$\mathcal{DIFU}_{100}ci@$
Revista de Difusión Científica, Ingeniería y Tecnologías
Vol. 16, No. 3, Septiembre - Diciembre 2022
ISSN:2007-3585
Artículo arbitrado
Publicado: 31 de diciembre de 2022

Parallel segmentation algorithm over aquaculture images

Algoritmo de segmentación paralela en imágenes de acuicultura

Yael Alejandro Santana-Michel[1], **María Guadalupe Sanchéz-Cervantes**[*1], **Marco Antonio Meza-Aguilar**[1], and **Daniel Fajardo-Delgado**[1]

[1] *TecNM/Instituto Tecnológico de Ciudad Guzmán, División de Estudios de Posgrado e Investigación,*
Av. Tecnológico 100, Ciudad Guzmán, Jalisco , 49100.
{m21290940, maria.sc1, marco.ma, daniel.fd}@cdguzman.tecnm.mx

Abstract

Within the field of aquaculture, when working with image analysis, the correct execution of preprocessing and segmentation algorithms for the selection of fish regions becomes a primary factor. Depending on the case, the number of images to be processed by these algorithms can significantly increase the time required to obtain the final result, thus delaying the ability to provide results. When it comes to processing large amounts of data, the paradigm of parallelization has given important results, reducing the execution time to complete complex tasks. In this article, it is proposed to carry out the preprocessing and segmentation processes in parallel, trying to optimize the time required to obtain the set of images prepared for their subsequent analysis. A comparison was made in execution times with different computers, and the experimental results show that it is better to use four or six execution threads by parallelizing the image preprocessing and segmentation algorithm

***Keywords*—** parallelization, images segmentation, aquaculture

Resumen

Dentro del ámbito de la acuicultura, cuando se trabaja con análisis de imágenes, se vuelve un factor primordial la ejecución correcta de algoritmos de preprocesamiento y segmentación para selección de regiones de interés de los peces. Dependiendo el caso, la cantidad de imágenes a ser procesadas por estos algoritmos puede incrementar de manera significativa el tiempo requerido para obtener las imágenes segmentadas, retardando así, la capacidad de brindar resultados. Cuando se trata de procesamiento en grandes cantidades de datos el paradigma de la paralelización ha dado resultados importantes, reduciendo el tiempo de ejecución para completar tareas complejas. En este artículo se propone realizar el preprocesamiento y segmentación de imágenes de manera paralela, para optimizar el tiempo requerido para obtener el conjunto de imágenes para su análisis posterior. Se realizó una comparativa en tiempos de ejecución con distintos equipos de cómputo paralelo, y los resultados experimentales muestran que es mejor utilizar cuatro o seis hilos de ejecución al paralelizar el algoritmo de preprocesamiento y segmentación de imágenes.

***Palabras clave*—** paralelización, segmentación de imágenes, acuicultura

I. Introducción

La acuicultura es la actividad agrícola que se dedica a la cría de organismos acuáticos en cualquiera de sus fases de desarrollo. Dicha crianza supone la intervención del humano con el objetivo de aumentar la producción y calidad de organismos bajo el concepto de producir crías de óptima calidad, mejorar la resistencia a enfermedades, desarrollo de dietas y utilizar métodos de cultivo [1].

En este ámbito, el monitoreo del estado de los peces durante el cultivo puede ayudar a mejorar la rentabilidad de los productores al asegurar la salud del espécimen y reducir la probabilidad de pérdidas graves debido a

[*]Autor de correspondencia

enfermedades e incidentes de estrés, esto apoyado en que el patrón de coloración de la piel en los peces, no solo es representativo de las tasas de crecimiento, sino que también, muestra el bienestar de los peces [2], entonces, es ahí donde el análisis de imágenes puede ofrecer un aporte a este objetivo.

El procesamiento digital de imágenes es la disciplina que busca extraer información del mundo real de manera automática a partir de una imagen, de un conjunto o secuencia de imágenes, para lo cual se emplean diversos algoritmos operacionales sobre los píxeles que componen las imágenes. Esta extracción de características depende del proceso de detección de los elementos que constituyen una imagen. Para este propósito pueden ser usadas técnicas que permiten el reconocimiento de imágenes, y para ello el elemento clave es el filtrado realizado sobre las imágenes [3].

El curso general de todo sistema que implementa análisis sobre imágenes consta de las siguientes etapas: adquisición de imágenes, preprocesamiento, segmentación, detección de objetos y clasificación y, finalmente el análisis de imagen [4].

Durante la primera etapa de la adquisición de imágenes, existen factores físicos difíciles de controlar tales como: niveles bajos de iluminación, artefactos (elementos dentro de la imagen que pueden, o no, ser de utilidad) y niveles de contraste que afectan el análisis [5].

Estos problemas generalmente son creados durante las etapas de adquisición de la imagen, sin embargo, pueden ser mitigados con la aplicación de la etapa de preprocesamiento. Dicha etapa está constituida por técnicas de filtrado, posteriormente durante la etapa de segmentación se realiza la definición de regiones de interés sobre las imágenes, destacando únicamente las zonas de la imagen con la que la etapa de detección de objetos realizará su trabajo, definiendo los objetos que se utilizarán para la última etapa. Finalmente en la etapa de análisis, es donde se determina la manera en que serán clasificados los objetos encontrados dentro de la imagen [6].

El éxito de la etapa de segmentación sobre imágenes, en muchos casos, depende de la etapa de preprocesamiento, ya que se puede determinar de una mejor manera las regiones de interés. En la etapa de preprocesamiento se realizan diversas operaciones sobre los píxeles, del mismo modo, pueden ser aplicados diversos filtros sobre la imagen para mejorar la definición de objetos o resaltar características dentro de ésta [7].

El tiempo computacional que se invierte en realizar estas operaciones es considerable, y más cuando se toman en cuenta las imágenes de gran tamaño o cuando se requiere procesar varias imágenes.

Para el caso de procesar grandes cantidades de imágenes, así como imágenes con una gran cantidad de píxeles, el paradigma de programación paralela, o multiprocesamiento, puede ser una solución en donde cada uno de las unidades centrales de procesamiento (CPU, por sus siglas en inglés) puede realizar operaciones con cada imagen o píxel [8].

En la revisión de artículos realizada por los autores, se han encontrado trabajos que tratan el tema de la paralelización de segmentación de imágenes de distintas índoles, por ejemplo, en [9] los autores hacen uso de la paralelización por tarjeta gráfica para segmentación de imágenes de resonancias magnéticas, mientras que en [10] el autor usa una aproximación utilizando los hilos de trabajo de un procesador para trabajar imágenes geográficas de satélites. Este artículo propone una solución para el área de la acuicultura. Cabe mencionar que hasta donde conocen los autores, no hay trabajos que propongan algún algoritmo paralelo para realizar trabajos de segmentación en imágenes para acuicultura.

En este artículo se propone un algoritmo paralelo que realice el preprocesamiento y segmentación en imágenes de peces, más específicamente, sobre el pez llamado Lenguado Senegalés (*Solea senegalensis*). Además, para demostración práctica, se llevó acabo comparativas de los tiempos de ejecución con dos equipos de cómputo, así como con distintas configuraciones de hilos.

El artículo se organiza de la siguiente manera: en la sección II se describen las herramientas de software que fueron empleadas para el desarrollo del algoritmo propuesto. En la sección III se dan las pautas del paralelismo necesaria para entender el tema. En la sección IV se detalla la metodología empleada para el trabajo de preprocesamiento y segmentación paralela sobre las imágenes. En la sección V se describen las pruebas a realizar sobre las imágenes. Finalmente, en las secciones VI y VII se muestran los resultados y las conclusiones de las pruebas realizadas, respectivamente.

II. Herramientas de software

Para llevar a cabo los experimentos se utilizó el lenguaje de programación Python y la biblioteca libre de visión artificial OpenCV. A continuación se explica brevemente cada uno de ellos.

II.1. Python

Python es un lenguaje de alto nivel de uso general, esto significa que el código es compilado a código de bytes y ejecutado por el CPU, es consistente con el uso del paradigma de programación orientada a objetos, permitiendo escribir código tanto para pequeñas como grandes tareas. Además, debido a que Python puede ser extendido en C y C++, provee la velocidad de cómputo intensivo sin perder velocidad [11]. Por todo lo anterior, Python se ha convertido en el lenguaje de programación más popular para ciencia de datos y otras áreas emergentes [12].

II.2. OpenCV

La librería de visión artificial y código abierto (OpenCV, por sus siglas en inglés), proporciona un marco de trabajo de alto nivel para el desarrollo de aplicaciones de visión por computadora en tiempo real tales como: estructuras de datos, procesamiento y análisis de imágenes, análisis estructural de objetos, entre otras [13].

Este marco de trabajo facilita en gran manera el aprendizaje e implementación de distintas técnicas de visión por computador, tanto a nivel educacional como de investigación, aislando al desarrollador de las peculiaridades de los distintos sistemas de visión, así como las distintas maneras de manipulación de imágenes [13].

III. Paralelismo

El paralelismo es la ejecución de tareas de manera simultánea sobre distintos procesadores, esto puede lograrse cuando el hardware utilizado cuenta con un conjunto de procesadores con capacidad para ejecutar de manera coordinada un algoritmo. En general, dado un problema, es dividido en subproblemas que son resueltos simultáneamente por los procesadores disponibles [14].

III.1. Clasificación del paralelismo

En el trabajo [14] el autor menciona que la clasificación hecha por Flynn está centrada en la manera en que las instrucciones son ejecutadas sobre los datos, ya que, cualquier computadora opera ejecutando instrucciones sobre los datos y, de acuerdo con esto, se puede identificar cuatro clases básicas, las cuales se mencionan a continuación.

SISD

Es la arquitectura usada por la mayoría de los equipos de cómputo. Las instrucciones son ejecutadas una después de otra, una por ciclo de instrucción y la memoria afectada solo es usada para esa instrucción. Este tipo de ejecución de instrucciones suele utilizarse empleando la CPU del equipo de cómputo [14].

MISD

En este caso, un número de procesadores ejecutan un flujo de instrucciones distinto, sin embargo, estos comparten datos comunes entre sí. El mismo conjunto de datos es operado simultáneamente por los procesadores de manera que para un mismo conjunto de datos existen diversas salidas distintas, dependiendo de las instrucciones ejecutadas, y suele ser ejecutado utilizando tres tecnologías distintas: CPU, unidad de procesamiento gráfico (GPU, por sus siglas en inglés) y unidades de procesamiento tensorial (TPU, por sus siglas en inglés) [14].

SIMD

Para este caso, el equipo cuenta con un conjunto de procesadores idénticos con sus propias memorias, estos ejecutan la misma instrucción bajo el control de una unidad de control o procesador (*host*) sobre distintos datos. El *host* hace *broadcast* de la instrucción a ser ejecutada a cada uno de los procesadores paralelos, ejecutando simultáneamente las operaciones a realizar. Aunque este tipo de instrucciones puede ser ejecutado tanto en CPU, GPU o TPU, suele recomendarse ejecutar este tipo de instrucciones en CPU o TPU [14].

MIMD

Cada uno de los procesadores tiene su propio flujo de instrucciones y de datos, por lo que puede ejecutar un programa específico independientemente de los otros. Son las computadoras paralelas de propósito más general. Constan de *n* procesadores, cada uno con su propia unidad de control, donde cada procesador ejecuta una instrucción distinta sobre diferentes datos, en este tipo de sistemas puede, o no, existir una memoria compartida, así como un sistema de sincronización. Este tipo de instrucciones pueden ser ejecutados tanto por CPU, como GPU o TPU, sin embargo, se asocia más a ser ejecutados por GPU [14].

IV. Metodología empleada

Para realizar la segmentación de una imagen se propone el Algoritmo 1, el cual será descrito a continuación.

Algoritmo 1 Preprocesamiento y segmentación de la imagen.

Entrada: $imagen$
Salida: $imagenSegmentada$

1: $datosImagen \leftarrow imread(imagen)$
2: $imagenGris \leftarrow cvtColor(datosImagen)$
3: $imagenBinaria \leftarrow threshold(imagenGris, 127, 255, THRESH\_BINARY)$
4: $contornos \leftarrow findContours(imagenBinaria)$
5: **para todo** *contorno en contornos* **hacer**
6: **si** $area(contorno) > 100$ **entonces**
7: $pez \leftarrow contorno$
8: *romper ciclo*
9: **fin si**
10: **fin para**
11: $mascara \leftarrow zeros(datosImagen, uint8)$
12: $drawContours(mascara, pez, transparencia)$
13: $imagenSegmentada \leftarrow bitwise(datosImagen, mascara)$
14: **devolver** $imagenSegmentada$

El algoritmo recibe como entrada la imagen del pez, sobre la cual se realiza las operaciones de preprocesamiento y segmentación. La salida del algoritmo es la imagen del pez en la cual se eliminó el fondo, dejando únicamente la región de interés, es decir, la región que comprende el cuerpo del pez; el resultado esperado se puede observar en la Figura 1.

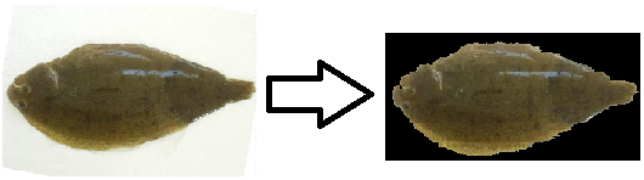


Figura 1: *Ejemplo de salida esperada del algoritmo*

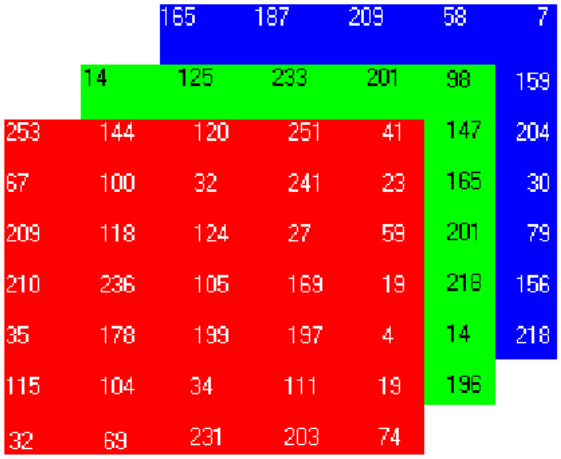


Figura 3: *Matriz tridimensional RGB*

IV.1. Preprocesamiento

Lectura de la imagen

Para realizar las operaciones pertinentes sobre cada una de las imágenes, se requiere leer los datos de ésta y almacenarlas de manera matricial para su manipulación; para este caso las imágenes se vuelven una matriz del ancho de píxeles por su alto, en donde cada elemento de la matriz representa un píxel. La representación matricial de una imagen se puede observar en la Figura 2. Además, si la imagen se encuentra en formato RGB (siglas en ingles de Red, Green, Blue), se agrega una tercera dimensión, en donde cada uno de los índices se representa por cada color, como se observa en la Figura 3.

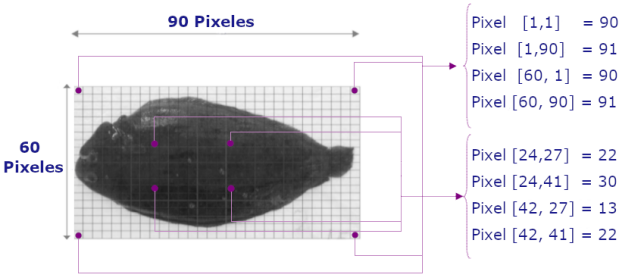


Figura 2: *Ejemplo de representación de una imagen en forma matricial*

La matriz de píxeles se obtiene utilizando la librería OpenCV de Python llamada *"cv2"*. El método a utilizar para leer la imagen es *"imread"* de *"cv2"*, en donde el parámetro "imagen" representa la ruta de la imagen a leer por el método, retornando un arreglo tridimensional con los valores RGB de la imagen.

Binarización

Una vez que se obtiene la información completa de la imagen, se busca hacer una detección de bordes a través del método de la binarización. En una imagen binaria cada uno de los píxeles asume un valor discreto, esencialmente dichos valores son 1 ó 0. Para poder hacer una binarización se requiere de trabajar una imagen en escala de grises, para lo cual, openCV cuenta un método para convertir una imagen en escala de grises a una imagen binaria, dicha secuencia de acciones se muestra en el Algoritmo 1 en las líneas 2 y 3.

El método *"cvtColor"* es utilizado para modificar la configuración de colores de la imagen y la cual recibe dos parámetros: la imagen (la cual es la imagen a modificar); y la configuración de color resultante (el cual para este caso se proporciona la constante de openCV *"COLOR_RGB2GRAY"* para convertir la imagen a escala de grises). Con el resultado *"imagenGris"*, se convierte la imagen binaria utilizando el método *"threshold"*, el cual recibe una imagen en escala de grises, un valor de color que es el umbral, el cual define la condición de conversión de color, después requiere de un valor de sobre-escritura (255 en este caso) y finalmente, el método de binarización a utilizar, dado por constantes de la librería openCV, siendo *"THRES_BINARY"* el seleccionado.

IV.2. Segmentación

Este paso del procesamiento de imágenes consta de operaciones para hacer una partición de la imagen en varias regiones, estas regiones representan la información necesaria para resolver el problema [4]. En este caso el

algoritmo hace uso de la binarización del paso anterior para señalar la presencia de objetos, para posteriormente seleccionar únicamente la región de interés para su respectivo análisis. En el Algoritmo 1 puede observarse dicha implementación en las líneas 4 a la 14. Se requiere importar la librería de *"numpy"*. El algoritmo toma la imagen binarizada y la utiliza para hacer la identificación de contornos, obtienendo la lista de contornos dentro de la imagen; posteriormente, elimina todos aquellos contornos que no son relevantes, seleccionando únicamente la región del cuerpo del pez, finalmente se crea una máscara de la imagen para posteriormente realizar una fusión entre la imagen original y la máscara, dejando únicamente el área de interés para su análisis.

IV.3. Paralelizando la ejecución

La paralelización del Algoritmo 1 se realiza con la finalidad de reducir el tiempo de ejecución requerido en las etapas de preprocesamiento y segmentación sobre imágenes de peces, haciendo uso del tipo de instrucciones SIMD de la clasificación de Flynn (explicado en la Sección III.1). En el Algoritmo 2 se muestra la paralelización.

Algoritmo 2 Paralelización de la segmentación

Entrada: $imagenes, procesos$
Salida: $imagenesSegmentadas$
1: *crear pilaProcesos indicando procesos*
2: $imagenesSegmentadas \leftarrow pilaProcesos.starmap(Algoritmo1, imagenes)$
3: *cerrar pilaProcesos*
4: **devolver** $imagenesSegmentadas$

El algoritmo hace uso de la clase *Pool*, perteneciente a la librería *multiprocessing*. Primero se crea una variable de resultados llamada *"imagenesSegmentadas"*, para posteriormente crear la pila de procesos con la cantidad de procesos indicada por el usuario, de esta manera pasar el Algoritmo 1 a todos los hilos disponibles para su procesamiento utilizando el método *"starmap"*, el cual requiere como parámetros el algoritmo a ejecutar y la información que será enviada, finalmente se guardan las imágenes segmentadas en la variable correspondiente.

V. Resultados experimentales

V.1. Hardware utilizado

Para realizar los experimentos se utilizaron dos equipos de uso general con especificaciones distintas entre sí. El equipo de cómputo número 1 cuenta con un procesador Intel Core i7-4610M con 4 núcleos y 4 hilos, 16 GB de memoria RAM DDR3 y disco duro de 250 GB, mientras que el equipo de cómputo número 2 cuenta con un procesador Ryzen 5-5600G con 6 núcleos y 12 hilos, 16 GB de memoria RAM DDR4 y disco duro de 1 TB. El objetivo de realizar la prueba con dos equipos distintos es hacer una comparativa con los resultados y reducir sesgos con los mismos.

V.2. Métricas de evaluación

Existen diversas métricas para evaluar a los algoritmos paralelos, sin embargo, para este trabajo se utilizó el *speedup* (velocidad comparativa en que se ejecutan los algoritmos), el cual se calcula utilizando la Ecuación (1).

$$Speedup = \frac{\text{tiempo secuencial}}{\text{tiempo paralelo}} \tag{1}$$

Además, es importante mencionar que se obtuvo el promedio del tiempo de ejecución (de 100 ejecuciones), para cada conjunto de imágenes procesadas.

V.3. Descripción de las pruebas

Para medir el desempeño del algoritmo se utilizaron cuatro conjuntos de imágenes de peces, específicamente hablando, son 61 imágenes del Lenguado Senegalés (Solea senegalensis), las cuales, fueron duplicadas con el propósito de simular *data sets* con grandes cantidades de imágenes por procesar; la cantidad de imágenes es de 100, 200, 600, 1000 y 1500 imágenes para cada conjunto. Cada uno de los conjuntos fue ejecutado 100 veces, de tal manera que se toma el tiempo de ejecución promedio de cada conjunto, así mismo, las pruebas del algoritmo paralelo se ejecutaron con distintas cantidades de procesos simultáneos (hilos -H-), las cuales son 2, 4 y 6, esto con la finalidad de realizar una comparativa con los tiempos de la ejecución.

VI. Resultados

Para cada uno de los equipos de cómputo se obtuvieron distintos resultados, después de la ejecución de las pruebas, los datos son congruentes con la teoría de la paralelización, en donde se obtiene un aumento en el rendimiento. Los datos de las pruebas ejecutadas en los equipos número 1 y 2 pueden ser observados en las tablas 1 y 2 respectivamente, marcando con **negritas** los tiempos de ejecución menores para completar la segmentación de las imágenes.

Para analizar el rendimiento, se graficaron los resultados de tal manera que se pueda observar la reducción de tiempos de ejecución, dichas gráficas se encuentran en la Figura 4.

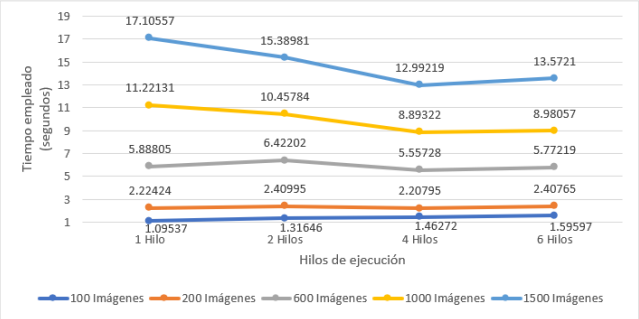
En la Figura *4a* pueden ser observados los tiempos de ejecución con distintos hilos de ejecución del equipo 1; se destaca que hay una reducción de tiempo de ejecución cuando se procesan más de 600 imágenes con pico mínimo al utilizar 4 hilos, además, esta reducción se hace

Tabla 1: *Tiempo de ejecución promedio en segundos equipo No 1*

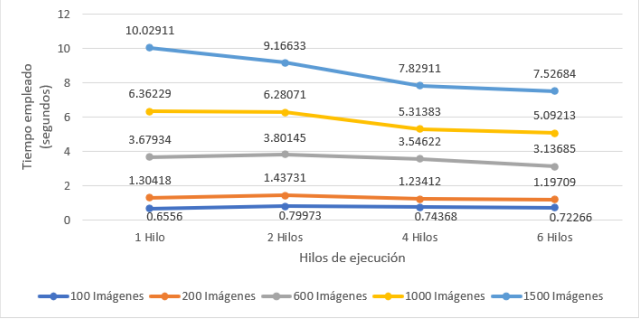
Imágenes	Secuencial	2 H	**4 H**	6 H
100	1.095	1.316	**1.462**	1.595
200	2.224	2.409	**2.207**	2.407
600	5.888	6.422	**5.557**	5.772
1000	11.221	10.457	**8.893**	8.9805
1500	17.105	15.389	**12.992**	13.572

Tabla 2: *Tiempo de ejecución promedio en segundos equipo No 2*

Imágenes	Secuencial	2 H	4 H	**6 H**
100	0.655	0.799	0.743	**0.722**
200	1.304	1.437	1.234	**1.197**
600	3.679	3.801	3.546	**3.136**
1000	6.362	6.281	5.313	**5.092**
1500	10.029	9.166	7.829	**7.526**



a) Tiempos de ejecución en segundos del equipo 1



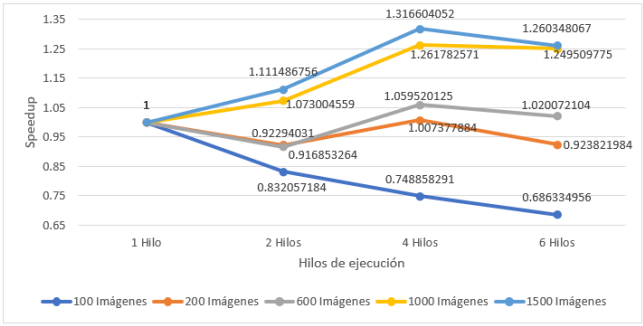
b) Tiempos de ejecución en segundos del equipo 2

Figura 4: *Tiempos de ejecución. a) Equipo 1 b) Equipo 2*

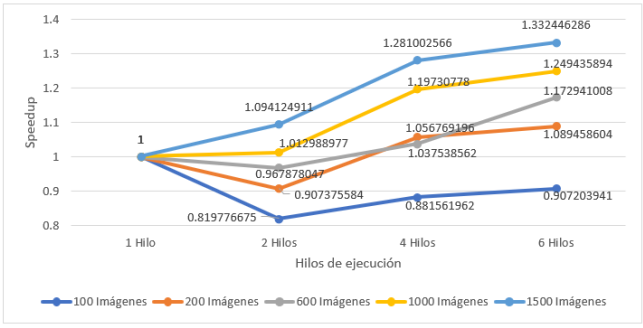
aún más notoria cuando se incrementa el número de imágenes.

En cambio, en la Figura *4b* se puede observar la reducción de tiempo del equipo número 2, en donde notamos una variación en las condiciones con respecto al equipo 1, para este caso el pico mínimo de tiempo de ejecución se da cuando se utilizan 6 hilos, sin embargo, la cantidad de imágenes a procesar para notar una reducción en tiempo es el mismo, siendo 600 imágenes.

En cuanto a lo que se refiere al *speedup*, se utilizaron los datos de los tiempos de ejecución para calcular el *speedup* y se graficaron para facilitar su análisis, estas gráficas incluyen la información de cada equipo de cómputo y se muestran en la Figura 5.



a) Speedup del equipo 1



b) Speedup del equipo 2

Figura 5: *Speedup. a) Equipo 1 b) Equipo 2*

Los resultados muestran que el *speedup* de cada equipo de cómputo varía dependiendo de los componentes que los integra, para este caso se puede observar, en la Figura *5a*, que en el equipo de cómputo número 1, el *speedup* más alto se obtiene cuando se cumplen dos condiciones: la primera es que la cantidad de imágenes a procesar debe ser igual o mayor a 600, esto es debido a que la gestión de los procesadores tiene repercusión en el tiempo de ejecución para cuando el número es menor a esa cantidad; y que la cantidad de procesadores empleados debe ser igual a la cantidad de procesadores físicos con los que cuenta el equipo, lo anterior debido a que la generación de hilos de ejecución superiores a la cantidad de procesadores físicos merma el rendimiento de los mismos. Para ilustrar el caso experimentado se utilizó el mejor caso, en donde procesando 1500 imágenes con 4 hilos se puede observar un *speedup* de 1.3166, lo cual indica una reducción de tiempo de casi una tercera parte.

En cambio, con el equipo de cómputo número 2, se genera un pequeño cambio en una de las condiciones, y este cambio es únicamente en la cantidad de hilos de ejecución necesaria para poder obtener un *speedup* destacado, para este caso, se utilizó el resultado de procesar

1500 imágenes, en donde utilizando 6 hilos de ejecución se obtiene un *speedup* de 1.3324, reduciendo, al igual que el equipo 1, el tiempo de ejecución a dos terceras partes que haciéndolo de manera secuencial, esto se puede observar en la Figura *5b*.

VII. Conclusiones

El algoritmo paralelo propuesto en este artículo se utiliza para llevar a cabo los procesos de preprocesamiento y segmentación en imágenes de peces, más específicamente, sobre el pez llamado Lenguado Senegalés (*Solea senegalensis*). El paralelismo resulta ser una buena opción para cuando es requerida la preparación de una gran cantidad de imágenes, permitiendo acelerar el proceso de tal manera que se pueda pasar a la etapa de análisis y clasificación más rápido.

Esto deja la conclusión de que se debe planear y comprobar los resultados de acuerdo con el equipo de cómputo a utilizar. Para realizar una buena ejecución paralela se debe considerar la cantidad de procesadores físicos con los que cuenta el equipo, además, la cantidad de imágenes para activar la ejecución paralela. En este estudio se utilizaron dos equipos de cómputo con distintas características, en donde los resultados experimentales muestran que cuatro o seis hilos de ejecución son los óptimos para la ejecución del algoritmo de segmentación de las imágenes, dependiendo del equipo de cómputo, además, se puede reducir el tiempo de ejecución a dos terceras partes que haciéndolo de manera secuencial.

Como trabajo futuro se puede explorar en adaptar y ejecutar el algoritmo en las GPU, o también implementaciones basadas en la nube, esto con el fin de realizar la comparativa de los tiempos de ejecución con este trabajo, así mismo medir el *speedup* con dichas soluciones.

Agradecimientos

Agradecimientos al CONACYT por otorgar una beca al autor Yael Santana, con CVU 1137661, para así, poder enfocar los esfuerzos en el trabajo y estudio a realizar.

Referencias

[1] The Food and Agriculture Organization. *Acuicultura: Principales conceptos y definiciones*. Excerpted from 5th edition of the APA Publication Manual. 2003. URL: https://www.fao.org/3/x6941e/x6941e04.htm.

[2] Mohammadmehdi Saberioon y col. «Application of machine vision systems in aquaculture with emphasis on fish: state-of-the-art and key issues». En: *Reviews in Aquaculture* 9.4 (2017), págs. 369-387.

[3] Leonardo Mereles. *Preprocesamiento de imágenes digitales a través de su Transformada de Fourier*. 2012.

[4] Ramırez Q Juan y Chacón M Mario. «Redes neuronales artificiales para el procesamiento de imágenes, una revisión de la última década». En: *RIEE&C, Revista de Ingenierıa Eléctrica, Electrónica y Computación* 9.1 (2011), págs. 7-16.

[5] Damián A Álvarez, Germán A Holguín y Marta L Guevara. «Preprocesamiento de imágenes aplicadas a mamografıas digitales». En: *Scientia et technica* 12.31 (2006), págs. 1-6.

[6] Miguel Vera y col. «Técnicas de preprocesamiento de imágenes cardıacas: fundamentos y alcance». En: *Revista Latinoamericana de Hipertensión* 11.3 (2016), págs. 60-66.

[7] José F Valencia-Murillo, Daniel A Poveda-Sendales y Daniel F Valencia-Vargas. «Evaluación del impacto del preprocesamiento de imágenes en la segmentación del iris». En: *TecnoLógicas* 17.33 (2014), págs. 31-41.

[8] Thomas Bräunl y col. *Parallel image processing*. Springer Science & Business Media, 2013.

[9] Erik Smistad y col. «Medical image segmentation on GPUs–A comprehensive review». En: *Medical image analysis* 20.1 (2015), págs. 1-18.

[10] PN Happ y col. «Multiresolution segmentation: a parallel approach for high resolution image segmentation in multicore architectures». En: *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences* 38.4 (2010), pág. C7.

[11] Dave Kuhlman. *A python book: Beginning python, advanced python, and python exercises*. Dave Kuhlman Lutz, 2009.

[12] Sebastian Raschka y Vahid Mirjalili. *Python machine learning: Machine learning and deep learning with Python, scikit-learn, and TensorFlow 2*. Packt Publishing Ltd, 2019.

[13] V Arévalo, J González y G Ambrosio. «La librerıa de visión artificial opencv. aplicación a la docencia e investigación». En: *Base Informática* 40 (2004), págs. 61-66.

[14] Marcelo Naiouf. «Procesamiento paralelo». Tesis doct. Universidad Nacional de La Plata, 2004.