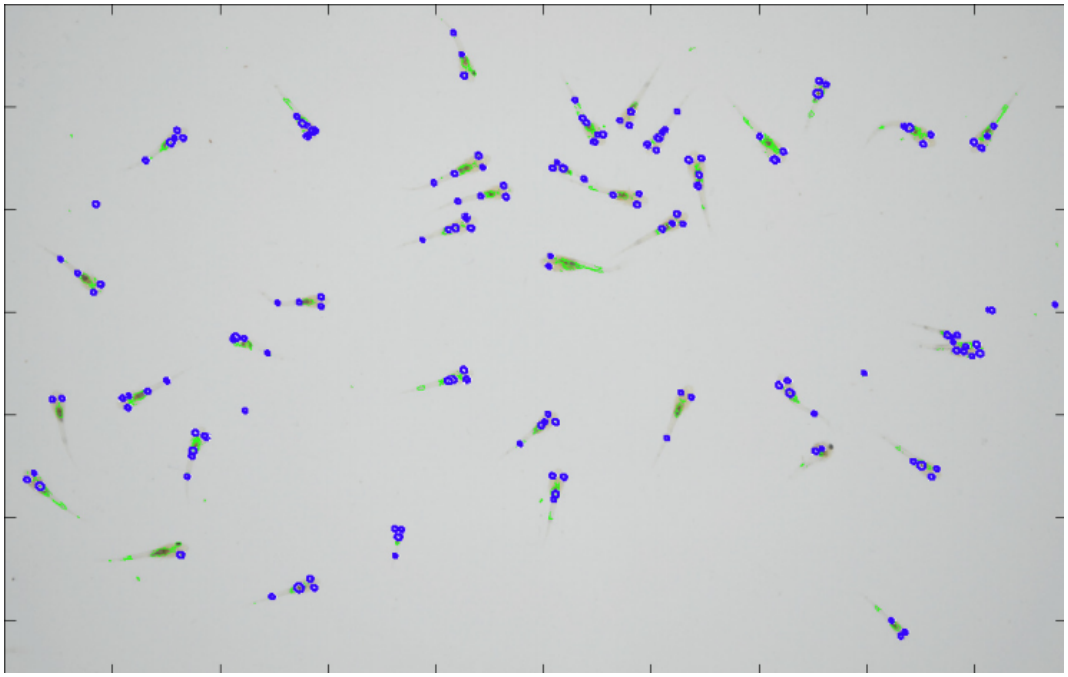


Figura 2.22: Figura procesada por el método de Martínez [1]

Capítulo 3

Desarrollo de Investigación

En este capítulo se explicará a detalle lo realizado en esta investigación. Se presentan los algoritmos propuestos, los métodos de PDI utilizados, y adaptaciones realizadas a este problema en específico. Se explica la utilización de un generador sintético de patrones para poder partir de un esquema controlado, y pasar a un esquema real, es decir, uno en el cual se trabaje con larvas de peces.

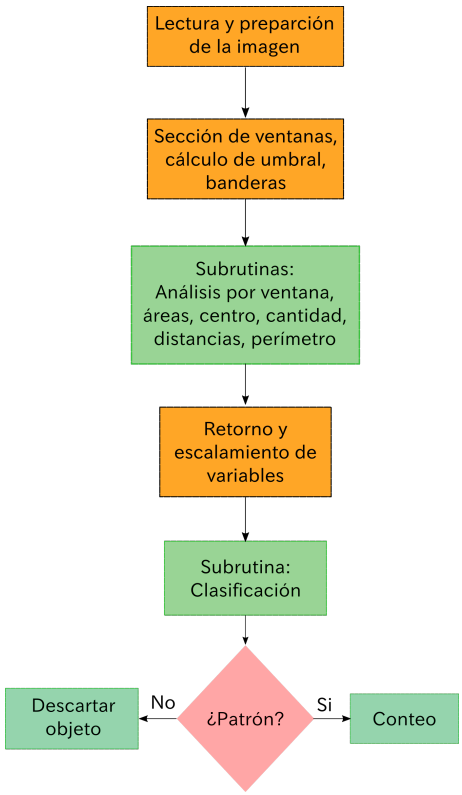


Figura 3.1: Diagrama de flujo sobre el proceso de Martínez [1]

3.1. Algoritmo propuesto por Martínez

En esta sección explicaremos de manera general el trabajo realizado por Martínez; primeramente mostraremos el proceso general que propone para la detección de larvas en un imagen, tal como se muestra en la Figura 3.1.

1) **Lectura y preparación de la imagen:**
   Se mejora el contraste para obtener mejores resultados, se carga la imagen y es convertida en escala de grises. Ese cambio de contraste se hace previo al cambio a escala de grises con un software externo.

2) **Sección de ventanas, cálculo de umbral, banderas:**
   La imagen es seccionada en una medida específica *(150 x 150 px)* y se forma un grid como el mostrado en la Figura 3.2.

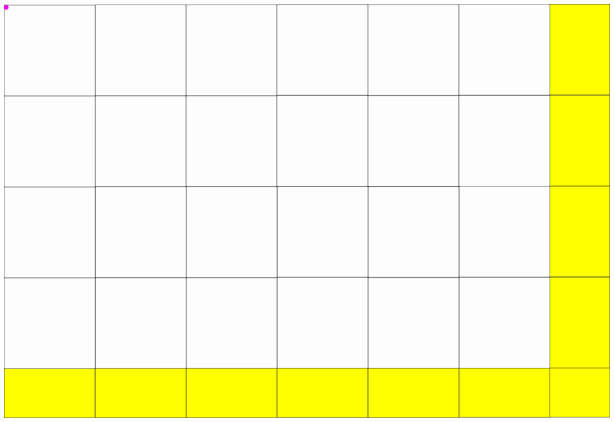


Figura 3.2: Imagen seccionada en ventanas (Zona especial).

   El recorrido se realiza de manera matricial; se calcula un umbral (usando el método de Otsu) se detecta si existe información en alguna de las ventanas, de ser así, se activa una bandera indicando que esa ventana contiene áreas de interés, generando una nueva matriz de imagen para cada ventana con una bandera activada. La zona especial se deja al final y se hace el mismo proceso. Las matrices de imagen de las ventanas se guardan binarizadas con el umbral conocido. Las medidas de las ventanas se calcularon de manera empírica, ya que un tamaño diferente daba resultados distintos; esto podría dar datos falsos en la zona especial; para ello se deberá dividir la imagen de manera uniforme.

3) **Subrutinas:**

   - **Análisis por ventana:**
     Se analiza cada una de las nuevas matrices de imagen guardadas.

- **Áreas:**
  Se define el área dónde se encuentran las zonas de interés como se muestra en las Figuras 3.3 y 3.4.

Figura 3.3: 40 px de ancho

Figura 3.4: 100 px de ancho

- **Centros:**
  Se localizan los centros de las áreas donde se identificó la información.
- **Distancias:**
  Se miden las distancias entre los centros localizados, buscando el patrón de 3 centros. Los 3 más cercanos (**Distancias I**); y todos contra todos (**Distancias II**).
- **Cantidad:**
  Se contabiliza cada patrón encontrado.
- **Perímetros:**
  Se traza una línea alrededor de las zonas de interés localizadas, tal como se muestra en las Figuras 3.5 y 3.6.

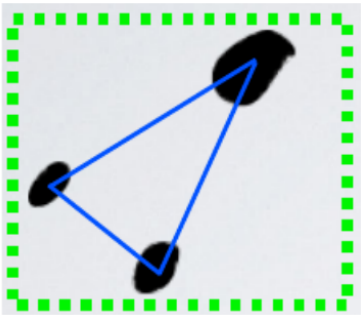


Figura 3.5: Triangulación de los centros localizados

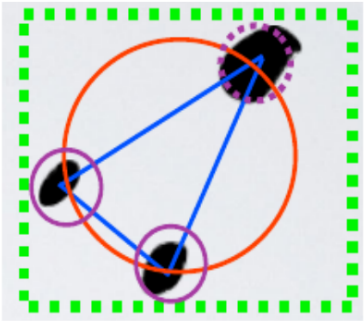


Figura 3.6: Perímetros y cálculo de distancias

Para detectar los círculos se usó la Transformada de Hough.

Transformada de Hough para círculos (por sus siglas en inglés CHT, *Circle Hough Transform*):
Usando la ecuación de la circunferencia mostrada en la Ecuación 3.1. De esta manera el círculo tiene tres parámetros r, a y b. Dónde a y b son el centro del círculo en las direcciones x e y respectivamente y r representa el radio. La representación paramétrica está dada por las Ecuaciones 3.2 y 3.3.

$$r^2 = (x-a)^2 + (y-b)^2 \tag{3.1}$$

$$x = a + r\cos(\theta) \tag{3.2}$$

$$y = b + r\sin(\theta) \tag{3.3}$$

Dónde θ es el ángulo formado entre (a, b) y (x, y). El espacio tridimensional es mostrado en la Figura 3.7; cabe mencionar que el espacio se limita al valor de entrada r que CHT ocupa para iniciar su proceso.

En escencia, toman un imagen de bordes obtenida por una imagen de entrada usando un detector de bordes (Canny, Sobel u operadores morfológicos). Otro parámetro

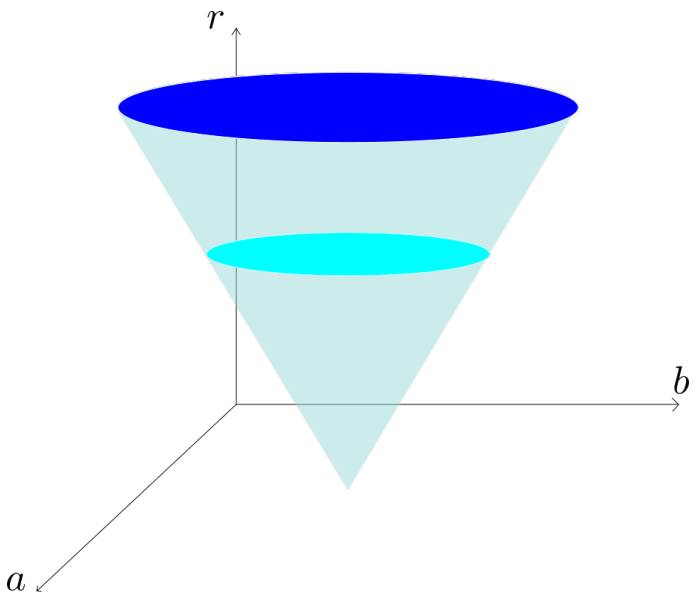


Figura 3.7: Espacio paramétrico por CHT

de entrada es un radio r deseado a encontrar, de esta manera cada punto de borde es tomado como centro de un círculo de radio r. Este círculo es dibujado en el espacio paramétrico (Figura 3.7), donde el eje x es represtado por a, el eje y por b y el eje z por r. En las coordenadas que corresponden al perímetro del círculo dibujado incrementamos el valor de la matriz de acumulación, la cual es del mismo tamaño que el espacio paramétrico. De esta manera se pasa por todos los puntos correspondientes a bordes de la imagen de entrada, dibujando círculos con el radio r, e incrementando los valores de nuestro acumulador. El acumulador contendrá los números correspondientes al número de círculos que pasan a través de las coordenadas individuales. Siendo así, los números mayores corresponden al centro de los círculos en una imágen [37].

3.2. Selección de representación de pixeles (espacio de color o escala de grises)

Como se mencionó en el capítulo anterior, el primer paso para empezar a procesar una imagen es decidir si se va a trabajar con algún espacio de color o se trabajará en escala de grises. Las imágenes a las que tenemos acceso (proporcionadas por el trabajo de Martínez [1]), se encuentran en espacio RGB; una de las razones por las cuales se decidió trabajar en *escala de grises* es que, como se mencionó anteriormente, el tiempo computacional requerido para procesar es menor en este espacio que en RGB, lo cual encaja dentro de los objetivos específicos de este trabajo. Por otro lado la complejidad se reduce y por lo tanto esto también impacta directamente al tiempo de procesamiento.

En la Sección 2.1.2 se mencionan dos Ecuaciones (2.1 y 2.2) para pasar del espacio RGB a escala de grises. En la Figura 3.8 se muestran los resultados de aplicar ambas ecuaciones.

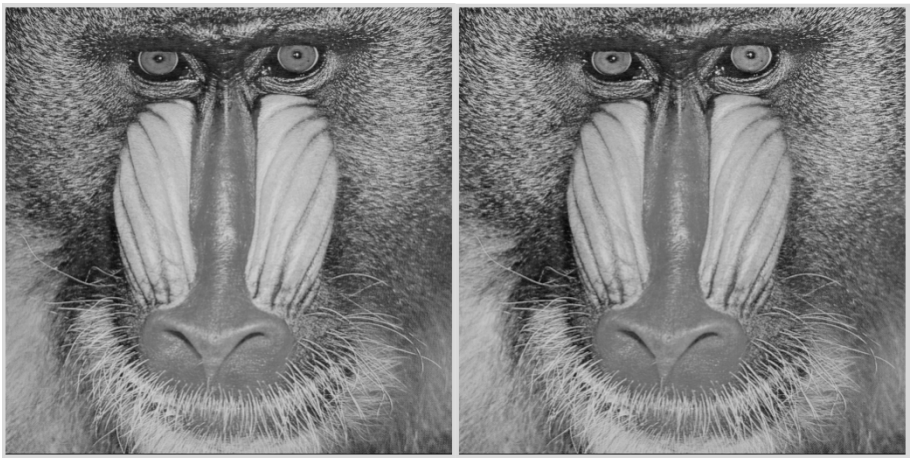


Figura 3.8: Imágenes en escala de grises. Usando Ecuación 2.1 (izquierda), esando Ecuación 2.2 (derecha)

Aparentemente los resultados son iguales, pero si hacemos una segmentación por un valor de umbral igual en una misma imagen, podemos ver como la Ecuación 2.1 genera pérdida de información (Figura 3.9), en comparación a la Ecuación 2.2, como se muestra en la Figura 3.10.

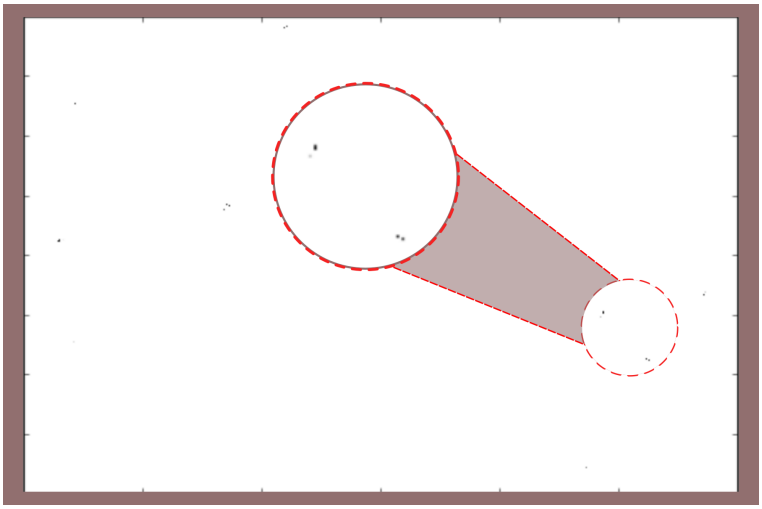


Figura 3.9: Usando Ecuación 2.2 y umbralizado con $T = 128$

Al binarizar una imagen (imagen convertida a tonos de grises usando Ecuación 2.2) en un umbral T, y haciendo un conteo de los pixeles correspondientes a un pixel con valor 1 en la Figura 3.9, tenemos un total de 1,340 pixeles, y en la Figura 3.10 un total de 3,656 pixeles, por lo tanto para pasar del espacio RGB a escala de grises usaremos la Ecuación 2.2, ya que existe una pérdida de información por más del 50 % al usar la Ecuación 2.1.

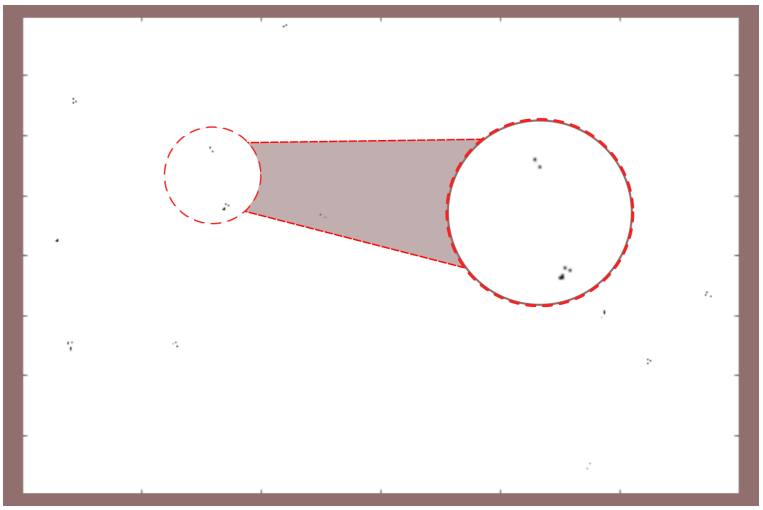


Figura 3.10: Usando Ecuación 2.1 y umbralizado con $T = 128$

3.3. Cálculo de umbral adecuado

Como se mencionó en la sección 2.1.3, una imagen binaria es creada a partir de un valor T, al contener nuestro banco de imágenes únicamente imágenes de 8 bits, el valor de $T = \{0, 1, 2, ..., 255\}$. Para encontrar el umbral adecuado para nuestra investigación se hizo un análisis por histograma de imagen.

Histograma de imagen: Un histograma de imagen es una abstracción de una imagen donde la frecuencia de cada valor de pixel en la imagen es determinada. Sea A una matriz de imagen en escala de grises donde su histograma h está dado por su función de densidad:

$$h_A(l) = |\{(i, j)|A(i, j) = l, i = 0, ..., N - 1, j = 0, ..., M - 1\}| \tag{3.4}$$

Dónde N y M son las dimensiones de A.

Para obtener un umbral adecuado se hace un barrido en todos los niveles de umbral $(0 \leq T \leq 255)$, usando el Algoritmo 7.

Algoritmo 7: Algoritmo barrido de umbrales

Entrada: A: Imagen de entrada
Salida : V: Vector de cantidades de información en cada nivel de umbral
$A_{gs} \leftarrow$ pasar a escala de grises A usando la Ecuación 2.1
para $t \leftarrow 0$ *hasta* 255 **hacer**
  $B \leftarrow$ Binarizar A_gs en nivel t
  $V \leftarrow$ agregar cantidad de 1's en B
fin
regresar V

Se encuentra de manera empírica el intervalo de umbrales adecuado, entre los valores $115 \leq T \leq 120$. Los resultados se muestran en la Figura 3.11.

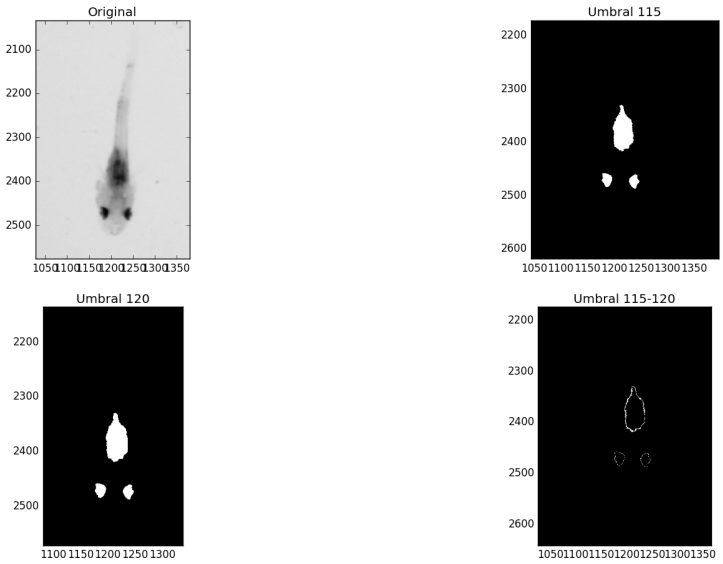


Figura 3.11: Resultado intervalo de umbrales $115 \leq T \leq 120$

3.4. Cúmulos de pixeles

La intención del procesado de las secciones anteriores es obtener una imagen binarizada B a partir de un umbral adecuardo T tal como se muestra en la Figura 3.12.

Figura 3.12: Ejemplo imagen binarizada

Lo siguiente es tratar de localizar las posiciones de los centros de los objetos detectados, para obtener un vector de posiciones V, en el cual se almacene en forma (x, y) todas las posiciones de los centros de los objetos localizados.

Usando la función de OpenCV *findContours()* extraemos los cuatro puntos que abarquen el área mínima para cada contorno continuo encontrado, asi como su centro, la longitud del segmento $\overline{ab}$ y $\overline{bc}$, como se muesta en la Figura 3.13 [38]. De esta manera podemos obtener el área de ese rectángulo, así como también la relación que existe entre $\overline{ab}$ y $\overline{bc}$.

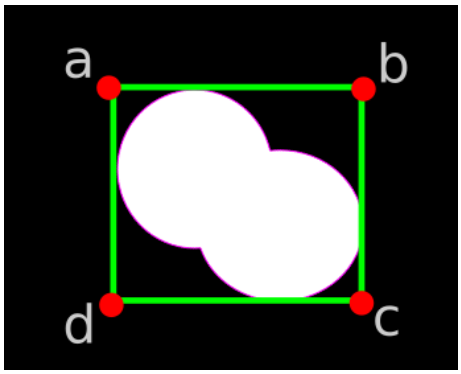


Figura 3.13: Descripción gráfica de *findContours()* , a, b, c y d son los puntos extraidos

Los resultados de la implementación de esta función son mostrados en la Figura 3.14.

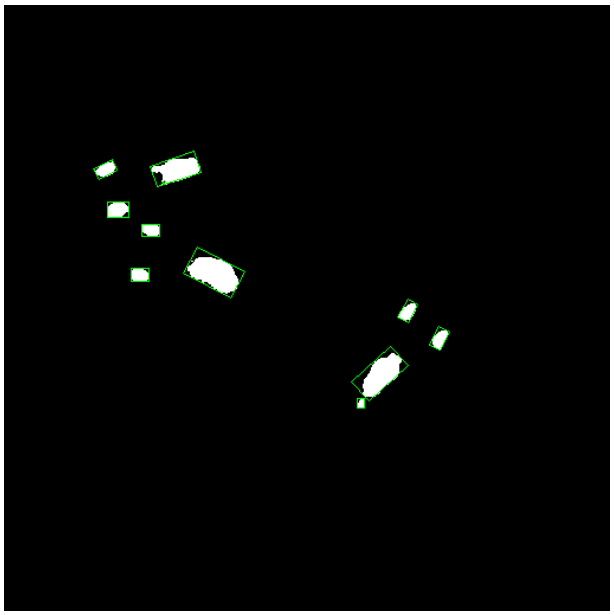


Figura 3.14: Implementación de *findContours()*

3.5. Clasificación de áreas

Una vez localizados estos centros de las áreas, es necesario obtener una clasificación adecuada para lo cual podemos destacar dos características ampliamente visibles:

- Tamaño de área
- Relación entre $\overline{ab}$ y $\overline{bc}$

Siendo así, podemos agrupar los objetos en 3 tipos:

1. Ojos
2. Aparatos digestivos
3. Ruido

Usando la función de OpenCV mencionada anteriormente se genera un archivo donde se almacenaron el área $A1$ y la relación R entre $\overline{ab}$ y $\overline{bc}$, se etiquetó cada objeto con un identificador y de manera manual se obtienen los valores de las áreas y las relaciones.

Donde:

$$A1 = \overline{ab} * \overline{bc}$$

$$R = \begin{cases} \frac{\overline{ab}}{\overline{bc}} & \text{Si } \overline{ab} \geq \overline{bc} \\ \frac{\overline{bc}}{\overline{ab}} & \text{Si } \overline{ab} < \overline{bc} \end{cases}$$

De esta manera se genera un estadístico, el cual se obtiene parcialmente automático, donde se establecen los valores mínimos y máximos de área para cada tipo de objeto, así como también la relación R, explicada previamente. En el siguiente capítulo se muestran las gráficas de dichas distribuciones y se muestran los resultados obtenidos.

3.6. Algoritmos propuestos

En esta sección explicaremos los métodos propuestos que solucionan nuestra problemática, los detalles encontrados y cómo fueron resueltos.

3.6.1. Patrón triangular

Cuando las larvas se encuentran en posición totalmente perpendicular con respecto al dispositivo de captura, como lo propone Martínez en sus tesis, podemos observar una especie de triangulación entre los ojos y el aparato digestivo de una larva. Siendo así, llamamos al ojo izquierdo el punto a, al ojo derecho punto b, y al aparato digestivo el punto c, tal como se muestra en la Figura 3.15.

Debido a que se requiere tener imágenes en situaciones controladas (únicamente larvas en posición totalmente perpendicular al dispositivo de captura), se propone un generador de patrones triangulares sintético.

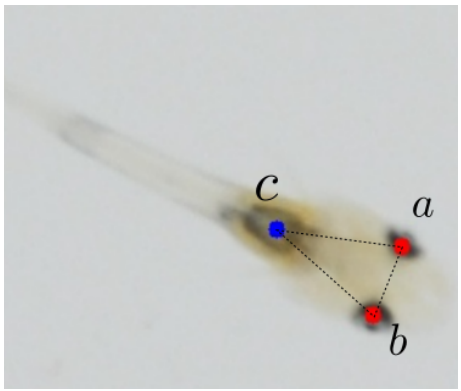


Figura 3.15: Triángulo formado por ojos y aparato digestivo

Generador Aleatorio-Sintético de Patrones Triangulares (GASPT)

Este generador sintético consiste en generar una serie de coordenadas (x, y) en un espacio limitado, en este caso tratando de replicar el espacio de imágenes de 16 megapixeles (4928 x 3264 px). Como se muestra en la Figura 3.16, necesitamos como parámetros de entrada, un punto c y un ángulo θ.

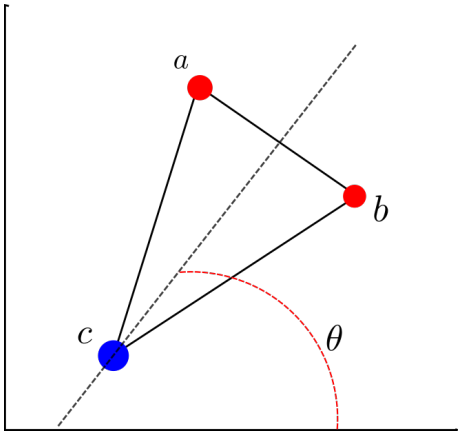


Figura 3.16: Triángulo formado por ojos y aparato digestivo (descripción gráfica)

Para explicar a detalle el generador sintético presentamos 8 casos dependiendo del valor del ángulo:

1. **Caso 1:** $0° < \theta < 90°$
   Como se muestra en la Figura 3.17, la intención es obtener los puntos a y b, teniendo como entrada el punto c y θ.

   Suponiendo que $\overline{ac} \approx \overline{bc}$ entonces:

$$\overline{h} = \sqrt{\overline{bc}^2 - \left(\frac{\overline{ab}}{2}\right)^2}$$

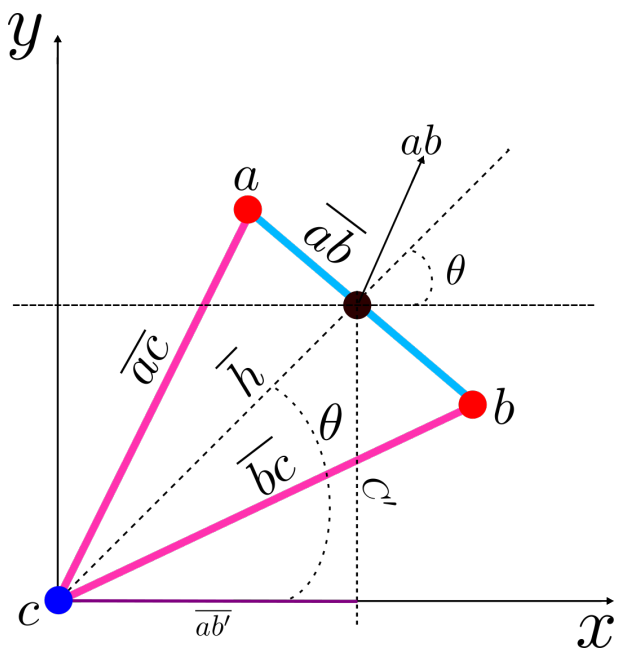


Figura 3.17: Descripción formal del patrón triangular (caso 1)

$$\sin(\theta) = \frac{C'}{\overline{h}}$$

$$C' = \sin(\theta) \cdot \overline{h}$$

$$\tan(\theta) = \frac{C'}{\overline{ab'}}$$

$$\overline{ab'} = \frac{C'}{\tan(\theta)}$$

$$x_{ab} = x_c + \overline{ab'}$$

$$y_{ab} = y_c + C'$$

Por lo tanto:

$$ab = (x_{ab}, y_{ab})$$

Conociendo el punto ab podemos obtener los puntos a y b, tal como lo muestra la Figura 3.18.

Suponemos que la distancia entre ab y a, es la misma que entre b y ab siendo igual a $\frac{\overline{ab}}{2}$:

$$\theta_b + \theta = 90°$$

$$\theta_b = 90 - \theta$$

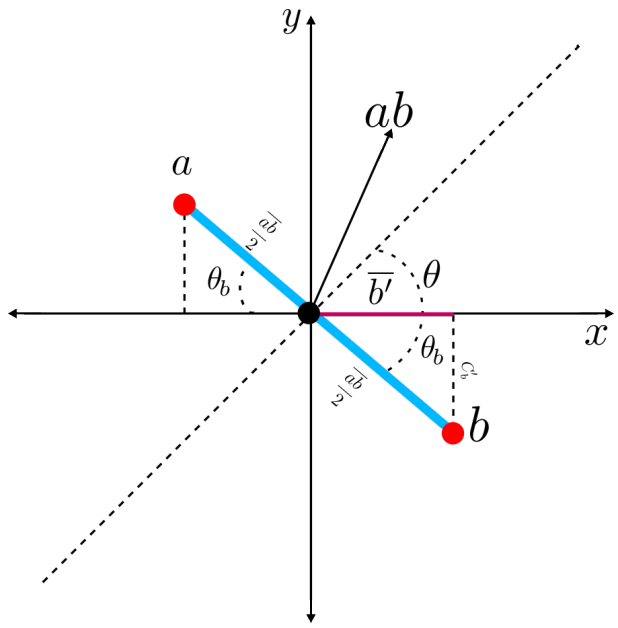


Figura 3.18: Obtención de los puntos a y b (patrón triangular)

$$\sin(\theta_b) = \frac{C'_b}{\left(\frac{\overline{ab}}{2}\right)} \quad \rightarrow \quad C'_b = \sin(\theta_b) \cdot \left(\frac{\overline{ab}}{2}\right)$$

$$\tan(\theta_b) = \frac{C'_b}{\overline{b'}} \quad \rightarrow \quad \overline{b'} = \frac{C'_b}{\tan(\theta_b)}$$

$$x_a = x_{ab} - \overline{b'} \quad \text{y} \quad y_a = y_{ab} + C'_b$$

$$x_b = x_{ab} + \overline{b'} \quad \text{y} \quad y_b = y_{ab} - C'_b$$

Por lo tanto:

$$a = (x_a, y_a) \quad \text{y} \quad b = (x_b, y_b)$$

2. **Caso 2:** $90° < \theta < 180°$
   Tal como se muestra en la Figura 3.19.

   Suponiendo que $\overline{ac} \approx \overline{bc}$ entonces:

$$\overline{h} = \sqrt{\overline{bc}^2 - \left(\frac{\overline{ab}}{2}\right)^2}$$

$$\theta' = 180 - \theta$$

$$\sin(\theta') = \frac{C'}{\overline{h}}$$

$$C' = \sin(\theta') \cdot \overline{h}$$

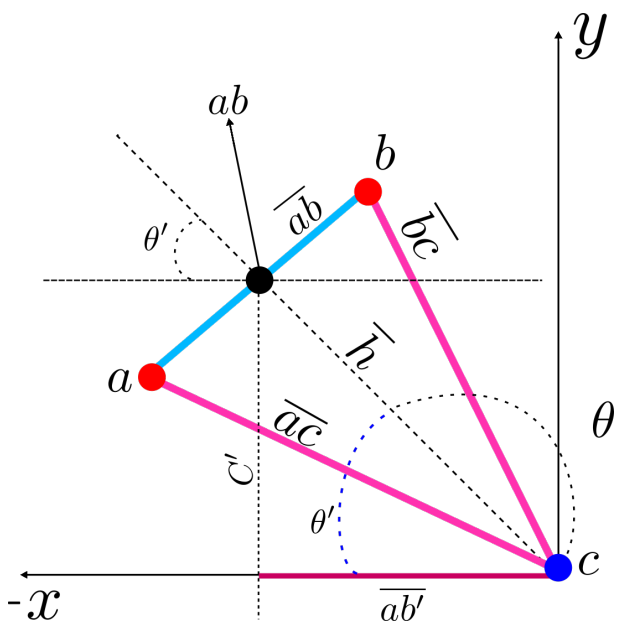


Figura 3.19: Descripción formal del patrón triangular (caso 2)

$$\tan(\theta') = \frac{C'}{\overline{ab'}}$$

$$\overline{ab'} = \frac{C'}{\tan(\theta')}$$

$$x_{ab} = x_c - \overline{ab'}$$

$$y_{ab} = y_c + C'$$

Por lo tanto:

$$ab = (x_{ab}, y_{ab})$$

Conociendo el punto ab podemos obtener los puntos a y b, tal como lo muestra la Figura 3.20.

Suponemos que la distancia entre ab y a, es la misma que entre b y ab siendo igual a $\frac{\overline{ab}}{2}$:

$$\theta'_b + \theta' = 90°$$

$$\theta'_b = 90 - \theta'$$

$$\sin(\theta'_b) = \frac{C'_b}{\left(\frac{\overline{ab}}{2}\right)} \quad \rightarrow \quad C'_b = \sin(\theta'_b) \cdot \left(\frac{\overline{ab}}{2}\right)$$

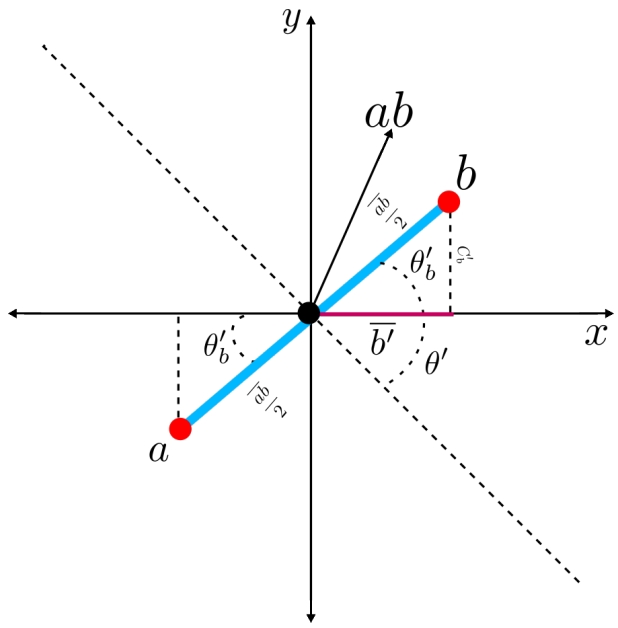


Figura 3.20: Obtención de los puntos a y b (patrón triangular)

$$\tan(\theta'_b) = \frac{C'_b}{\overline{b'}} \quad \rightarrow \quad \overline{b'} = \frac{C'_b}{\tan(\theta'_b)}$$

$$x_a = x_{ab} - \overline{b'} \quad \text{y} \quad y_a = y_{ab} - C'_b$$

$$x_b = x_{ab} + \overline{b'} \quad \text{y} \quad y_b = y_{ab} + C'_b$$

Por lo tanto:

$$a = (x_a, y_a) \quad \text{y} \quad b = (x_b, y_b)$$

3. **Caso 3:** $180° < \theta < 270°$
   Tal como se muestra en la Figura 3.21.

   Suponiendo que $\overline{ac} \approx \overline{bc}$ entonces:

$$\overline{h} = \sqrt{\overline{bc}^2 - \left(\frac{\overline{ab}}{2}\right)^2}$$

$$\theta' = \theta - 180$$

$$\sin(\theta') = \frac{C'}{\overline{h}}$$

$$C' = \sin(\theta') \cdot \overline{h}$$

$$\tan(\theta') = \frac{C'}{\overline{ab'}}$$

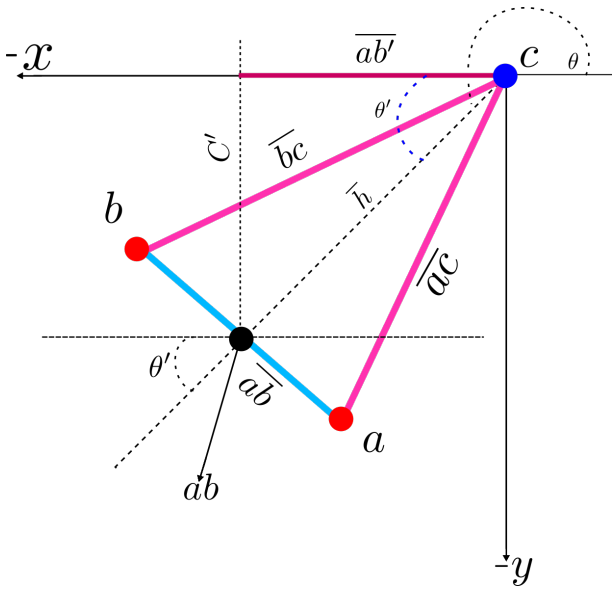


Figura 3.21: Descripción formal del patrón triangular (caso 3)

$$\overline{ab'} = \frac{C'}{\tan(\theta')}$$
$$x_{ab} = x_c - \overline{ab'}$$
$$y_{ab} = y_c - C'$$

Por lo tanto:

$$ab = (x_{ab}, y_{ab})$$

Conociendo el punto ab podemos obtener los puntos a y b, tal como lo muestra la Figura 3.22.

Suponemos que la distancia entre ab y a, es la misma que entre b y ab siendo igual a $\frac{\overline{ab}}{2}$:

$$\theta'_a + \theta' = 90°$$
$$\theta'_a = 90 - \theta'$$
$$\sin(\theta'_a) = \frac{C'_a}{\left(\frac{\overline{ab}}{2}\right)} \quad \rightarrow \quad C'_a = \sin(\theta'_a) \cdot \left(\frac{\overline{ab}}{2}\right)$$
$$\tan(\theta'_a) = \frac{C'_a}{\overline{a'}} \quad \rightarrow \quad \overline{a'} = \frac{C'_a}{\tan(\theta'_a)}$$
$$x_a = x_{ab} - \overline{a'} \quad \text{y} \quad y_a = y_{ab} + C'_a$$

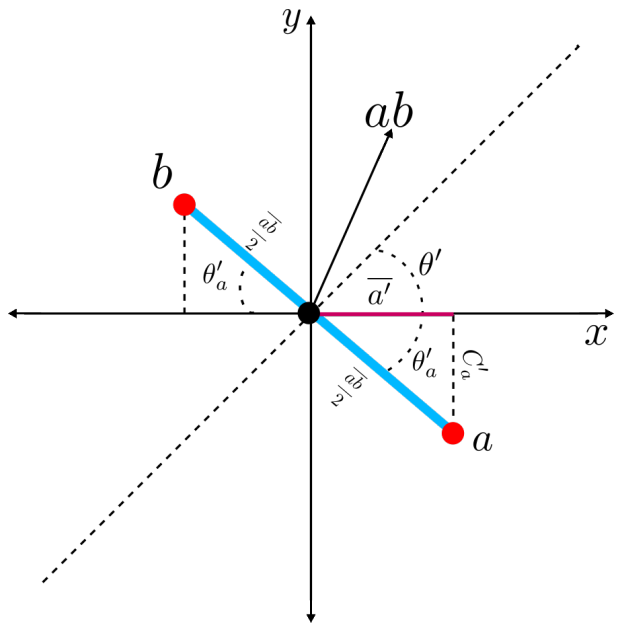


Figura 3.22: Obtención de los puntos a y b (patrón triangular)

$$x_b = x_{ab} + \overline{a'} \quad \text{y} \quad y_b = y_{ab} - C'_a$$

Por lo tanto:

$$a = (x_a, y_a) \quad \text{y} \quad b = (x_b, y_b)$$

4. **Caso 4:** $270° < \theta < 360°$
   Tal como se muestra en la Figura 3.23.
   Suponiendo que $\overline{ac} \approx \overline{bc}$ entonces:

$$\overline{h} = \sqrt{\overline{bc}^2 - \left(\frac{\overline{ab}}{2}\right)^2}$$

$$\theta' = 360 - \theta$$

$$\sin(\theta') = \frac{C'}{\overline{h}}$$

$$C' = \sin(\theta') \cdot \overline{h}$$

$$\tan(\theta') = \frac{C'}{\overline{ab'}}$$

$$\overline{ab'} = \frac{C'}{\tan(\theta')}$$

$$x_{ab} = x_c + \overline{ab'}$$

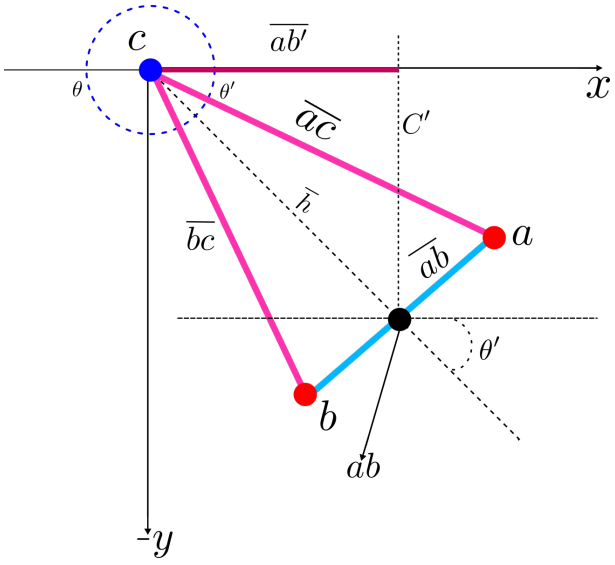


Figura 3.23: Descripción formal del patrón triangular (caso 4)

$$y_{ab} = y_c - C'$$

Por lo tanto:

$$ab = (x_{ab}, y_{ab})$$

Conociendo el punto ab podemos obtener los puntos a y b, tal como lo muestra la Figura 3.24.

Suponemos que la distancia entre ab y a, es la misma que entre b y ab siendo igual a $\frac{\overline{ab}}{2}$:

$$\theta'_a + \theta' = 90^\circ$$

$$\theta'_a = 90 - \theta'$$

$$\sin(\theta'_a) = \frac{C'_a}{\left(\frac{\overline{ab}}{2}\right)} \quad \rightarrow \quad C'_a = \sin(\theta'_a) \cdot \left(\frac{\overline{ab}}{2}\right)$$

$$\tan(\theta'_a) = \frac{C'_a}{\overline{a'}} \quad \rightarrow \quad \overline{a'} = \frac{C'_a}{\tan(\theta'_a)}$$

$$x_a = x_{ab} + \overline{a'} \quad \text{y} \quad y_a = y_{ab} + C'_a$$

$$x_b = x_{ab} - \overline{a'} \quad \text{y} \quad y_b = y_{ab} - C'_a$$

Por lo tanto:

$$a = (x_a, y_a) \quad \text{y} \quad b = (x_b, y_b)$$

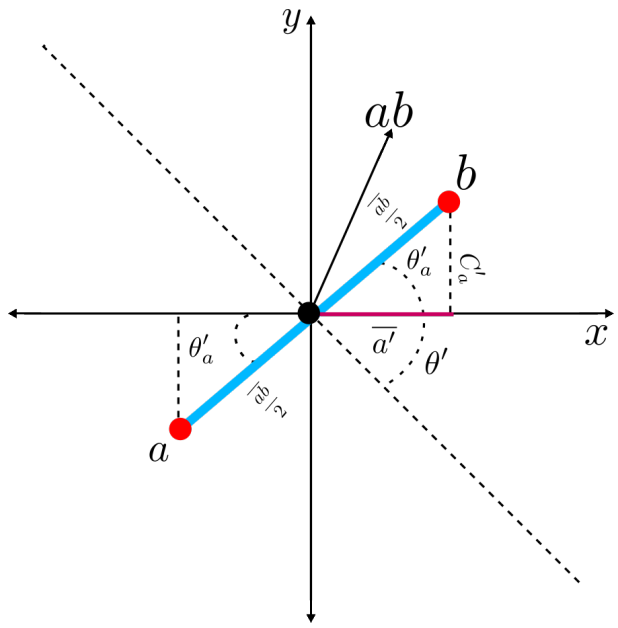


Figura 3.24: Obtención de los puntos a y b (patrón triangular)

5. **Caso 5:** $\theta = 90°$
   Tal como se muestra en la Figura 3.25.

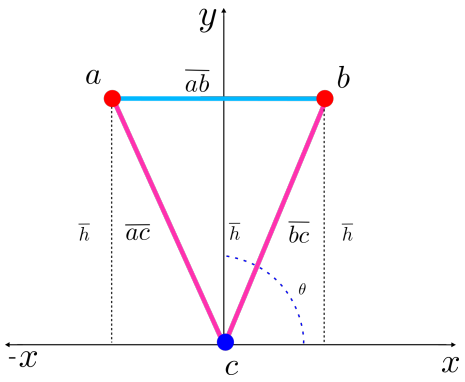


Figura 3.25: Descripción formal del patrón triangular (caso 5)

Suponiendo que $\overline{ac} \approx \overline{bc}$ entonces:

$$\overline{h} = \sqrt{\overline{bc}^2 - \left(\frac{\overline{ab}}{2}\right)^2}$$

$$x_a = x_c - \left(\frac{\overline{ab}}{2}\right) \quad \text{y} \quad y_a = y_c + \overline{h}$$

$$x_b = x_c + \left(\frac{\overline{ab}}{2}\right) \quad \text{y} \quad y_b = y_c + \overline{h}$$

Por lo tanto:

$$a = (x_a, y_a) \quad \text{y} \quad b = (x_b, y_b)$$

6. **Caso 6:** $\theta = 180°$
   Tal como se muestra en la Figura 3.26.

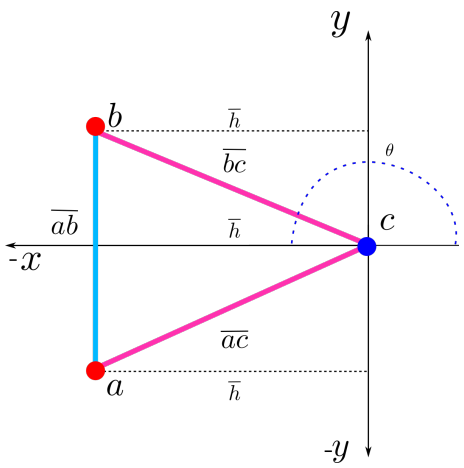


Figura 3.26: Descripción formal del patrón triangular (caso 6)

Suponiendo que $\overline{ac} \approx \overline{bc}$ entonces:

$$\overline{h} = \sqrt{\overline{bc}^2 - \left(\frac{\overline{ab}}{2}\right)^2}$$

$$x_a = x_c - \overline{h} \quad \text{y} \quad y_a = y_c - \left(\frac{\overline{ab}}{2}\right)$$

$$x_b = x_c - \overline{h} \quad \text{y} \quad y_b = y_c + \left(\frac{\overline{ab}}{2}\right)$$

Por lo tanto:

$$a = (x_a, y_a) \quad \text{y} \quad b = (x_b, y_b)$$

7. **Caso 7:** $\theta = 270°$
   Tal como se muestra en la Figura 3.27.

   Suponiendo que $\overline{ac} \approx \overline{bc}$ entonces:

$$\overline{h} = \sqrt{\overline{bc}^2 - \left(\frac{\overline{ab}}{2}\right)^2}$$

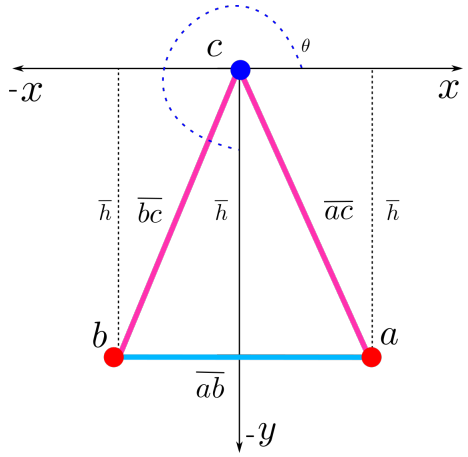


Figura 3.27: Descripción formal del patrón triangular (caso 7)

$$x_a = x_c + \left(\frac{\overline{ab}}{2}\right) \quad \text{y} \quad y_a = y_c - \overline{h}$$

$$x_b = x_c - \left(\frac{\overline{ab}}{2}\right) \quad \text{y} \quad y_b = y_c - \overline{h}$$

Por lo tanto:

$$a = (x_a, y_a) \quad \text{y} \quad b = (x_b, y_b)$$

8. **Caso 8:** $\theta = 360°$
   Tal como se muestra en la Figura 3.28.

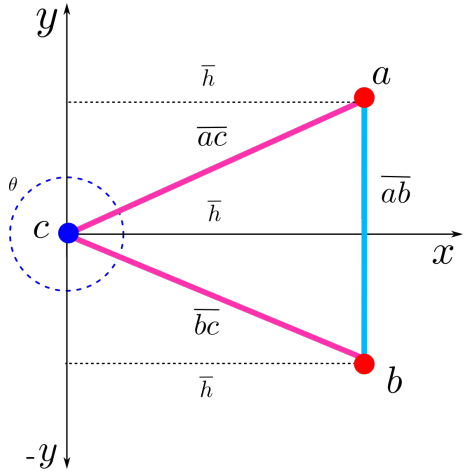


Figura 3.28: Descripción formal del patrón triangular (caso 8)

Suponiendo que $\overline{ac} \approx \overline{bc}$ entonces:

$$\overline{h} = \sqrt{\overline{bc}^2 - \left(\frac{\overline{ab}}{2}\right)^2}$$

$$x_a = x_c + \overline{h} \quad \text{y} \quad y_a = y_c + \left(\frac{\overline{ab}}{2}\right)$$

$$x_b = x_c + \overline{h} \quad \text{y} \quad y_b = y_c - \left(\frac{\overline{ab}}{2}\right)$$

Por lo tanto:

$$a = (x_a, y_a) \quad \text{y} \quad b = (x_b, y_b)$$

El Algoritmo 8 describe el proceso del GASPT. Este proceso puede hacer iterativamente que aparezca el patrón las veces que deseemos en un espacio determinado.

Algoritmo 8: Descripción general del GASPT

Entrada: $It \rightarrow$ un valor entero que corresponde a la cantidad de patrones a crear
Entrada: $n \rightarrow$ un valor entero que indique la cantidad de renglones en el espacio a crear
Entrada: $m \rightarrow$ un valor entero que indique la cantidad de columnas en el espacio a crear
Entrada: $T1 \rightarrow$ un arreglo con las medidas reales obtenidas de manera manual de las distancias de $\overline{ac}$ ó $\overline{bc}$
Salida : $PS \rightarrow$ una matriz donde cada renglón corresponde a un patrón sintético (a, b, c, θ)

$PS \leftarrow \emptyset$
$PS' \leftarrow \emptyset$
`//`PS' `Para almacenar cada renglón de la matriz`

para $i \leftarrow 0$ *hasta* $(It - 1)$ **hacer**
    $PS' \leftarrow calcularPuntos(T1, n, m)$
    $PS(i, 0) \leftarrow PS'(0)$
    $PS(i, 1) \leftarrow PS'(1)$
    $PS(i, 2) \leftarrow PS'(2)$
    $PS(i, 3) \leftarrow PS'(3)$
regresar $[PS]$

A su vez la función $calcularPuntos(Medidas, filas, columnas)$ arroja como resultado $[a, b, c, \theta]$, usando como condición el valor del ángulo θ y usando el caso que le corresponde de los 8 casos explicados anteriormente. El punto c es generado aleatoriamente, de tal manera

que x_c es igual a un valor aletorio entre 0 y m; y_c es un valor aletorio entre 0 y n, y por último θ es un valor aletorio entre 1 y 360. Siendo los resultados para 50 patrones sintéticos los mostrados en la Figura 3.29 (los puntos azules corresponden a aparatos digestivos c y los puntos rojos a ojos a, b).

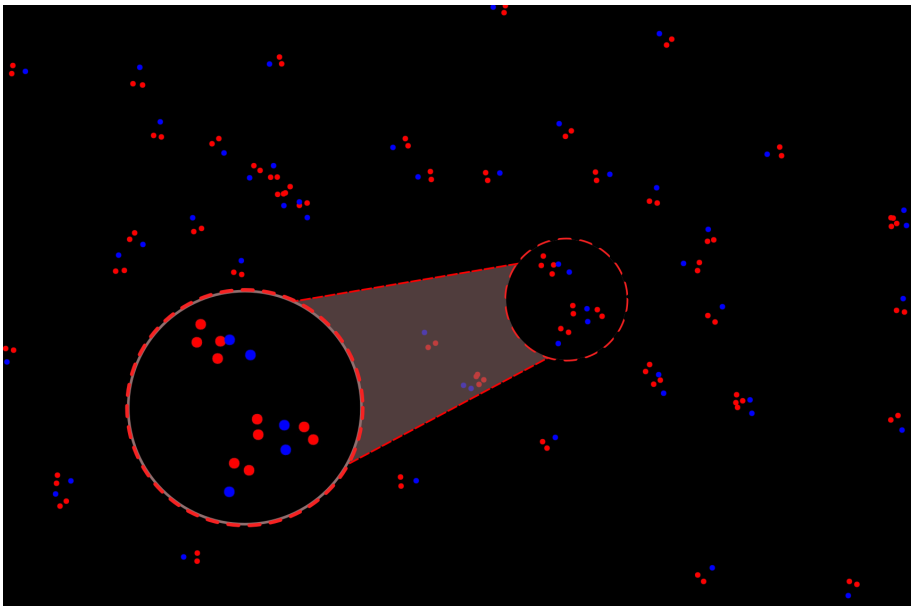


Figura 3.29: 50 patrones sintéticos creados con el generador propuesto (implementación en C++)

Cabe mencionar que $T1$ es un arreglo con las medidas reales en pixeles contabilizadas manualmente de una imagen de larvas, y para obtener las de medidas de $\overline{ac}$ ó $\overline{bc}$ se genera un número aletorio que indique el índice en $T1$ y se toma una medida real.

3.6.2. Usando un método exacto

Una vez que se tiene un conjunto de puntos creados de manera sintética los cuales representan It patrones triangulares, se requiere poder identificar de manera automática, cuáles 3 puntos corresponden a la tercia de un patrón triangular.

Para dar solución a este problema se propone el siguiente procedimiento:
clasificar en dos vectores de coordenadas.

- L_1 : Este vector contendrá todas las coordenadas de la matriz SP que corresponden a un punto c.
- L_2 : Este vector guardará todas las coordenadas de la matriz SP que corresponden a los puntos a y b.

Como se mencionó previamente ya tenemos caracterizada una clasificación entre puntos a, b y puntos c, por esta razón se reordenan los puntos creados sintéticamente de esta manera.

Al llegar a este punto, nos dimos cuenta de dos cosas: primero podemos ver nuestro espacio de exploración (imagen sintética) como una nube de puntos en dos dimensiones; segundo, también podemos abordarlo como si fuese un grafo y las conexiones entre los puntos que generan un patrón triangular, las rutas a seguir.

a) **Nube de puntos:**
Representación en un plano cartesiano a un conjunto de pares ordenados (x_i, y_i).

b) **Grafo:**
Conjunto de puntos (vértices o nodos), que son unidos por líneas (aristas).

Tal como se muestra en la Figura 3.30.

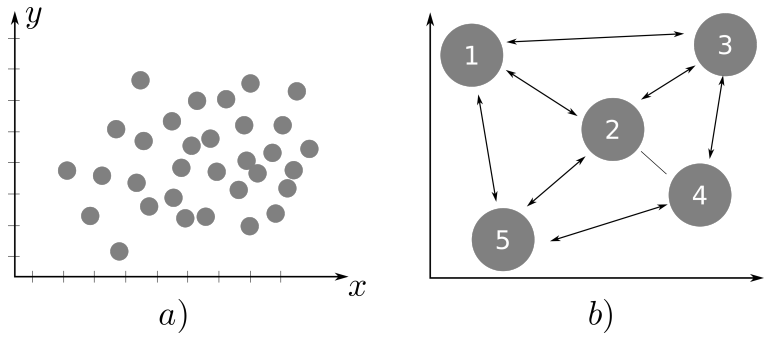


Figura 3.30: Representacón gráfica de una nube de puntos y un grafo

Para poder localizar los pares de puntos más cercanos en un espacio k-dimensional se puede utilizar la teoría del problema: *El par más cercano* y en cual podemos recalcar el Teorema 3.6.1 mencionado en el libro de Franco P. Preparata et al. [39].

Teorema 3.6.1 *En el modelado de árboles para la solución de problemas, cualquier algoritmo que solucione los problemas,* El par más cercano*,* Todos los vecinos más cercanos*,* Triangulación *y* EMST *(Euclidian Minimum Spanning Tree) necesita* $\Omega(N \log N)$ *operaciones.*

Nuestro problema puede ser representado por cualquiera de los problemas mencionados en el Teorema 3.6.1 para lo cual explicaremos cada uno de ellos de manera general.

El par más cercano

El par más cercano (por sus traducción al inglés CLOSEST PAIR), describe que: dado N puntos en un plano encontrar aquellos dos, que su distancia mutua es la más pequeña [39].

Todos los vecinos más cercanos

El problema de *Todos los vecinos más cercanos* (por su traducción al inglés ALL NEAREST NEIGHBORS), describe que: Dados N puntos en un plano encontrar el vecino más cercano de cada uno. *El vecino más cercano* es una relación en un conjunto S de puntos como se muestra: el punto b es el vecino más cercano al punto a, denotado por $a \rightarrow b$, si [39]:

$$dist(a, b) = mindist(a, c)$$

$$c \in S - a$$

EMST

Este problema dice que: Dados N puntos en un plano, construir un árbol con la mínima distancia total, cuyos vértices son los puntos dados. La solución a este problema significará la lista de $N - 1$ aristas de las comparaciones de pares de puntos de los bordes del árbol. La aplicación de este problema es comúnmente usado en redes de servicio (telefonía, agua potable, gas, etc.) [39].

Triangulación

El problema de *Triangulación* dice que: Dados N puntos en un plano, juntándolos sin intersectar segmentos de línea recta, de tal manera que cada región interna al casco convexo sea un triángulo (Figura 3.31). Siendo un gráfico plano, una triangulación de N vértices tiene por lo menos $3N - 6$ bordes. Una solución a este problema debe dar al menos una lista de estos bordes [39].

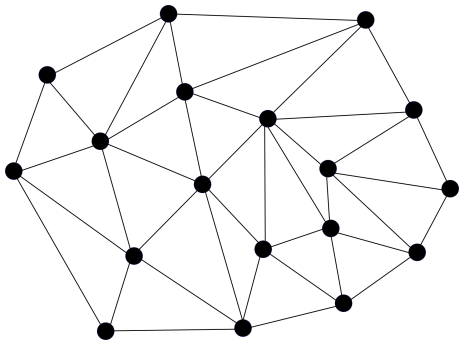


Figura 3.31: Triangulación de un conjunto de puntos

Primera aproximación

Para ser capaces de resolver nuestra problemática, se inició con una con un conjunto de $N = 50$ patrones, esto da como resultado que el tamaño del conjunto $L_2 = N$ y del conjunto $L_2 = 2N$, para tener un total de $3N$ puntos en un conjunto correspondiente a una imágen

de larvas de peces. Como se mencionó, tenemos la ventaja de tener clasificados los puntos en 2 grupos de tal manera que necesitamos comparar L_1 contra L_2, es decir necesitamos obtener los dos puntos de L_2 que correspondan a un punto L_1 y que cumpla con las siguientes condiciones encontradas:

- Como se puede observar en la Figura 3.29, generalmente los puntos $L_2(i)$ y $L_2(j)$ que corresponden a $L_1(k)$ son los más cercanos
- Se establecen dos tipos de relaciones R_1 y R_2 las cuales corresponden a:

$$R_1 = \frac{\overline{ac}}{\overline{ab}}$$

$$R_2 = \frac{\overline{bc}}{\overline{ab}}$$

Lo primero que se propone es obtener los dos puntos de L_2 más ceranos a cada punto de L_1. Sea $L_2(i)$ y $L_2(j)$ los dos puntos más cercanos a $L_1(k)$ tal como se muestra en la Figura 3.32

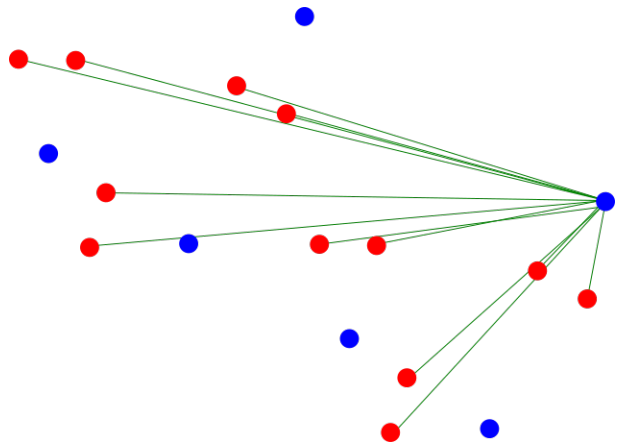


Figura 3.32: La distancia de los puntos L_2 a un punto L_1

Para solucionar este problema proponemos el siguiente procedimiento:

1. Crear dos aristas candidatas a solución en forma $Ar_1(L_1(i), L_2(j))$ y $Ar_2(L_1(i), L_2(k))$ si su distancia euclidiana es la más corta
2. Crear una tercera arista $Ar_3(L_2(j), L_2(k))$, almacenar las 3 aristas y borrarlas de las listas L_1, L_2 si se cumplen las siguientes condiciones:
   - $R_{min} \leq \frac{length(Ar_2)}{length(Ar_3)} \leq R_{max}$
   - $R_{min} \leq \frac{length(Ar_1)}{length(Ar_3)} \leq R_{max}$

Dónde R_{min} es el tamaño mínimo encontrado en la relación del patrón triangular, y R_{max} el valor máximo ($1.6 \leq R \leq 2.1$) y podemos verlo en la Figura 3.34.

3. Repetir desde el paso 1 hasta que L_1 y/o L_2 no contengan elementos.

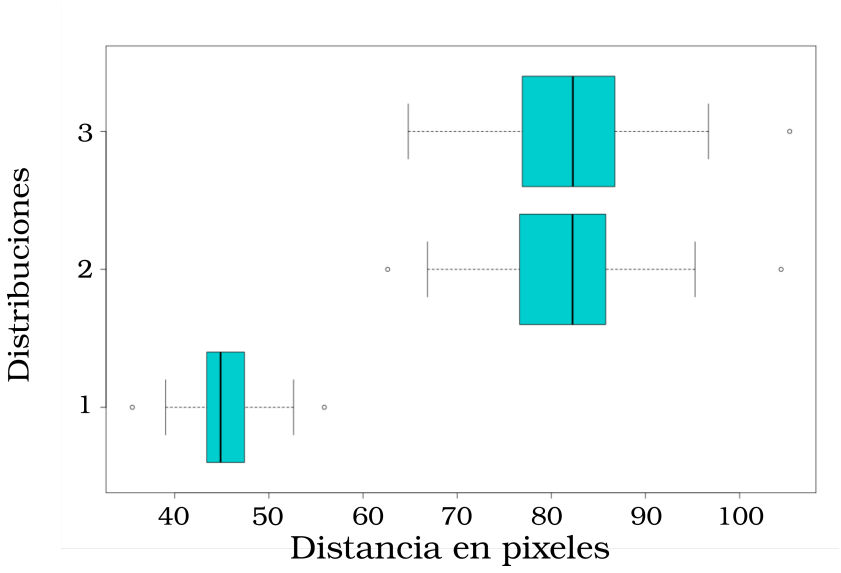


Figura 3.33: Gráfica de bigotes de medidas reales: (1)→ $\overline{ab}$, (2)→ $\overline{ac}$, (3)→ $\overline{bc}$

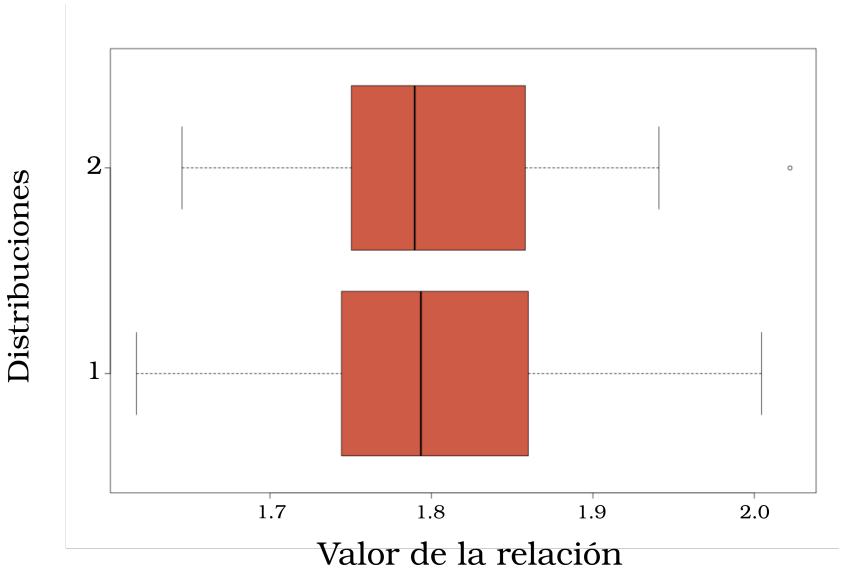


Figura 3.34: Gráfica de bigotes de medidas reales: (1)→ R_1, (2)→ R_1

3.6.3. Usando una metaheurística

Para esta parte del desarrollo de esta investigación se hace el uso del concepto de modelo de islas y el uso de un GA, con la finalidad de poder reconocer el patrón triagular cuando se tiene una gran cantidad de patrones a reconcer. También se hace con la finalidad de ser capaces de determinar los alcances del algoritmo verificando cuando cae su certeza y eficiencia.

Como se mencionó en la Sección 2.2.6, la intención es poder hacer un proceso evolutivo tradicional sobre subpoblaciones y que exista un intercambio de individuos a fin de que al final tengamos una población con la mayor cantidad de individuos adecuados. Para esto se explicarán los detalles de esta implementación.

Codificación

En esta parte explicaremos cómo codificamos a un indivuo y se retoma lo mencionado en la Sección 3.6.2 los dos vectores de posiciones (L_1, L_2). Utilizamos una codificación binaria la cual corresponde a una concatenación de un índice de L_1 y dos de L_2. Tal como se muestra en la Figura 3.35.

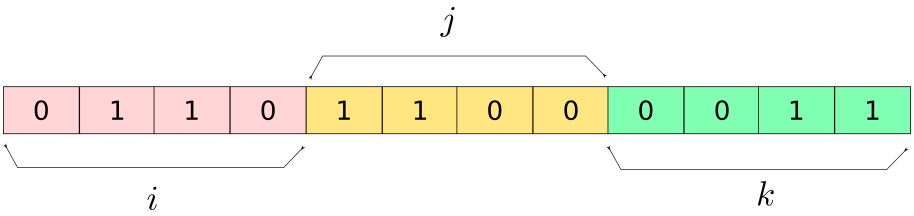


Figura 3.35: Representación binaria de cada indivuo

De tal manera que i represtará el índice de alguna posición en el vector L_1; j y k los índices de alguna posición de L_2 de tal manera que: $j \neq k$. Y así convertimos a código binario esos números enteros i, j, k y los concatenamos para formar un individuo.

Operadores Evolutivos

Para generar el proceso evolutivo tomamos los parámetro utilizados por Ayala et al. [7], los cuales se describen a continuación. Para el proceso de selección usamos el método de selección por ruleta, el cual está basado en el fitness del individuo, de tal manera que un individuo con mayor fitness tiene mayor probabilidad de ser seleccionado. Para el proceso de cruzamiento se utiliza el de 1 punto (1-point crossover), el cual consiste en cruzar a los padres para generar nuevos hijos desde 1 punto de la cadena. Para mutación utilizamos el método bitwise, que consiste en evaluar la probabilidad de mutación para cambiar bit a bit la cadena. Los parámetros se muestran en la Tabla 3.1

Población

Como se menciona en la Tabla 3.1, el tamaño de la población va a depender de la longitud del vector L_1, ya que partimos de la primicia de que debe haber tantos patrones como

Tabla 3.1: Parámetros para el algoritmo evolutivo

Parámetro	Valor
Tamaño de la población	El tamaño de L_1
Probabilidad de cruzamiento	0.55
Probabilidad de mutación	0.10
Método de selección	Ruleta
Método de cruzamiento	1 punto

aparatos digestivos identificados. Luego usando la teoría del modelo de islas, utilizamos la comunicación tipo estrella, en la cual creamos 3 subpoblaciones: $Pop1$, representa la subpoblación maestra y $Pop2, Pop3$ representan las esclavas. Antes de someter las poblaciones al proceso evolutivo éstas se fijan de la siguiente manera:

1) $Pop1$: Es la población vacía que va almacenando todos aquellos individuos con el fitness más alto.

2) $Pop2$: Tomado de la Sección 3.6.2, creamos esta población con los $L_1(i)$ con menor distancia a $L_2(j)$ y $L_2(k)$.

3) $Pop3$: Creamos esta población con los $L_1(i)$ con tercer y cuarta menor distancia a $L_2(j)$ y $L_2(k)$.

De esta manera cada cierto número de generaciones los individuos con mayor fitness son llamados de $Pop2$ y $Pop3$ a $Pop1$, y el resto se sigue evolucionando.

Función de *fitness*

Para evaluar el *fitness* de cada individuo se propone lo siguiente: como se menciona en la Sección 3.6.1, triángulo formado por par de ojos (puntos a y b) y aparto digestivo (punto c), tiene ciertas características; primero, existen dos relaciones de proporción R_1 y R_2 surgida de la división entre $\overline{ac}$ y $\overline{ab}$, y $\overline{bc}$ y $\overline{ab}$, respectivamente. También podemos destacar que la distancia entre ojos según el estadístco mostrado en la Figura 3.33 tiene el siguiente intervalo $(35.5 \leq \overline{ab} \leq 55.9)$, de tal manera que de aquí podemos establecer una función de *fitness*.

Si se cumple que: $(1.6 \leq R_1 \leq 2.1)$ y $(1.6 \leq R_2 \leq 2.1)$, normalizar los valores de $\overline{ac}$ y $\overline{ac}$ y evaluar:

$$F(I) = |1 - (\overline{ac} - \overline{bc})| \tag{3.5}$$

Dónde F representa la función de *fitness* e I cada individuo.

Si no se cumple con los valores de R_1 y R_2 se plantea la siguiente ecuación:

$$F(I) = |1 - (\overline{ac} - \overline{bc})| \cdot p1 \tag{3.6}$$

Dónde $p1$ es un factor de penalización por no cumplir con el patrón. Así mismo se penaliza usando un factor $p2$ si $(35.5 \leq \overline{ab} \leq 55.9)$ no se cumple, de tal manera que $F(I) = F(I) * p2$. Ambos valores $p1, p2$ son decimales con la finalidad de reducir el *fitness* aún más.

Esta propuesta usando un método evolutivo, se plantea escencialmente para el caso sintético, ya que en las imágenes reales con las que se cuenta, no hay más de 60 especímenes en ellas. Aunque el patrón encontrado se tiene bien caracterizado, existen situaciones donde no se presentan las larvas en posiciones ideales, para esto se identifican los casos especiales encontrados.

3.6.4. Casos especiales

Durante la parte de implementación, nos pudimos dar cuenta de la problemática que menciona Martínez en su trabajo de investigación: la posición de las larvas es crucial para generar una buena clasificación y un conteo certero. Para ello hemos llamado a una larva que no está perfectamente enfocada y en posición totalmente perpendicular al dispositivo de captura como un *Caso Especial* (CE).

Los CE, parecieran particularmente obvios, ya que se está trabajando con larvas de peces vivas las cuales se encuentran en constante movimiento. Debido a esto nos hemos dado a la tarea de clasificar los CE en 5 grupos.

1) CE en posición diferente a perfectamente perpendicular al dispositivo de captura. Mostrado en la Figura 3.36.

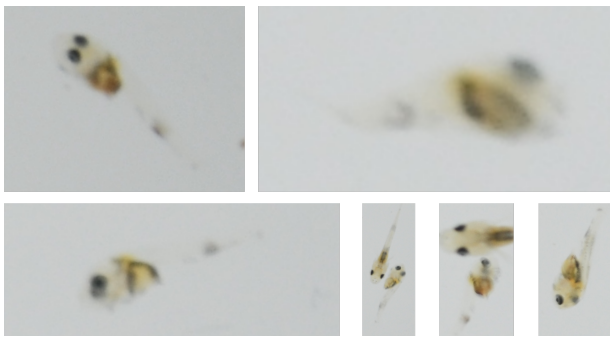


Figura 3.36: Larvas en posiciones no ideales

2) CE que presenta traslapes. Mostrado en la Figura 3.37.

3) CE que salió del espacio de captura. Mostrado en la Figura 3.38.

4) CE ruido que no corresponde a una parte de una larva y es detectado como tal. Mostrado en la Figura 3.39.

5) CE en el cual apreciamos ojos mas pequeños o pigmentación diferente en el aparato digestivo. Mostrado en la Figura 3.40.

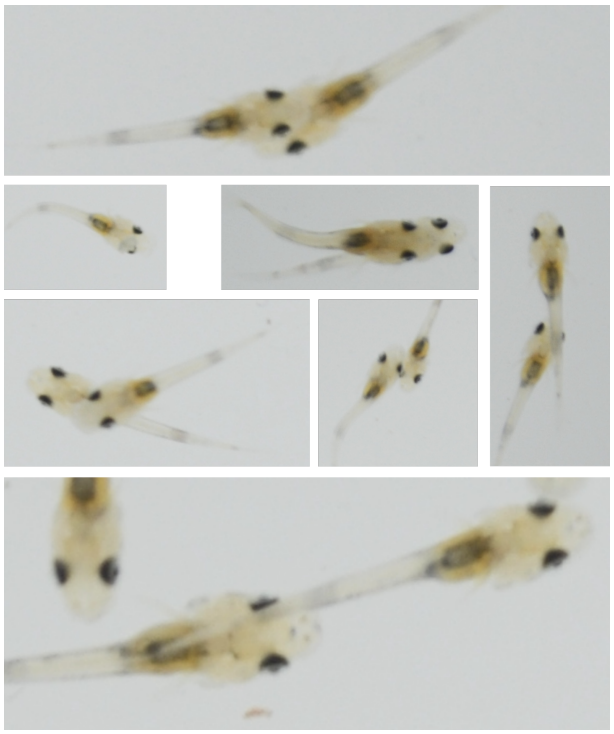


Figura 3.37: Larvas traslapadas

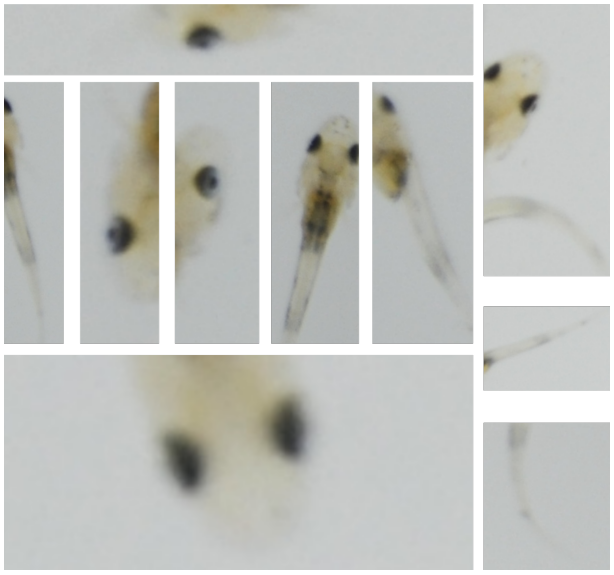


Figura 3.38: Larvas fuera del espacio de captura

Una vez que se tienen identificados todos los casos donde se cumple el patrón triangular, es decir, que no se trata de un CE podemos generar las siguientes soluciones para identificar y clasificar los CE de manera adecuada.

Figura 3.39: Ruido detectado como parte de una larva

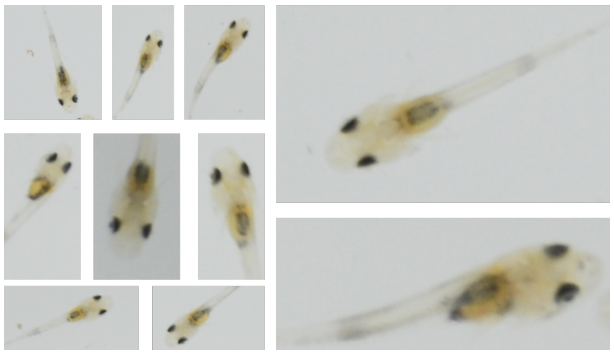


Figura 3.40: Larvas con pigmentación diferente y tamaño de ojos fuera de la clasificación

1) Como parte del proceso de reconocimiento del patrón triangular, es el borrado de los puntos que ya fueron asignados a un patrón reconocido, se establece que si dos puntos de L_2 están con una distancia menor a la del estadístico obtenido con respecto a un punto de L_1 este también es reconocido como un patrón, pero se etiqueta como un CE (Figura 3.36).

2) Una vez solucionado el caso anterior también se borran los objetos clasificados como CE de tipo I; luego establecemos que si un punto de L_1 está a una distancia menor que el estadístico de un punto de L_2 y el segundo punto no lo está, el punto de L_1 es almancenado el punto L_2 duplicado y es etiquetado como un CE, esto con la finalidad de detectar larvas con un solo ojo visible. También si dos puntos de L_2 están a menor distancia de lo que el estadístico nos dice, y no se encuentra ningún punto de L_1 con menor distancia al estadístico, también se clasifica como CE tipo II (Figura 3.37).

3) Este también se soluciona de la misma manera que el anterior (Figura 3.38).

4) Si la distancia del punto clasificado a cualquier otro punto cercano a él es mayor a la del estadístico, no es contemplado y es borrado (Figura 3.39).

5) Resolviendo lo anterior, este caso se resuelve también (Figura 3.40).

3.7. Método integrado

Con la intención de mostrar todo el proceso propuesto integrado, se muestra el procedimiento completo en el Algoritmo 9

Algoritmo 9: Algoritmo de todo el proceso integrado

Entrada: $A \rightarrow$ imagen RGB de larvas de peces
Entrada: $t \rightarrow$ nivel de umbral adecuado
Salida : $A_c \rightarrow$ imagen RGB con larvas identificadas
Salida : $F_c \rightarrow$ archivo de texto con posiciones de los patrones encontrados
$A_{gs} \leftarrow$ Pasar a escala de grises A usando Ecuación 2.2
$B \leftarrow$ binarizar A_{gs} en nivel t
$C \leftarrow$ `findContours(`B`)` de OpenCV
$L_1 \leftarrow$ clasificar áreas de C como aparatos digestivos
$L_2 \leftarrow$ clasificar áreas de C como ojos
$Pat \leftarrow \emptyset$
si $L_1 < 100$ **luego**
  $Pat \leftarrow$ usar método exacto L_1, L_2
fin
sino
  $Pat \leftarrow$ usar metaheurística L_1, L_2
fin
$F_c \leftarrow$ crear un archivo CSV con Pat
$A_c \leftarrow$ crear una copia de A
$A_c \leftarrow$ dibujar los patrones Pat

La función *findContours* de OpenCV, que se explicó en la Sección 3.4, tiene como parámetro de entrada una imagen binarizada de la cual se extrae las posiciones de todos los contornos contenidos en un área definida por puros 1's, y así mismo podemos extraer el rectángulo que puede contener dicha área; de este rectángulo se puede extraer la posición del centro, su área y su relación $\frac{base}{altura}$, estás últimas dos, nos sirven para clasificar según una validación estadística.

Se almacenan en el vector L_1 todas las posiciones que cumplan con la clasificación de un aparato digestivo. De la misma manera se almancenan en L_2 las posiciones que cumplan con las características de ojos. Suponemos que habrá tantas larvas como aparatos digestivos clasificados; partiendo de los resultados obtenidos con el generador sintético si existen menos de 100 especímenes en la imagen a procesar se utilizará el método exacto propuesto. De lo contrario si se clasifican 100 o más especímenes se utilizará el GA por modelo de islas también propuesto para resolver los patrones encontrados. Se crea una imagen con los patrones encontrados y un archivo de texto con la información de las posiciones de dichos patrones.

Como podemos darnos cuenta, al integrar todo lo visto en este capítulo podemos plasmar un algoritmo simple pero con resultados buenos, los cuales se explicarán a detalle en el siguiente capítulo.

Capítulo 4

Pruebas y resultados

En este capítulo se muestran todos los resultados detallados de los diseños explicados en el Capítulo 3. Las implementaciones se hicieron principalmente en C++, las pruebas de concepto fueron realizadas en python y octave. La librería para el procesamiento de imágenes utilizada fue OpenCV. Los equipos para realizar las implementaciones fueron dos, los cuales tienen las siguientes características:

- Laptop personal (Intel Core^{TM} i5 CPU M 540 @ 2.53GHz 4, 4 Gb RAM)
- Workstation (Intel(R) Xeon(R) CPU E5-1620 v3 @ 3.50GHz x8, 16 Gb RAM)

Para mostrar los resultados obtenidos usáremos como ejemplo 4 imágenes de muestra, las Figuras 4.1 y 4.2 muestran estos ejemplos.

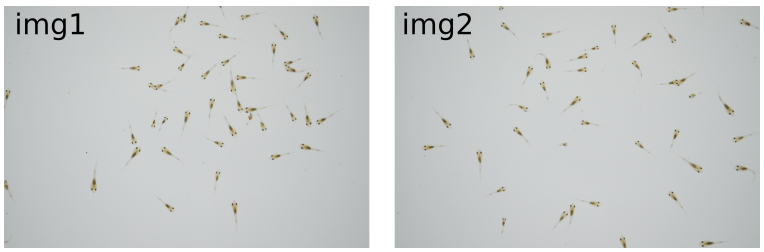


Figura 4.1: Imágenes en RGB (experimento realizado por Martínez en Junio de 2015 [1])

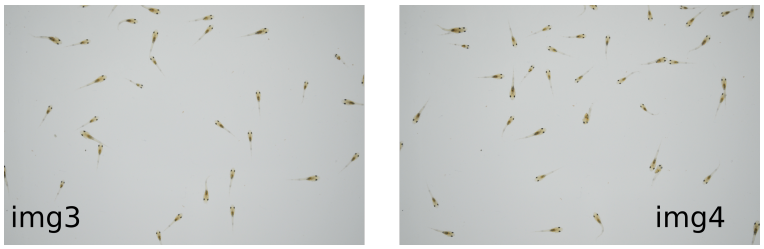


Figura 4.2: Imágenes en RGB (experimento realizado por Martínez en Junio de 2015 [1])

4.1. Escala de grises

Como se mencionó en el Capítulo 2, la elección de pasar del espacio de color RGB a escala de grises, fue realizada con la Ecuación 2.2, y los resultados se muestran en las Figuras 4.3 y 4.4

Figura 4.3: Pase a escala de grises usando la Ecuación 2.2: img1 (izquierda), img2 (derecha)

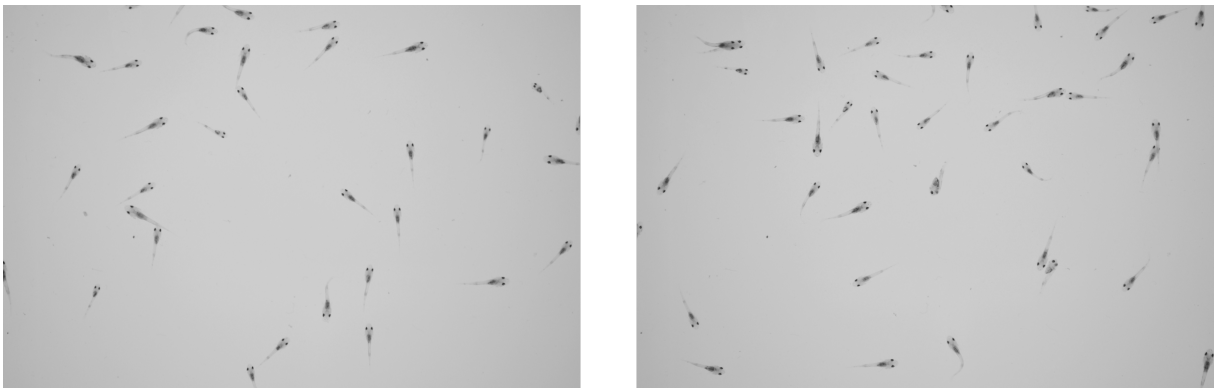


Figura 4.4: Pase a escala de grises usando la Ecuación 2.2: img3 (izquierda), img4 (derecha)

4.2. Umbral óptimo

En el Capítulo 3, se menciona el uso de un algoritmo para calcular umbrales adecuados, para lo cual nosotros usamos las 4 imágenes que se plasman en las Figuras 4.1 y 4.2. Los resultados graficados se muestran en las Figuras 4.5, 4.6 y 4.7.

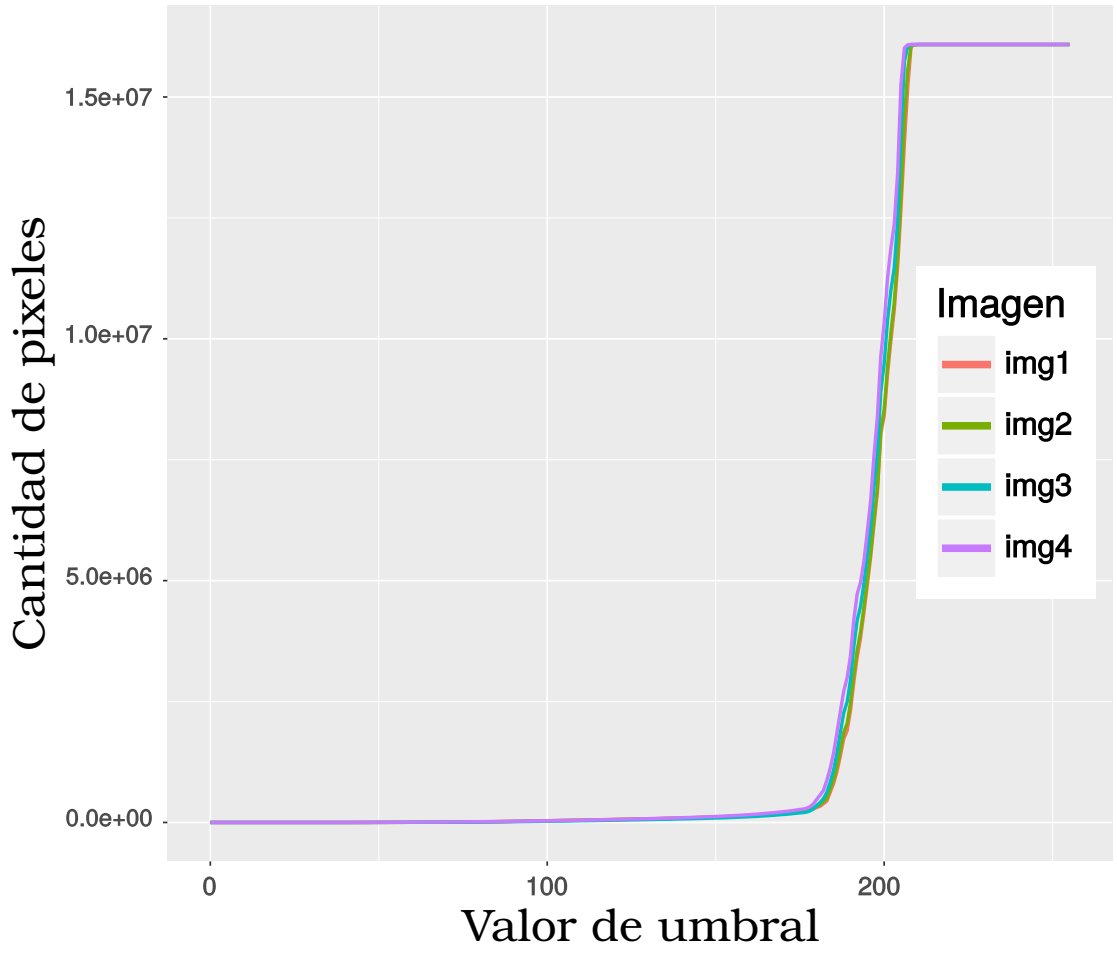


Figura 4.5: Gráfica obtenida del Algoritmo 7 para contabililzar la cantidad de pixeles obtenidos en cada nivel de umbral por imagen

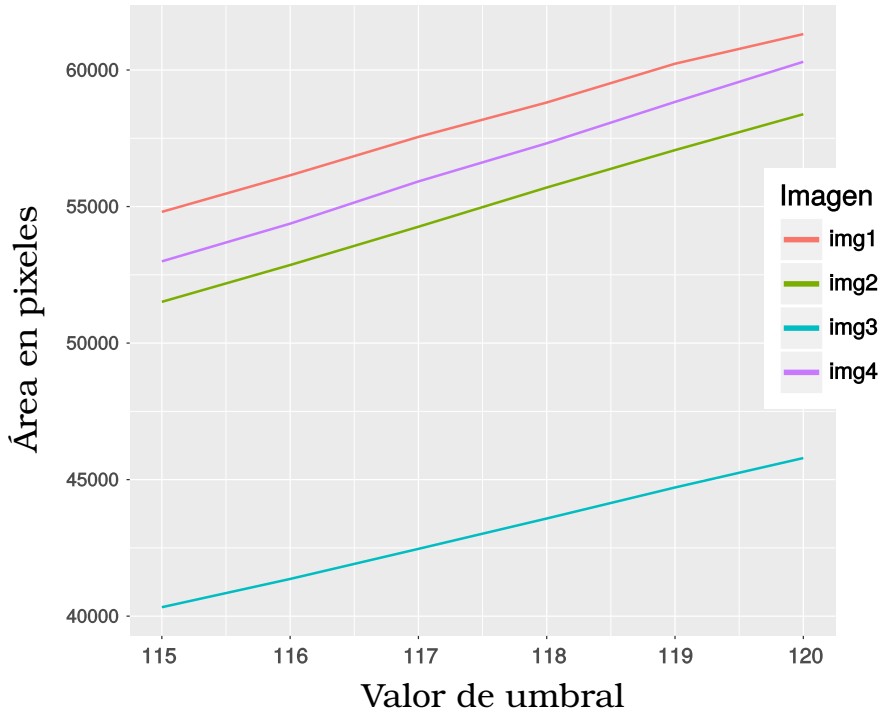


Figura 4.6: Acercamiento en el intervalo $115 \leq T \leq 120$

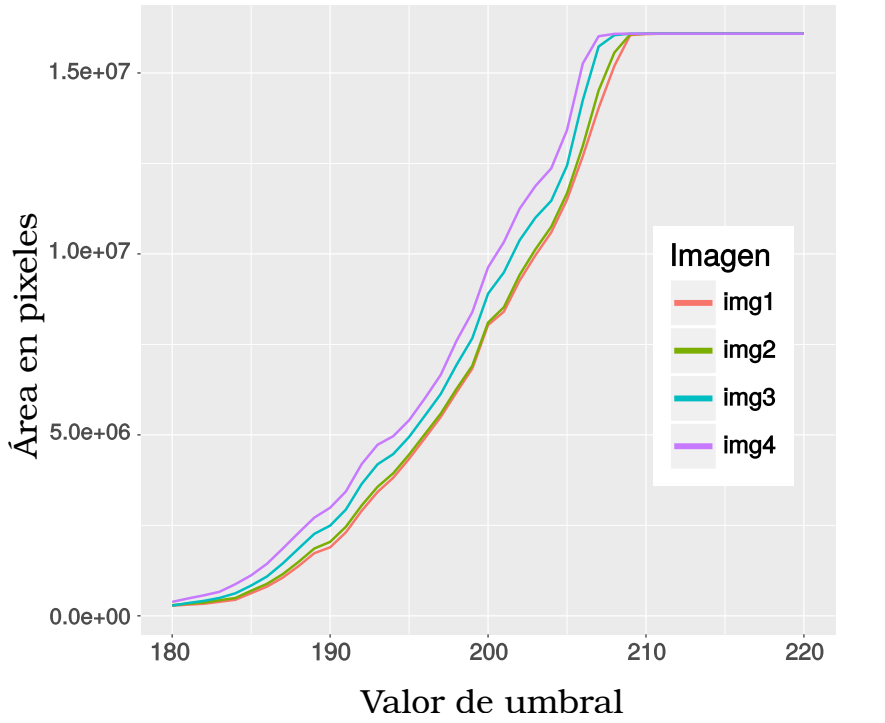


Figura 4.7: Acercamiento en el intervalo $180 \leq T \leq 220$

4.3. Cúmulos de pixeles

Dando una imagen binarizada en el umbral (o intervalo de umbrales) encontrado, a la función *findContours()* de OpenCV, podemos ser capaces de obtener los cuatro puntos (a, b, c, d) que formen el rectángulo mínimo que abarque cada una de las áreas en la imagen binarizada. De esto podemos extraer el centro C, la base y la altura $(\overline{ab}, \overline{bc})$, le llamaremos base al valor mayor de ambos, y altura al valor mínimo. Esto podemos verlo en las Figuras 4.8 y 4.9.

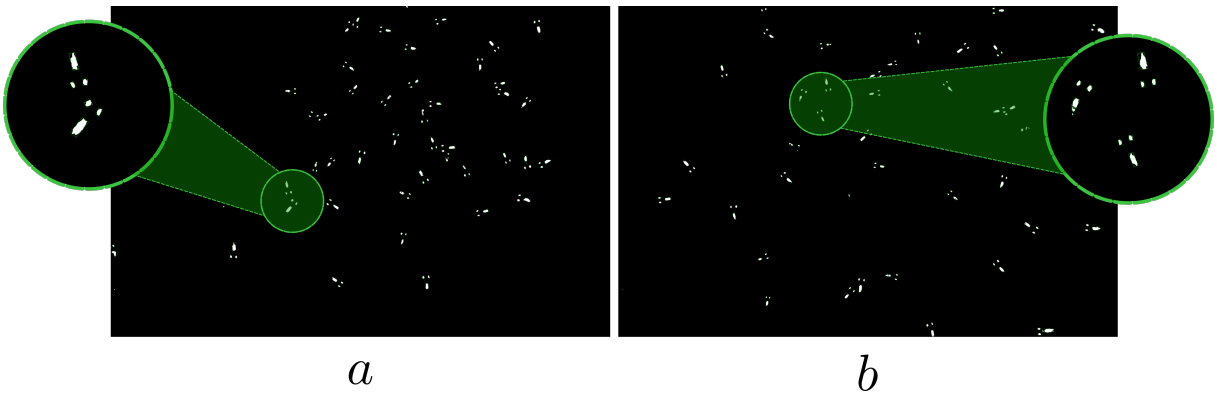


Figura 4.8: Resultado función *findContours()*:img1 (a), img2 (b)

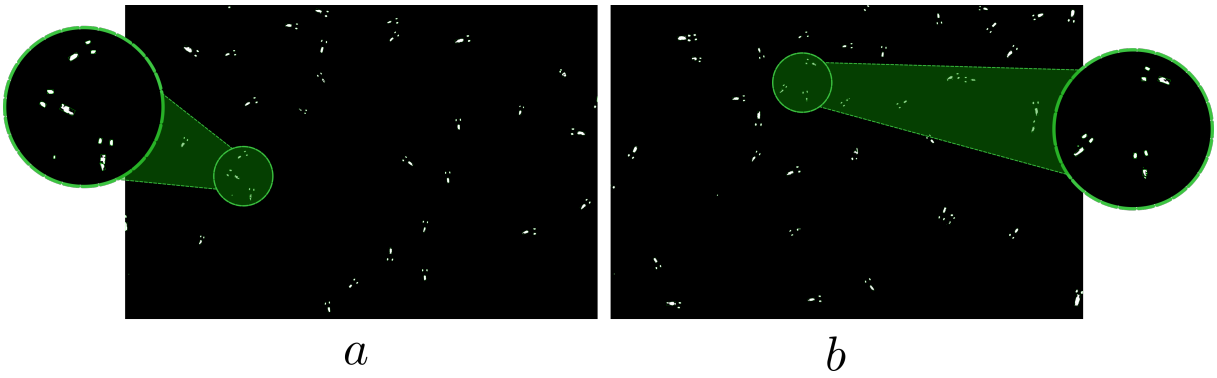


Figura 4.9: Resultado función *findContours()*:img3 (a), img4 (b)

4.4. Clasificación de áreas

Para hacer una clasificación más certera, generamos estadística de manera parcialmente manual sobre las áreas reconocidas. Usando la función OpenCV *findContours()*, se encuetran las medidas de las áreas de todos los objetos reconocidos en la imagen, y de manera manual se identifican y clasifican las áreas que corresponen a un ojo y aparatod digestivo. Las Figuras 4.10 y 4.11 muestran las distribuciones de éstas áreas.

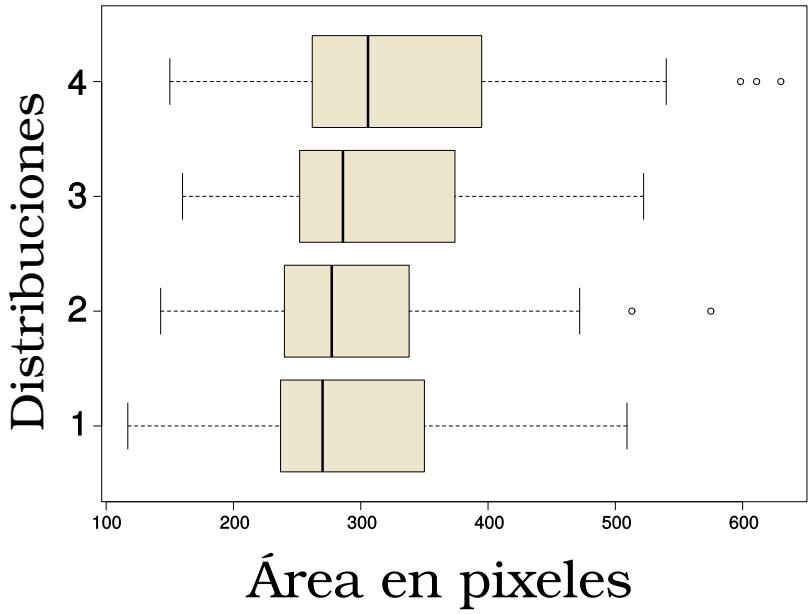


Figura 4.10: Gráfica de bigotes de los valores de áreas de ojos (1 → *img1*, 2 → *img2*, 3 → *img3*, 4 → *img4*)

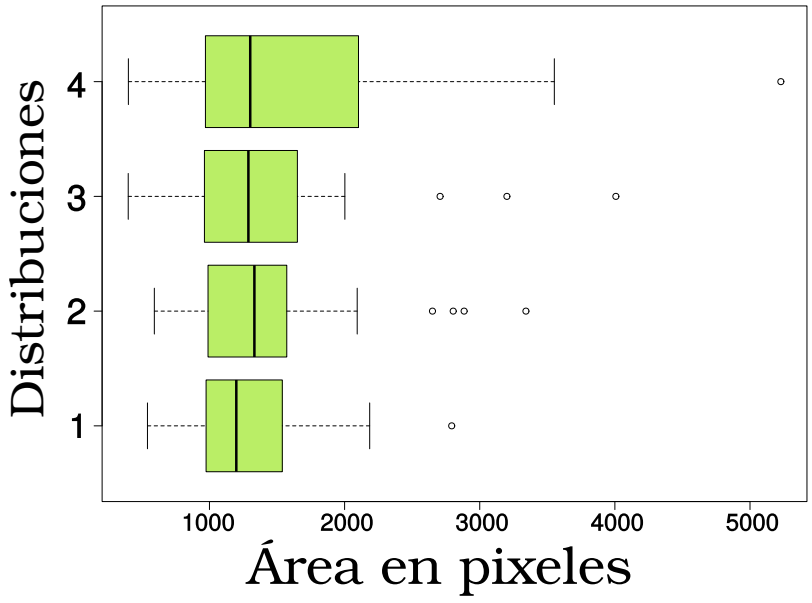


Figura 4.11: Gráfica de bigotes de los valores de áreas de los aparatos digestivos (1 → *img1*, 2 → *img2*, 3 → *img3*, 4 → *img4*)

De la misma manera para agregar una característica más en la clasificación, hacemos uso de la estadística y medimos de manera parcialmente manual las relaciones $\frac{base}{altura}$ reconocidas, las Figuras 4.12 y 4.13 muestran éstas distribuciones.

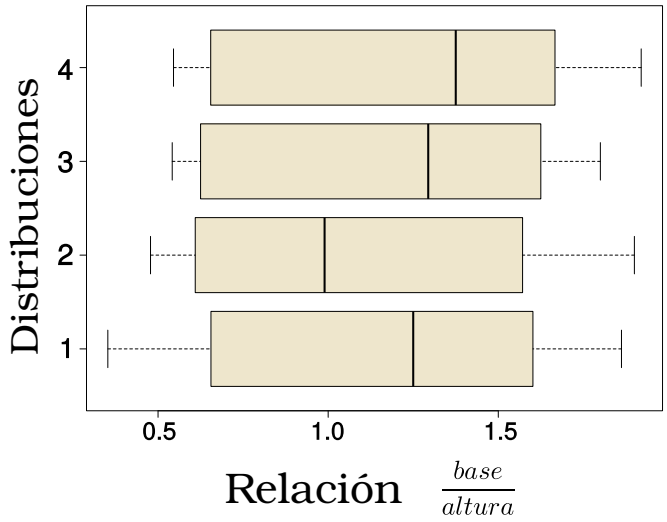


Figura 4.12: Gráfica de bigotes de los valores de la relación $\frac{base}{altura}$ de ojos (1 $\rightarrow$ *img1*, 2 $\rightarrow$ *img2*, 3 $\rightarrow$ *img3*, 4 $\rightarrow$ *img4*)

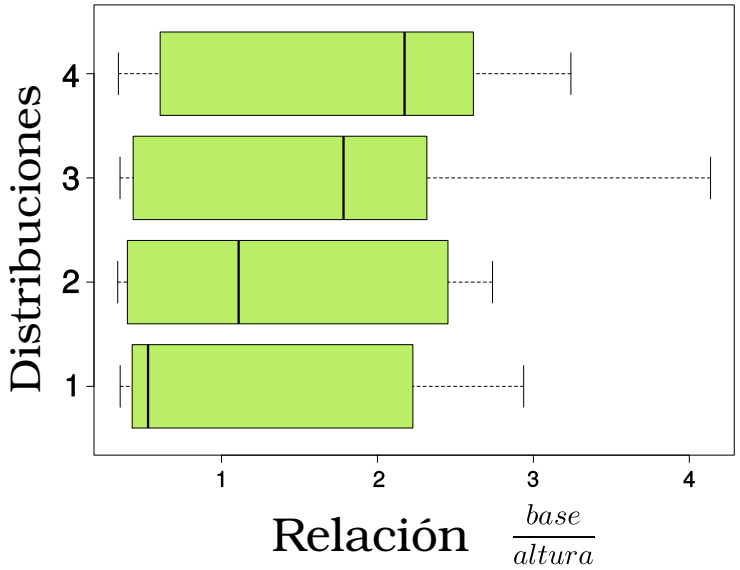


Figura 4.13: Gráfica de bigotes de los valores de la relación $\frac{base}{altura}$ de los aparatos digestivos (1 $\rightarrow$ *img1*, 2 $\rightarrow$ *img2*, 3 $\rightarrow$ *img3*, 4 $\rightarrow$ *img4*)

De esta manera podemos inferir que si el área localizada $A1$ está entre (área medida en pixeles):

- $117 \leq A \leq 630$ y que ademas su relación R $(\frac{base}{altura})$ $0.35 \leq R \leq 1.92$, esta área se cataloga como un ojo.
- $398 \leq A \leq 5228$ y que ademas su relación R $(\frac{base}{altura})$ $0.33 \leq R \leq 4.13$, esta área se cataloga como un aparato digestivo.

De tal manera que si el área $A1$ es un ojo se almacena la posición del centro de dicha área en el vector de posiciones L_1, y si es clasificado como un aparato digestivo, en el vector L_2. Así, podemos descartar rápidamente el *ruido*, y poder así realizar una detección más certera.

4.5. Patrón triangular (método exacto)

Con la finalidad de demostrar los resultados de la Sección 3.6.2, se generan de manera aleatoria 10 casos diferentes con 20, 40, 60, 100, 140 y 200 patrones sintéticos, y se evalua el desempeño del método exacto propuesto. Los resultados son mostrados en las Figuras 4.14, 4.15 y 4.16 que son un ejemplo de cada caso mencionado anteriormente.

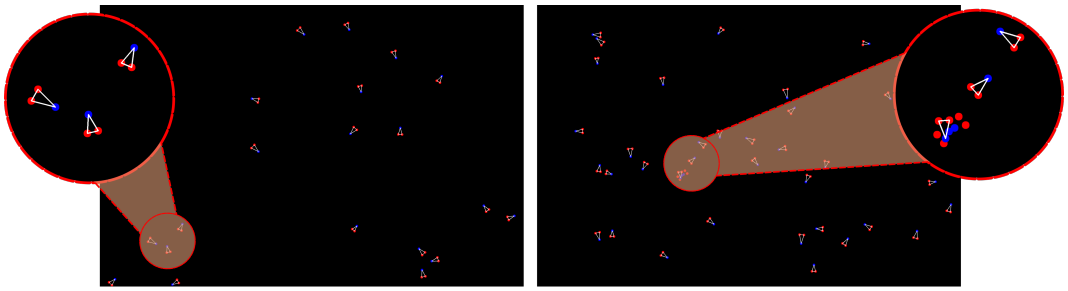


Figura 4.14: 20 patrones (a), 40 patrones (b)

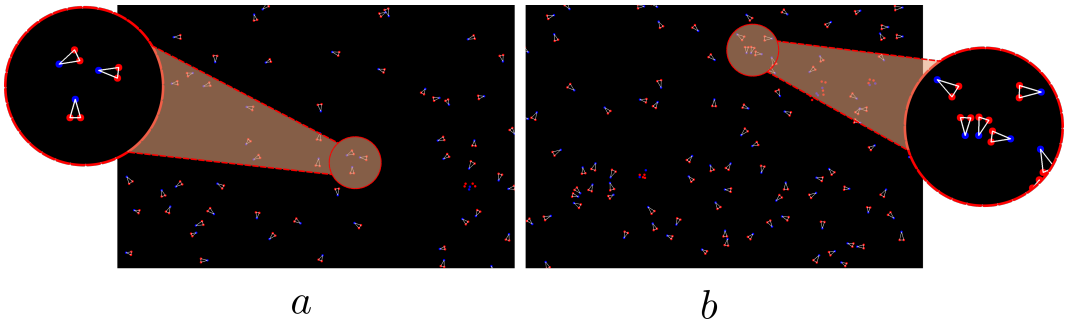


Figura 4.15: 60 patrones (a), 100 patrones (b)

La Figura 4.17 muestra el porcentaje de certeza en base a la cantidad de patrones a evaluar. La Figura 4.18 muestra el tiempo de procesamiento para evaluar los casos creados.

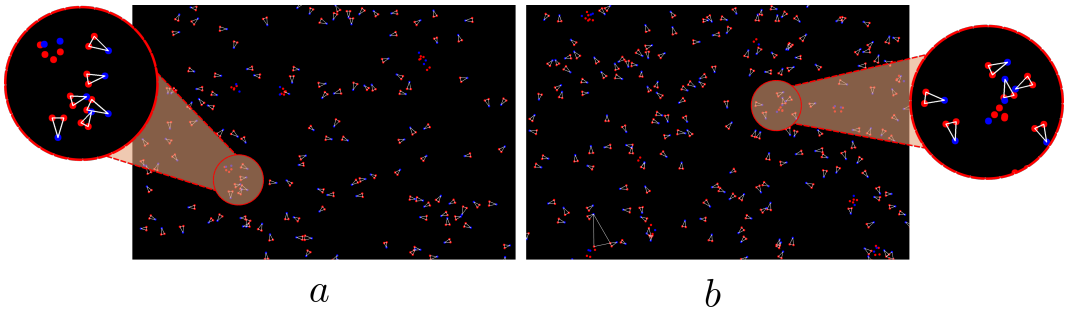


Figura 4.16: 140 patrones (a), 200 patrones (b)

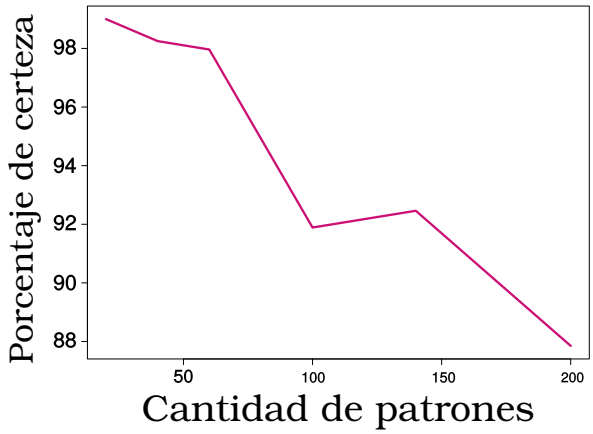


Figura 4.17: Porcentaje de certeza según la cantidad de patrones a evaluar (método exacto)

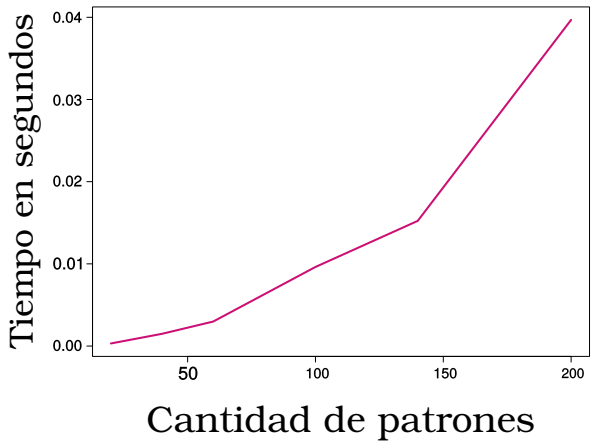
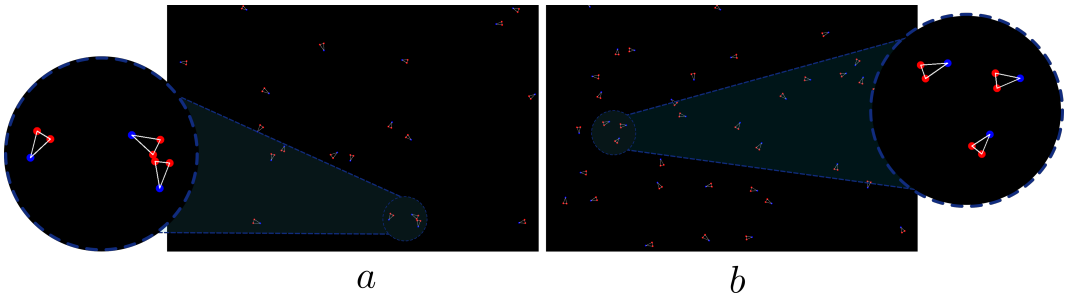


Figura 4.18: Tiempo de procesamiento según la cantidad de patrones a identificar (método exacto)

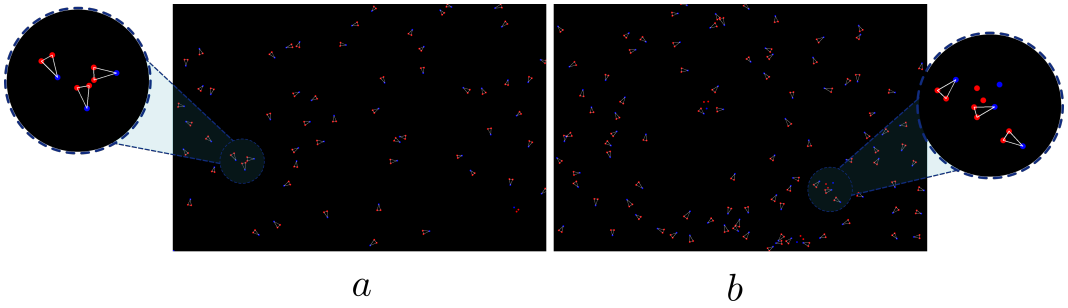
4.6. Patrón triangular (metaheurística)

A fin de mostrar los resultados de la Sección 3.6.3, se aplica el método propuesto sobre las 10 instancias diferentes con 20, 40, 60, 100, 140 y 200 patrones sintéticos con la intención de evaluar la eficiencia y certeza del método propuesto, para simplificar mostramos una imagen resultante como se muestran en las Figuras 4.19, 4.20 y 4.21.



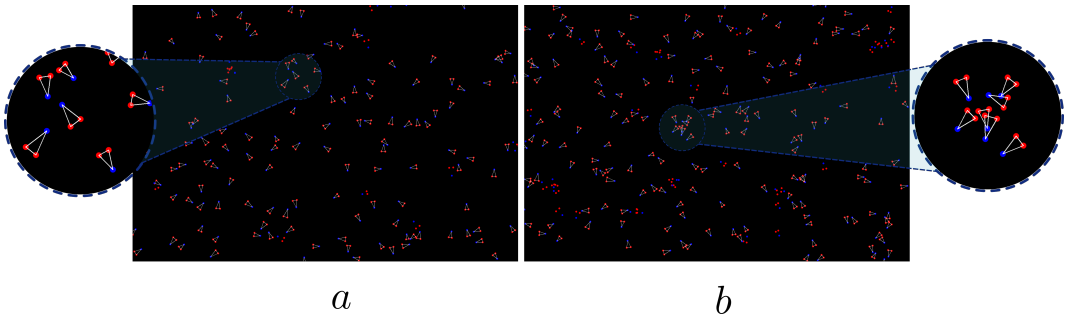
a b

Figura 4.19: 20 patrones (a), 40 patrones (b)



a b

Figura 4.20: 60 patrones (a), 100 patrones (b)



a b

Figura 4.21: 140 patrones (a), 200 patrones (b)

Para mostrar la eficiencia del método propuesto se muestra en la Figura 4.22 la gráfica de los porcentajes obtenidos de certeza contra la cantidad de patrones a detectar.

Así como también los tiempos de procesamiento según la cantidad de patrones a encontar (Figura 4.23).

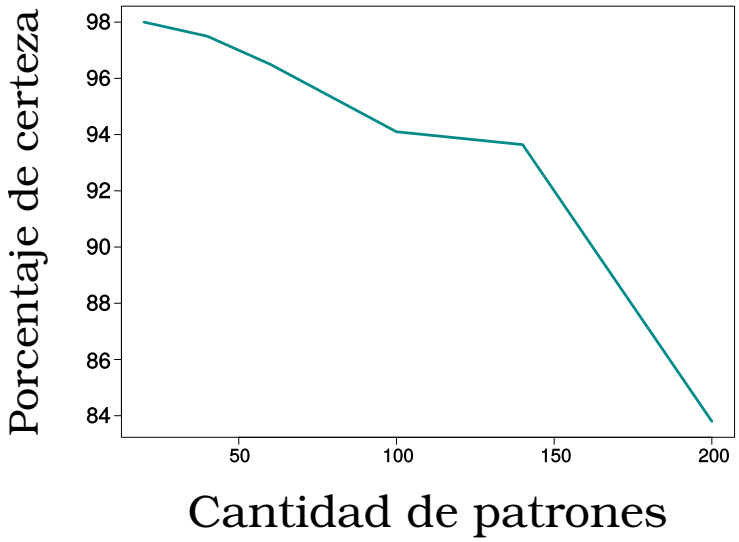


Figura 4.22: Tendencia del porcentaje de certeza según la cantidad de patrones a evaluar usando una metaheurística

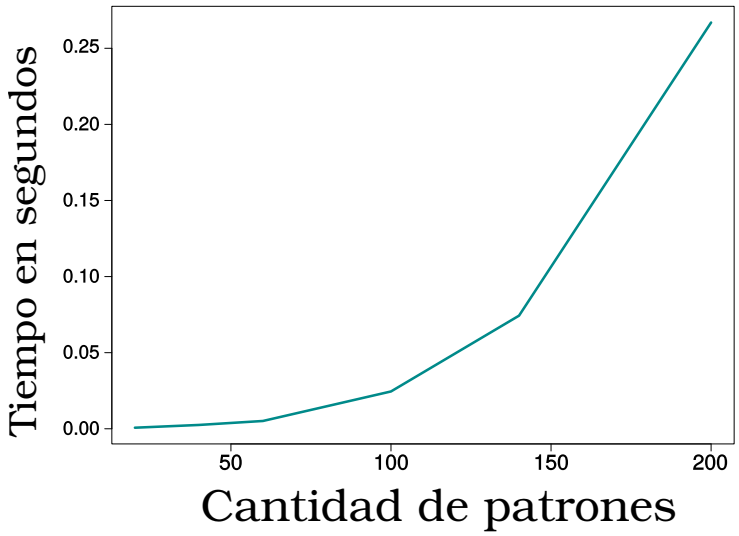


Figura 4.23: Tendencia del tiempo de procesamiento según la cantidad de patrones a identificar usando una metaheurística

4.7. Método integrado

Para este proceso mostramos 2 versiones de nuestra propuesta, para esto las primeras versiones muestran los resultados usando el método exacto.

4.7.1. Versión 1 (implementación sin tomar en cuenta casos especiales)

Utilizando los parámetros obtenidos y mencionados en la Sección 4.4, se muestran los resultados usando como imágenes de entrada las mostradas al principio de este capítulo. Los resultados son mostrados en las Figuras 4.24, 4.25.

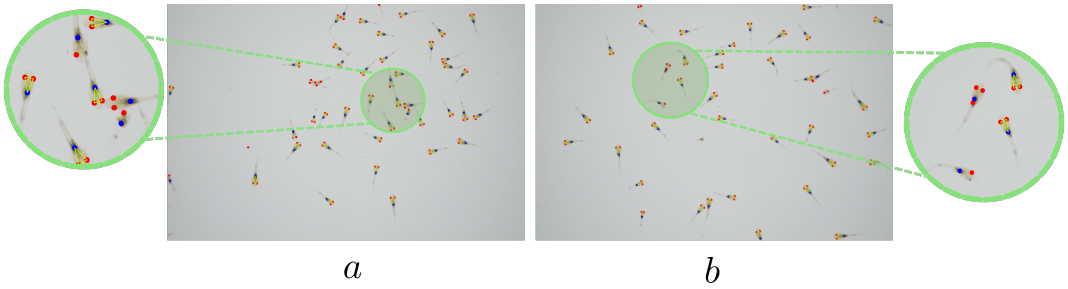


Figura 4.24: Resultados V 1.0: *img1* (a), *img2* (b)

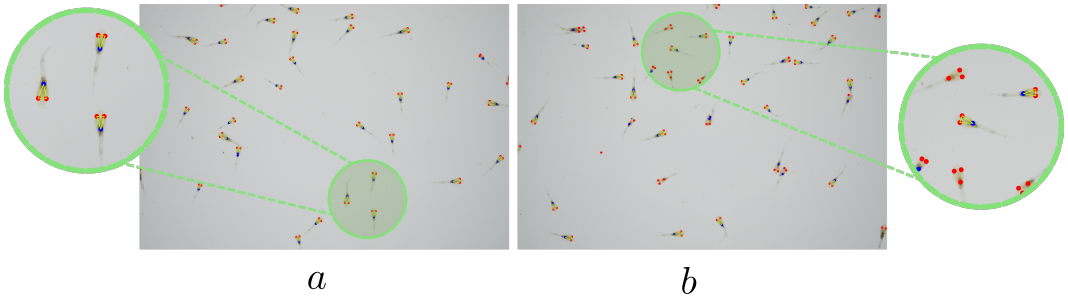


Figura 4.25: Resultados V 1.0: *img3* (a), *img4* (b)

4.7.2. Versión 1.1 (implementación tomando en cuenta casos especiales)

En esta versión se implementa lo mencionado en la Sección 3.6.4, para solucionar los 5 casos especiales identificados. Los resultados son mostrados en las Figuras 4.26, 4.27.

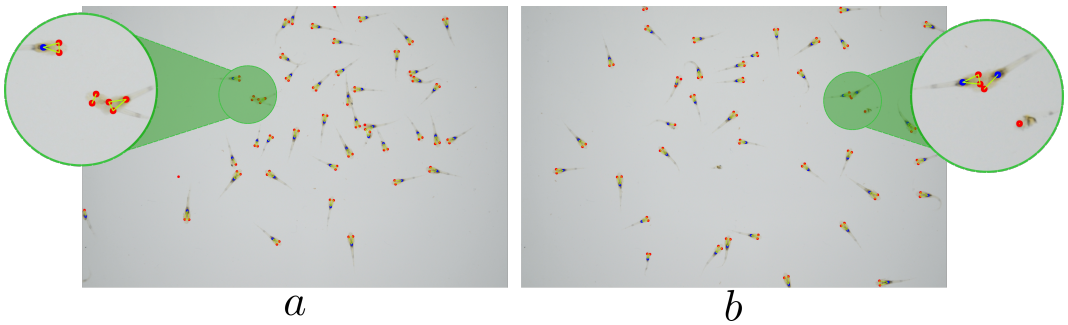
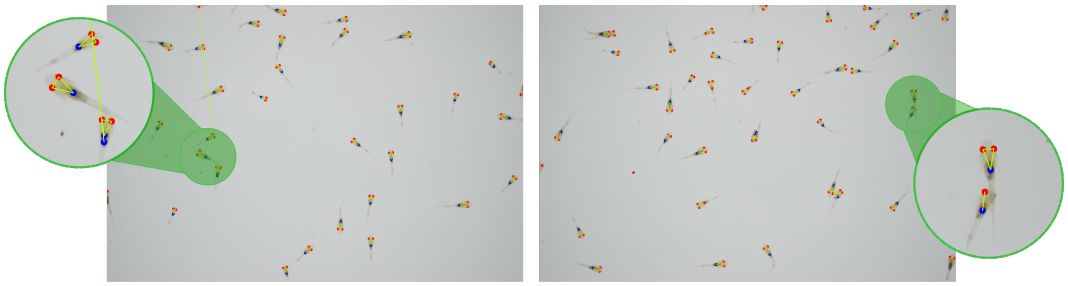


Figura 4.26: Resultados V 1.1: *img1* (a), *img2* (b)



a b

Figura 4.27: Resultados V 1.1: *img3* (a), *img4* (b)

Así mismo con la intención de poder cuantificar la certeza del algoritmo propuesto, se aplica la implementación sobre un banco de imágenes de 30 muestras reales (incluyendo las 4 anteriores). Se mide el tiempo de procesamiento y el porcentaje de certeza, la Tabla 4.1 muestra los porcentajes de certeza en estas 30 imágenes, así como también el tiempo en segundos que toma al pre-procesamiento de la imágen (pasar a escala de grises, binarizar, obtener las áreas y sus distribuciones, y por último clasificar dichas áreas encontradas), el tiempo en segundos del reconocmiento del patrón triangular y el de la revisión de CE. También se muestran los promedios de cada dato medido.

Tabla 4.1: Conteos manuales, resultados de los conteos usando el algoritmo V 1.1, y tiempos de procesamiento

	Conteo manual			Conteo algoritmo			Tiempos de procesamiento		
Imagen	**Larva ideal**	**CE**	**Total**	**Larva ideal**	**CE**	**% de Certeza**	**Preparación imagen**	**Reconocimiento del patrón**	**Total**
img1	39	7	46	39	7	100	0.36	0.0023	0.37
img2	36	5	41	36	5	100	0.29	0.0034	0.29
img3	28	5	33	28	5	100	0.37	0.0009	0.37
img4	34	6	40	34	6	100	0.58	0.0045	0.58
img5	35	3	38	35	3	100	0.32	0.0014	0.32
img6	35	5	40	35	4	98	0.35	0.0015	0.36
img7	31	2	33	30	2	97	0.37	0.0016	0.37
img8	34	6	40	33	6	98	0.36	0.0015	0.36
img9	13	1	14	13	1	100	0.26	0.0003	0.26
img10	12	0	12	12	0	100	0.35	0.0002	0.35
img11	18	5	23	18	5	100	0.26	0.0006	0.26
img12	15	3	18	15	3	100	0.32	0.0004	0.32
img13	7	2	9	7	2	100	0.26	0.0002	0.26
img14	12	4	16	11	5	100	0.25	0.0003	0.25
img15	14	3	17	14	3	100	0.30	0.0005	0.30
img16	13	5	18	14	4	100	0.31	0.0005	0.31
img17	14	3	17	14	3	100	0.27	0.0005	0.27
img18	16	3	19	16	3	100	0.25	0.0010	0.25
img19	10	4	14	10	4	100	0.35	0.0016	0.35
img20	14	3	17	14	2	96	0.35	0.0015	0.35
img21	11	2	13	11	2	100	0.34	0.0010	0.34
img22	16	3	19	16	2	94	0.27	0.0021	0.27
img23	19	8	27	19	7	94	0.33	0.0011	0.33
img24	26	7	33	24	8	97	0.28	0.0022	0.29
img25	20	10	30	18	11	97	0.35	0.0010	0.35
img26	16	10	26	14	11	96	0.26	0.0009	0.26
img27	18	9	27	18	9	100	0.26	0.0014	0.26
img28	22	5	27	22	4	96	0.31	0.0010	0.31
img29	23	6	29	22	6	97	0.36	0.0011	0.36
img30	21	7	28	21	7	100	0.27	0.0009	0.28
					Promedios	**98.6**	**0.32**	**0.0013**	**0.32**

4.8. Discusiones

En este trabajo se presentó un algoritmo capaz de clasificar y contar larvas de peces en una imagen, con un error promedio no mayor al 2 %, y un tiempo no mayor a 0.4 segundos. Cumpliendo así el objetivo de reducir el tiempo reportado por Martínez de 45 segundos promedio. Los algoritmos fueron ejecutados en una estación de trabajo con las mismas características del equipo usado por Martínez, debido a esto la comparación puede hacerce de manera directa. Ser capaz de hacer una reducción de tiempo a una fracción de menor del 1 % del reportado por Martínez, cumple y supera nuestras espectativas. Como podemos observar en la Tabla 4.1, en la columna "Reconocimiento patrón"de los tiempos de procesamiento, hay una relación directa entre el número de larvas a detectar y el tiempo que le toma al algoritmo en reconocerlas; también se pudo ver que el porcentaje de certeza se ve afectado por la cantidad de CE en la imagen. Además, podemos agregar que, el haber desarrollo un generador sintético de patrones triangulares nos da la posibilidad de generar un dato extremadamente importante: la frontera límite sobre la cantidad de larvas en la muestra, donde el método propuesto genera resultados eficientes, y a partir de qué número la eficiencia de nuestra propuesta empieza a decaer.

Capítulo 5

Conclusiones Generales

En este trabajo de tesis se describe la implementación de algoritmos de conteo y clasificación de larvas usando las características geométricas de larvas de peces, a través de Procesamiento Digital de Imágenes. Se identifica un patrón triagular encontrado, caracterizándolo por sus propiedades geométricas y se propone un generador de patrones sintético para lo cual se crean instancias de diferente número de patrones; de la misma manera se hace uso del modelo de islas de Algoritmos Genéticos, para reconocer dicho patrón triangular. Se establece un límite donde la certeza del algoritmo de conteo es reducida considerablemente, generando un parámetro para saber qué densidad de individuos por muestra es deseable al momento de la captura.

Se cumplieron los objetivos planteados al inicio de este trabajo:

Objetivo general:
Diseñar y evaluar un algoritmo para el conteo eficiente de larvas de peces a través del procesamiento digital de imágenes. Se propusieron 2 versiones de algoritmos que pertimen realizar una clasificación y un conteno certero.

Objetivos específicos

- *Establecer parámetros de referencia:*
    - Banco de imágenes a procesar (Se establace un banco de 30 imágenes de larvas de peces y 40 casos creados por el GASPT)
    - Tipo representación de pixeles a trabajar (Trabajamos con escala de grises)
    - Hacer una revisión de las diferentes metaheurísticas en PDI, especificando el uso de GA en el reconocimiento de un patrón definido (Se hizo una revisión de literatura y se hizo uso de GA obteniendo resultados satisfactorios)
    - Hacer uso de python y C++ como lenguaje de programación para implementar los algoritmos (Los algoritmos fueron implementados principalmente en C++)
    - Hacer uso de la paquetería de visión por computadora OpenCV para poder procesar las imágenes (Se utiliza OpenCV para auxiliar en el Procesamiento de Imágenes)

- *Hacer mediciones de tiempo de procesamiento*
  - Identificar cuáles de las variables tienen un mayor impacto sobre la eficiencia de nuestros algoritmos (La resolución de las imágenes y la cantidad de especímenes en la muestra)
  - Comparar métodos exactos contra la metaheurística propuesta (Encontrando que la metaheurística propuesta tiene un impacto positivo a partir de 100 especímenes en la muestra)
  - Comparar los tiempos de procesamiento obtenidos y el porcentaje de certeza, con los reportados por Martínez en su investigación [1] (Se disminuye el tiempo promedio a menos del 1 % de lo reportado por Martínez)

5.1. Aportaciones

Entre las principales aportaciones de éste trabajo se pueden mencionar:

- Se implementó un contador eficiente de larvas de peces (totoabas).
- Se hace una revisión de literatura dónde se comparan las diferentes Metaheurísticas implementadas en PDI
- Implementación del sistema de control automático para captura de imágenes
- Desarrollo de un generador aleatorio-sintético de patrones triagulares
- Artículo sometido en MethodsX: “Method to extract an enhanced cervical vertebrae area from a digital X-Ray image”

5.2. Trabajo a futuro

- Implementación de algoritmo propuesto usando GPU, haciendo uso de CUDA o una paquetería que haga uso eficiente de la tarjeta gráfica; comparar con los tiempos obtenidos en este momento.
- Hacer pruebas del algoritmo extrayendo cuadros (*frames*) de video, para verificar si se puede hacer un conteo en tiempo real.
- Verificación del patrón triangular en otras larvas de peces, ya que solo se cuenta con un banco de imágenes de larvas de *Totaba macdonali*, para verificar el impacto del generador sintético.

Glosario

Acrónimos

PDI	*Procesamiento Digital de Imágenes*
GA	*Algoritmos Genéticos*
ANN	*Redes Neuronales Artificales*
FA	*Autómatas Finitos*
CA	*Autómatas Celulares*
EA	*Algoritmos Evolutivos*
CED	*Detector de Bordes Canny*
SCED	*Detector de Bordes Shen-Castan*
SA	*Recocido Simulado*
TS	*Búsqueda Tabú*
PSO	*Optimización por Enjambre de Partículas*
CoEA	*Algoritmo Co-evolutivo*
CCGA	*Algoritmo Genético Co-evolutivo Cooperativo*
LS-SVM	*Marquina de Soporte Vectorial por Mínimos Cuadrados*
CHT	*Transformada de Hough para Círculos*
GASPT	*Generador Aleatorio-Sintético de Patrones Triangulares*
EMST	*Árbol de Mínima Expansión Euclidiana*
CE	*Casos Especiales*

Bibliografía

[1] Ernesto Martínez. Sistema de visión por computadora para contabilizar larvas de peces. Master's thesis, Universidad Autónoma de Baja California, 2015.

[2] Kenneth Dawson-Howe. *A practical introduction to computer vision with opencv*. John Wiley & Sons, 2014.

[3] Sihem Slatnia, Mohamed Batouche, and Kamal E Melkemi. Evolutionary cellular automata based-approach for edge detection. In *International Workshop on Fuzzy Logic and Applications*, pages 404–411. Springer, 2007.

[4] Baljit Singh and Amar Partap Singh. Edge detection in gray level images based on the shannon entropy 1. 2008.

[5] Tianzi Jiang and Faguo Yang. An evolutionary tabu search for cell image segmentation. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 32(5):675–678, 2002.

[6] Hao Gao, Wenbo Xu, Jun Sun, and Yulan Tang. Multilevel thresholding for image segmentation through an improved quantum-behaved particle swarm algorithm. *IEEE Transactions on Instrumentation and Measurement*, 59(4):934–946, 2010.

[7] Victor Ayala-Ramirez, Carlos H. Garcia-Capulin, Arturo Perez-Garcia, and Raul E. Sanchez-Yanez. Circle detection on images using genetic algorithms. *Pattern Recognition Letters*, 27(6):652–657, 2006.

[8] S Abdel-Khalek, Anis Ben Ishak, Osama A Omer, and A-SF Obada. A two-dimensional image segmentation method based on genetic algorithm and entropy. *Optik-International Journal for Light and Electron Optics*, 131:414–422, 2017.

[9] Mitchell A Potter and Kenneth A De Jong. A cooperative coevolutionary approach to function optimization. In *International Conference on Parallel Problem Solving from Nature*, pages 249–257. Springer, 1994.

[10] YH Toh, TM Ng, and BK Liew. Automated fish counting using image processing. In *Computational Intelligence and Software Engineering, 2009. CiSE 2009. International Conference on*, pages 1–5. IEEE, 2009.

[11] Liangzhong Fan and Ying Liu. Automate fry counting using computer vision and multi-class least squares support vector machine. *Aquaculture*, 380:91–98, 2013.

[12] Richard E. Woods Rafael C. Gonzalez. *Digital Image Processing*. Pearson, 3rd edition edition, 2007.

[13] J. R. Parker. *Algorithms for image processing and computer vision*. Wiley, 2 edition, 2010.

[14] Jaroslaw Skaruz, Franciszek Seredynski, and Anna Piwonska. Two-dimensional patterns and images reconstruction with use of cellular automata. *Journal of Supercomputing*, 69(1):9–16, 2014.

[15] Xitao Zheng and Yongwei Zhang. A fish population counting method using fuzzy artificial neural network. In *Progress in Informatics and Computing (PIC), 2010 IEEE International Conference on*, volume 1, pages 225–228. IEEE, 2010.

[16] D. Snyers and Y. Pétillot. Image processing optimization by genetic algorithm with a new coding scheme. *Pattern Recognition Letters*, 16(8):843–848, 1995.

[17] Nobuyuki Otsu. A threshold selection method from gray-level histograms. *IEEE transactions on systems, man, and cybernetics*, 9(1):62–66, 1979.

[18] John Canny. A computational approach to edge detection. *IEEE Transactions on pattern analysis and machine intelligence*, (6):679–698, 1986.

[19] Ilhem BoussaïD, Julien Lepagnot, and Patrick Siarry. A survey on optimization metaheuristics. *Information Sciences*, 237:82–117, 2013.

[20] Hin Leong Tan, Saul B Gelfand, and Edward J Delp. A cost minimization approach to edge detection using simulated annealing. In *Computer Vision and Pattern Recognition, 1989. Proceedings CVPR'89., IEEE Computer Society Conference on*, pages 86–91. IEEE, 1989.

[21] Fred Glover. Future paths for integer programming and links to artificial intelligence. *Computers & operations research*, 13(5):533–549, 1986.

[22] Russell Eberhart and James Kennedy. A new optimizer using particle swarm theory. In *Micro Machine and Human Science, 1995. MHS'95., Proceedings of the Sixth International Symposium on*, pages 39–43. IEEE, 1995.

[23] David A. Coley. *An Introduction to Genetic Algorithms for Scientists and Engineers*. World Scientific, har/dskt edition, 1997.

[24] John Henry Holland. *Adaptation in natural and artificial systems: an introductory analysis with applications to biology, control, and artificial intelligence*. MIT press, 1992.

[25] Constantino Tsallis. Generalized entropy-based criterion for consistent nonparametric testing. *preprint*, 1993.

[26] Alfréd Rényi. On measures of entropy and information. Technical report, HUNGARIAN ACADEMY OF SCIENCES Budapest Hungary, 1961.

[27] Margrit Betke and Nicholas C Makris. Fast object recognition in noisy images using simulated annealing. In *Computer Vision, 1995. Proceedings., Fifth International Conference on*, pages 523–530. IEEE, 1995.

[28] Shmuel Peleg, Danny Keren, and Limor Schweitzer. Improving image resolution using subpixel motion. *Pattern recognition letters*, 5(3):223–226, 1987.

[29] Manu Pratap Singh and Rinku Sharma Dixit. Optimization of stochastic networks using simulated annealing for the storage and recalling of compressed images using som. *Engineering Applications of Artificial Intelligence*, 26(10):2383–2396, 2013.

[30] Xin-She Yang and João Paulo Papa. *Bio-inspired computation and applications in image processing*. Academic Press, 2016.

[31] M Omran, Andries Petrus Engelbrecht, and A Salman. Particle swarm optimization method for image clustering. *International Journal of Pattern Recognition and Artificial Intelligence*, 19(03):297–321, 2005.

[32] M Omran, Ayed Salman, and Andries P Engelbrecht. Image classification using particle swarm optimization. In *Proceedings of the 4th Asia-Pacific conference on simulated evolution and learning*, volume 1, pages 18–22. Singapore, 2002.

[33] Markus Gudmundsson, Essam A El-Kwae, and Mansur R Kabuka. Edge detection in medical images using a genetic algorithm. *IEEE transactions on medical imaging*, 17(3):469–474, 1998.

[34] Sourav De, Siddhartha Bhattacharyya, and Paramartha Dutta. Automatic magnetic resonance image segmentation by fuzzy intercluster hostility index based genetic algorithm: An application. *Applied Soft Computing*, 47:669–683, 2016.

[35] W Daniel Hillis. Co-evolving parasites improve simulated evolution as an optimization procedure. *Physica D: Nonlinear Phenomena*, 42(1-3):228–234, 1990.

[36] Darrell Whitley. An executable model of a simple genetic algorithm. In *Foundations of genetic algorithms*, volume 2, pages 45–62. Elsevier, 1993.

[37] Simon Just Kjeldgaard Pedersen. Circular hough transform. *Aalborg University, Vision, Graphics, and Interactive Systems*, 123:123, 2007.

[38] Structural analysis and shape descriptors — opencv 2.4.13.6 documentation. https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html?highlight=findcontours#findcontours.

[39] Franco P Preparata and Michael I Shamos. *Computational geometry: an introduction*. Springer Science & Business Media, 2012.